Preprints of the 11th European Conference on Mobile Robots

ECMR 2023

September 4 – 7, 2023
Coimbra, Portugal

Editors:

Lino Marques
Ivan Marković

Coimbra, Portugal, 2023

Lino Marques
Institute of Systems and Robotics
Department of Electrical and Computer Engineering
University of Coimbra
Rua Silvio Lima – Polo II
3030-290 Coimbra
Portugal
lino@isr.uc.pt

Ivan Marković
Department of Control and Computer Engineering
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3
HR-10000 Zagreb
Croatia
ivan.markovic@fer.hr

ECMR 2023
Preprints of the 11th European Conference on Mobile Robots (ECMR 2023)
September 4–7, 2023
Coimbra, Portugal

Table of Contents

vii Welcome Message
viii Organisation
ix Committees
xi Keynote speakers

Day 1

Session 1: Welcome and Opening Session

Session 2: Oral Session I

1 Autonomous Navigation in Rows of Trees and High Crops with Deep Semantic Segmentation
Alessandro Navone, Mauro Martini, Andrea Ostuni, Simone Angarano, Marcello Chiaberge

6 A Map-Free LiDAR-Based System for Autonomous Navigation in Vineyards
Riccardo Bertoglio, Veronica Carini, Stefano Arrigoni, Matteo Matteucci

12 Surgical fine-tuning for Grape Bunch Segmentation under Visual Domain Shifts
Agnese Chiatti, Riccardo Bertoglio, Nico Catalano, Matteo Gatti, Matteo Matteucci

19 Multi-camera GPS-free Nonlinear Model Predictive Control strategy to traverse orchards
Antoine Villemazet, Adrien Durand-Petiteville, Viviane Cadenat

Session 3: Oral Session II

26 Learned Long-Term Stability Scan Filtering for Robust Robot Localisation in Continuously Changing Environments
Ibrahim Hroob, Sergi Molina, Riccardo Polvara, Grzegorz Cielniak, Marc Hanheide

34 GAFAR: Graph-Attention Feature-Augmentation for Registration - A Fast and Light-weight Point Set Registration Algorithm
Ludwig Mohr, Ismail Geles, Friedrich Fraundorfer

42 Revisiting Distribution-Based Registration Methods
Himanshu Gupta, Henrik Andreasson, Martin Magnusson, Simon Julier, Achim Lilienthal

48 Enhancing Door-Status Detection for Autonomous Mobile Robots during Environment-Specific Operational Use
Michele Antonazzi, Matteo Luperto, Nicola Basilico, N. Alberto Borghese

56 Self-supervised Learning for Fusion of IR and RGB Images in Visual Teach and Repeat Navigation
Xinyu Liu, Zdeněk Rozsypálek, Tomáš Krajník

Session 4: Keynote: Autonomous Flight of Tiny Drones, Guido de Croon

Session 5: Poster Spotlight Session I

63 White-box and Black-box Adversarial Attacks to Obstacle Avoidance in Mobile Robots
Inaki Rano, Anders Christensen

69 Evaluating Techniques for Accurate 3D Object Model Extraction through Image-based Deep Learning Object Detection and Point Cloud Segmentation
Alicia Mora, Alberto Mendez, Ramon Barber

76 TAICHI algorithm: Human-Like Arm Data Generation applied on Non-Anthropomorphic Robotic Manipulators for Demonstration
Adrian Prados, Blanca Lopez, Luis Moreno, Ramon Barber

83 Multi-Task Learning for Industrial Mobile Robot Perception Using a Simulated Warehouse Dataset
Dimitrios Arapis, Andrea Vallone, Milad Jami, Lazaros Nalpantidis

89 Artifacts Mapping: Multi-Modal Semantic Mapping Extension of Geometric Maps
Federico Rollo, Gennaro Raiola, Andrea Zunino, Nikolaos Tsagarakis, Arash Ajoudani

97 Dynamic Human-Aware Task Planner for Human-Robot Collaboration in Industrial Scenario
Alberto Gottardi, Matteo Terreran, Christoph Frommel, Manfred Schoenheits, Nicola Castaman, Stefano Ghidoni, Emanuele Menegatti

105 Decentralized Market-Based Task Allocation Algorithm for a Fleet of Industrial Mobile Robots
João Neto de Carvalho de Andrade Tavares, Alberto Vale, Rodrigo Ventura

111 Distributed 3D-Map Matching and Merging on Resource-Limited Platforms using Tomographic Features
Halil Utku Unlu, Anthony Tzes, Prashanth Krishnamurthy, Farshad Khorrami

117 A Temporal Perspective n-Point Problem with Model Uncertainties for Cooperative Pose Estimation in a Heterogeneous Robot Team
Florian Steidle, Simon Boche, Wolfgang Stürzl, Rudolph Triebel

124 Human-centered Benchmarking for Socially-compliant Robot Navigation
Iaroslav Okunevich, Vincent Hilaire, Stephane Galland, Olivier Lamotte, Liubov Shilova, Yassine Ruichek, Zhi Yan

131 Improved path planning algorithms for non-holonomic autonomous vehicles in industrial environments with narrow corridors: Roadmap Hybrid A* and Waypoints Hybrid A*
Alessandro Bonetti, Simone Guidetti, Lorenzo Sabattini

138 Assisted Localization of MAVs for Navigation in Indoor Environments Using Fiducial Markers
André Kirsch, Malte Riechmann, Matthias Koenig

144 Stable Yaw Estimation of Boats from the Viewpoint of UAVs and USVs
Benjamin Kiefer, Timon Höfer, Andreas Zell

150 Graph-based Simultaneous Localization and Mapping with incorporated dynamic object motion
Peter Aerts, Peter Slaets, Eric Demeester

157 Visual-LiDAR Odometry and Mapping with Monocular Scale Correction and Motion Compensation
Hanyu Cai, Ni Ou, Junzheng Wang

Session 6: Poster Session I

Day 2

Session 7: Oral Session III

164 Difficulty-Aware Time-Bounded Planning under Uncertainty for Large-Scale Robot Missions
Michal Staniaszek, Lara Brudermüller, Raunak Bhattacharyya, Bruno Lacerda, Nick Hawes

171 Robust Multi-Agent Pickup and Delivery with Delays
Giacomo Lodigiani, Nicola Basilico, Francesco Amigoni

179 On Improvement Heuristic to Solutions of the Close Enough Traveling Salesman Problem in Environments with Obstacles
Jindřiška Deckerová, Kristýna Kučerová, Jan Faigl

185 Context-Conditional Navigation with a Learning-Based Terrain- and Robot-Aware Dynamics Model
Suresh Guttikonda, Jan Achterhold, Haolong Li, Joschka Boedecker, Joerg Stueckler

192 Social Robot Navigation through Constrained Optimization: a Comparative Study of Uncertainty-based Objectives and Constraints
Timur Akhtyamov, Aleksandr Kashirin, Aleksey Postnikov, Gonzalo Ferrer

Session 8: Oral Session IV

200 Delta filter – robust visual-inertial pose estimation in real-time: A multi-trajectory filter on a spherical mobile mapping system
Fabian Arzberger, Fabian Wiecha, Jasper Zevering, Julian Rothe, Dorit Borrmann, Sergio Montenegro, Andreas Nüchter

208 Dataset Generation for Deep Visual Navigation in Unstructured Environments
Yoshinobu Uzawa, Shigemichi Matsuzaki, Hiroaki Masuzawa, Jun Miura

214 Towards camera parameters invariant monocular depth estimation in autonomous driving
Karlo Koledić, Ivan Marković, Ivan Petrović

221 Synthetic Data-based Detection of Zebras in Drone Imagery
Elia Bonetto, Aamir Ahmad

229 Navigating in 3D Uneven Environments through Supervoxels and Nonlinear MPC
Fetullah Atas, Grzegorz Cielniak, Lars Grimstad

Session 9: Keynote: A Robot Web for Many-Device Localisation and Planning, Andrew Davison

Session 10: Poster Spotlight Session II

237 Ant Colony Optimization for Retail based Capacitated Vehicle Routing Problem with Pickup and Delivery for Mobile Robots
Agha Ali Haider Qizilbash, Anoj Kumar Yadav, Kevin Bregler, Werner Kraus

243 Direct Object Reconstruction on RGB-D Images in Cluttered Environment
Mikołaj Zieliński, Dominik Belter

250 Robust Perception Skills for Autonomous Elevator Operation by Mobile Robots
Steffen Müller, Benedict Stephan, Trsitan Müller, Horst-Michael Gross

257 Scalable Evaluation Pipeline of CNN-based perception for Robotic Sensor Data under different Environment Conditions
Naeem Iqbal, Mark Niemeyer, Christoph Krause, Joachim Hertzberg

263 Teach and Repeat and Wheel Calibration for LiDAR-equipped Omnidirectional Drive Robots
Sebastián Bedín, Javier Civera, Matias Alejandro Nitsche

269 Adaptive Compliant Robot Control with Failure Recovery for Object Press-Fitting
Ekansh Sharma, Christoph Henke, Alex Mitrevski, Paul G. Plöger

277 Graph-based LiDAR-Inertial SLAM Enhanced by Loosely-Coupled Visual Odometry
Vsevolod Hulchuk, Jan Bayer, Jan Faigl

285 Symmetric Object Pose Estimation via Flexible Modular CNN
Simone Mentasti, Claudia Speranza, Matteo Matteucci

292 Learning State-Space Models for Mapping Spatial Motion Patterns
Junyi Shi, Tomasz Piotr Kucner

298 Monocular Person Localization and Lidar Fusion for Social Navigation
Sedat Dogru, Carlos A. Silva, Lino Marques

305 Towards Data-Driven Discovery of Governing Swarm Robots Flocking Rules
Belkacem Khaldi, Erhan Ege Keyvan, Mehmet Şahin, Ali Emre Turgut, Erol Sahin

311 Where to Place a Pile?
Miroslav Kulich, David Woller, Sarah Carmesin, Masoumeh Mansouri

318 An EKF-based Multi-Object Tracking Framework for a Mobile Robot in a Precision Agriculture Scenario
Andrea Arlotta, Martina Lippi, Andrea Gasparri

324 Stereo Visual Localization Dataset Featuring Event Cameras
Antea Hadviger, Vlaho-Josip Štironja, Igor Cvišić, Sacha Vražić, Ivan Marković, Ivan Petrović

330 Analyzing Data Efficiency and Performance of Machine Learning Algorithms for Assessing Low Back Pain Physical Rehabilitation Exercises
Aleksa Marusic, Louis Annabi, Sao Mai Nguyen, Adriana Tapus

Session 11: Poster Session II

Day 3

Session 12: Keynote: Inductive Biases for Robot Reinforcement Learning, Jan Peters

Session 13: Oral Session V

336 Enhanced Visual Predictive Control Scheme for Mobile Manipulator
Hugo Bildstein, Adrien Durand-Petiteville, Viviane Cadenat

343 Motion Planning for Multi-legged Robots using Levenberg-Marquardt Optimization with Bézier Parametrization
David Valouch, Jan Faigl

348 Social APF-RL: Safe Mapless Navigation in Unknown & Human-Populated Environments
S. Batuhan Vatan, Kemal Bektaş, H. Işıl Bozma

354 A new flex-sensor-based umbilical-length management system for underwater robots
Ornella Tortorici, Cedric Anthierens, Vincent Hugel

Session 14: Oral Session VI + Best WS papers

360 Multi-Formation Planning and Coordination for Object Transportation
Weijian Zhang, Charlie Street, Masoumeh Mansouri

367 Impact of UAV Propellers on gas plume tracking
Rui Baptista, Hugo Magalhães, Lino Marques

374 Finite-Time Standoff Target Tracking in The Presence of Wind
Pallov Anand, Pranav Niturkar, A. Pedro Aguiar, Rajat Agarwal, Manav Mishra, P. B. Sujit

380 Late-Fusion Multimodal Human Detection based on RGB and Thermal Images for Robotic Perception
Elísio Sousa, Kennedy Mota, Iago Gomes, Luís Garrote, Denis Wolf, Cristiano Premebida

387 **Index of Authors**

Welcome Message

It is our pleasure to welcome you to the 11th European Conference on Mobile Robots – ECMR 2023, which is held in Coimbra, Portugal on September 4–7, 2023. ECMR is a biannual European forum, internationally open, allowing researchers to become acquainted with the latest accomplishments and innovations in advanced mobile robotics and mobile human-robot systems. ECMR especially seeks to attract young researchers to present their work to an international audience. The first ECMR meeting was held in September 2003 in Radziejowice, Poland, followed by ECMR in September 2005, while previous edition of ECMR was organized virtually in September 2021 in Bonn, Germany due to unfortunate pandemic circumstances. Now, we are honored to be able to organize ECMR in person after twenty years of its inception.

Papers submitted to ECMR 2023 were co-authored by 213 authors from 28 countries, and in our view, this serves as a testimony to the appeal of the conference. Each paper was evaluated by expert reviewers and 56 of them have been accepted by the Program Committee. These papers are included in the proceedings and will be presented at the conference. They cover a wide spectrum of research topics in mobile robotics: 3D perception, navigation, path planning and tracking, SLAM, mapping and exploration, cooperative multi-robot systems, deep learning, various service applications, etc. We also appreciate workshops organizers who have enriched the conference program by organizing the following workshops: "Robotics in Agriculture and Forestry," "Ethical, Legal and User Perspectives on Social and Assistive Robots," "Robotic Perception and Situation Awareness in Real-World Applications," and "Deploying Mobile Robots in Unconstrained Real-World Environments."

We are especially proud to welcome our distinguished keynote speakers: Professor Guido de Croon from the Delft University of Technology, Netherlands, who will give a talk titled “Autonomous flight of tiny drones”, Professor Andrew Davison from Imperial College London, United Kingdom, who will give the talk titled “A Robot Web for Many-Device Localisation and Planning”, and Professor Jan Peters from the Technical University of Darmstadt, Germany, who will give the talk titled “Inductive Biases for Robot Reinforcement Learning.” We must thank the IEEE Robotics and Automation Society, for its technical sponsorship, the Institute of Systems and Robotics, for the logistics support, and the University of Coimbra, for providing the necessary facilities to host the conference.

Finally, our sincere thanks are due to all people whose hard work made this conference possible. First and foremost, we would like to thank the members of the Organizing Committee and the Program Committee for their outstanding work. Our special thanks go to the authors for submitting their work to ECMR 2023 and to the reviewers for their time and effort in evaluating the submissions. The results of their joint work are visible in the program of ECMR 2023. It is now up to all of us to make ECMR 2023 a great success and a memorable event by participating in the technical program and by enjoying the beauty and history of Coimbra, as well as traditions and culture of Portugal!

Lino Marques
Ivan Marković

Organisation

Institute of Electrical and Electronics Engineers – IEEE

IEEE Robotics & Automation Society – RAS

UNIVERSIDADE DE COIMBRA

University of Coimbra, Portugal

Department of Electrical Engineering and Computing
University of Coimbra, Portugal

Institute of Systemas and Robotics
University of Coimbra, Portugal

Committees

General Chair
Lino Marques (Portugal)

Programme Chair
Ivan Marković (Croatia)

Workshops and Tutorials Chair
Rui P. Rocha (Portugal)

Publication Chair
Paulo Menezes (Portugal)

Exhibition Chair
Cristiano Premebida (Portugal)

Local Organizing Committee
Helder Araújo (Portugal)
Jorge Miranda Dias (Portugal)
Pedro Moura (Portugal)
Urbano Nunes (Portugal)

ECMR Conference Board

Permanent Members
Wolfram Burgard (Germany)
Ivan Petrović (Croatia)
Primo Zingaretti (Italy)

Rotating Members
Libor Preucil (Czech Republic)
Miroslav Kulich (Czech Republic)
Sven Behnke (Germany)

Advisory Committee
Adam Borkowski (Poland)
Juan Andrade Cetto (Spain)
Tom Duckett (UK)
Udo Frese (Germany)
Emanuele Frontoni (Italy)
Horst-Michael Gross (Germany)
Miroslav Kulich Czech (Czech Republic)
Achim J. Lilienthal (Sweden)
Adriana Tapus (France)

Programme Committee

Lino Marques
Ivan Marković
Sven Behnke
Todor Stoyanov
Luis Merino
Luis Paulo Reis
Jan Faigl
Björn Hein
Antonio Sgorbissa
Julien Marzat
Nick Hawes
Ruediger Dillmann
Alberto Finzi
Shai Arogeti
Jorge Dias
Robert Cupec
Antonio Chella
Emmanuel Karlo Nyarko
Emanuele Frontoni
Thomas Wiemann
Tim Laue
Adriana Tapus
Andreas Zell
Fabio Morbidi
Syed Atif Mehdi
Primo Zingaretti
Dominik Belter
Paul Checchin
Javier Civera
Jun Miura
Kenji Koide
Giovanni Muscato
Robert Penicka
Matteo Luperto
Andreu Corominas Murtra
Dongbing Gu
Zdenko Kovacic
Vojtech Vonasek
Vladyslav Usenko
Sebastien Lengagne
Andreas Nuechter
Alberto Pretto
Pedro U. Lima
Carmine Recchiuto
Tomáš Krajník
Giovanni Indiveri
Arne Roennau
Jose-Luis Blanco-Claraco
Polina Kurtser
Pablo Bustos
Miroslav Kulich
Stefano Ghidoni
Rudolph Triebel
Matthias Koenig
Rui P. Rocha
Anas Fattouh
Estela Bicho
António Pedro Aguiar
Joao Borges de Sousa
Francesco Amigoni
Eric Demeester
Cyrill Stachniss
Rui P. Rocha
Domenico G. Sorrenti
Tomi Westerlund
Andrej Gams
Karsten Berns
Monica Sileo
Angel Rodriguez Castaño
Jens Behley
Darius Burschka
Joschka Boedecker
Dmitriy Shutin
Farshad Arvin
Nicola Basilico
Ramón Barber
Nikos Tsagarakis
Vincent Hugel
Ulrich Rueckert
Fabio Bonsignorio
Mercedes Paoletti
Antonio Pascoal
Lounis Adouane
Antonio Agudo
Ivana Palunko
Thierry Fraichard
Alessandro Freddi
Elmar Rueckert
Christophe Grand
H. Isil Bozma
Karel Košnar
Paul G. Plöger
Markus Vincze
Libor Přeučil
Andrea Monteriù
Ketao Zhang
Horst-Michael Gross
Pablo De Cristóforis
Nabil Aouf
Antonios Tsourdos
Ulrike Thomas
Adriano Mancini
Roberto Capobianco
Marko Orsag
Rui P. Rocha
Federico Magistri
Nived Chebrolu
Marija Popović
Elias Marks
Karoline Heiwolt
Jim Torresen
Tobias Mahler
Diana Saplacan
Rui P. Rocha
Cristiano Premebida
Cunjia Liu
Cédric Pradalier
Jingjing Jiang

Preprints created by:
Antea Hadviger (Croatia)

Keynote speakers

Autonomous flight of tiny drones

Guido de Croon

Tiny drones are promising for many applications, such as search-and-rescue, greenhouse monitoring, or keeping track of stock in warehouses. Since they are small, they can fly in narrow areas. Moreover, their light weight makes them very safe for flight around humans. However, making such tiny drones fly completely by themselves is an enormous challenge. Most approaches to Artificial Intelligence for robotics have been designed with self-driving cars or other large robots in mind – and these are able to carry many sensors and ample processing. In my talk, I will argue that a different approach is necessary for achieving autonomous flight with tiny drones. In particular, I will discuss how we can draw inspiration from flying insects, and endow our drones with similar intelligence. Examples include the fully autonomous "DelFly Explorer", a 20-gram flapping wing drone, and swarms of CrazyFlie quadrotors of 30 grams able to explore unknown environments and find gas leaks. Moreover, I will discuss the promises of novel neuromorphic sensing and processing technologies, illustrating this with recent experiments from our lab. Finally, I will discuss how insect-inspired robotics can allow us to gain new insights into nature. I illustrate this with a recent study, in which we proposed a new theory on how flying insects determine the gravity direction.

Guido de Croon received his M.Sc. and Ph.D. in the field of Artificial Intelligence (AI) at Maastricht University, the Netherlands. His research interest lies with computationally efficient, bio-inspired algorithms for robot autonomy, with an emphasis on computer vision. Since 2008 he has worked on algorithms for achieving autonomous flight with small and light-weight flying robots, such as the DelFly flapping wing MAV. In 2011-2012, he was a research fellow in the Advanced Concepts Team of the European Space Agency, where he studied topics such as optical flow based control algorithms for extraterrestrial landing scenarios. After his return at TU Delft, his work has included fully autonomous flight of a 20-gram DelFly, a new theory on active distance perception with optical flow, a swarm of tiny drones able to explore unknown environments, and neuromorphic sensing and processing. Currently, he is Full Professor at TU Delft and scientific lead of the Micro Air Vehicle lab (MAVLab) of Delft University of Technology.

A Robot Web for Many-Device Localisation and Planning

Andrew Davison

Safe and useful robots for complex environments must use their on-board sensors and computation to map, understand and localise within their surroundings, and we can envision a future where many such devices, with different functions and made by different companies, should operate in the same space. Is there a more modular way for this to work than all devices needing to use the same unified cloud-based "maps" system?

I will present and demonstrate our Robot Web proposal for distributed solutions to many robot localisation and planning based on per-device local computation and storage, and peer-to-peer communication between heterogenous devices via standardised open protocols. Our method uses Gaussian Belief Propagation-based distributed inference on full non-linear factor graph, and is highly robust and scalable while remaining simple and modular.

Andrew Davison is Professor of Robot Vision and Director of the Dyson Robotics Laboratory at Imperial College London. His long-term research focus is on SLAM (Simultaneous Localisation and Mapping) and its evolution towards general 'Spatial AI':computer vision algorithms which enable robots and other artificial devices to map, localize within and ultimately understand and interact with the 3D spaces around them. With his research group and collaborators he has consistently developed and demonstrated breakthrough systems, including MonoSLAM, KinectFusion, SLAM++ and CodeSLAM, and recent prizes include Best Paper at ECCV 2016, Best Paper Honourable Mention at CVPR 2018 and the Helmholtz Prize at ICCV 2021. He has also had strong involvement in taking this technology into real applications, in particular through his work with Dyson on the design of the visual mapping system inside the Dyson 360 Eye robot vacuum cleaner and as co-founder of applied SLAM start-up SLAMcore. He was elected Fellow of the Royal Academy of Engineering in 2017.

Inductive Biases for Robot Reinforcement Learning

Jan Peters

Autonomous robots that can assist humans in situations of daily life have been a long-standing vision of robotics, artificial intelligence, and cognitive sciences. A first step towards this goal is to create robots that can learn tasks triggered by environmental context or higher-level instruction. However, learning techniques have yet to live up to this promise as only few methods manage to scale to high-dimensional manipulator or humanoid robots. In this talk, we investigate a general framework suitable for learning motor skills in robotics which is based on the principles behind many analytical robotics approaches. To accomplish robot reinforcement learning from just few trials, the learning system can no longer explore all learn-able solutions but has to prioritize one solution over others – independent of the observed data. Such prioritization requires explicit or implicit assumptions, often called 'induction biases' in machine learning. Extrapolation to new robot learning tasks requires induction biases deeply rooted in general principles and domain knowledge from robotics, physics and control. Empirical evaluations on a several robot systems illustrate the effectiveness and applicability to learning control on an anthropomorphic robot arm. These robot motor skills range from toy examples (e.g., paddling a ball, ball-in-a-cup) to playing robot table tennis, juggling and manipulation of various objects.

Jan Peters is a full professor (W3) for Intelligent Autonomous Systems at the Computer Science Department of the Technische Universitaet Darmstadt since 2011, and, at the same time, he is the dept head of the research department on Systems AI for Robot Learning (SAIROL) at the German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) since 2022. He is also a founding research faculty member of the Hessian Center for Artificial Intelligence. Jan Peters has received the Dick Volz Best 2007 US PhD Thesis Runner-Up Award, the Robotics: Science & Systems – Early Career Spotlight, the INNS Young Investigator Award, and the IEEE Robotics & Automation Society's Early Career Award as well as numerous best paper awards. In 2015, he received an ERC Starting Grant and in 2019, he was appointed IEEE Fellow, in 2020 ELLIS fellow and in 2021 AAIA fellow. Despite being a faculty member at TU Darmstadt only since 2011, Jan Peters has already nurtured a series of outstanding young researchers into successful careers. These include new faculty members at leading universities in the USA, Japan, Germany, Finland and Holland, postdoctoral scholars at top computer science departments (including MIT, CMU, and Berkeley) and young leaders at top AI companies (including Amazon, Boston Dynamics, Google and Facebook/Meta). Jan Peters has studied Computer Science, Electrical, Mechanical and Control Engineering at TU Munich and FernUni Hagen in Germany, at the National University of Singapore (NUS) and the University of Southern California (USC). He has received four Master's degrees in these disciplines as well as a Computer Science PhD from USC. Jan Peters has performed research in Germany at DLR, TU Munich and the Max Planck Institute for Biological Cybernetics (in addition to the institutions above), in Japan at the Advanced Telecommunication Research Center (ATR), at USC and at both NUS and Siemens Advanced Engineering in Singapore. He has led research groups on Machine Learning for Robotics at the Max Planck Institutes for Biological Cybernetics (2007-2010) and Intelligent Systems (2010-2021).

Preprints

Autonomous Navigation in Rows of Trees and High Crops with Deep Semantic Segmentation

Alessandro Navone[1], Mauro Martini[1], Andrea Ostuni[1], Simone Angarano[1] and Marcello Chiaberge[1]

***Abstract*— Segmentation-based autonomous navigation has recently been proposed as a promising methodology to guide robotic platforms through crop rows without requiring precise GPS localization. However, existing methods are limited to scenarios where the centre of the row can be identified thanks to the sharp distinction between the plants and the sky. However, GPS signal obstruction mainly occurs in the case of tall, dense vegetation, such as high tree rows and orchards. In this work, we extend the segmentation-based robotic guidance to those scenarios where canopies and branches occlude the sky and hinder the usage of GPS and previous methods, increasing the overall robustness and adaptability of the control algorithm. Extensive experimentation on several realistic simulated tree fields and vineyards demonstrates the competitive advantages of the proposed solution.**

I. INTRODUCTION

In recent years, precision agriculture has pushed the boundaries of technology to optimize crop production, improve the efficiency of farming operations, and reduce waste [1]. Modern farming systems must be able to extract synthetic key information from the environment, take or suggest optimal decisions based on that information, and execute them with high precision and timing. Deep learning techniques have shown great potential in realizing these systems by analyzing data from multiple sources, allowing for large-scale, high-resolution monitoring, and providing detailed insights for both human and robotic agents. The most recent advancements in deep learning also provide competitive advantages for real-world applications, such as model optimization for fast inference on low-power embedded hardware [2], [3] and generalization to unseen data [4], [5], [6]. At the same time, progress in service robotics has enabled autonomous mobile agents to embody AI perception systems and work in synergy with them to accomplish complex tasks in unstructured environments [7].

In particular, row-based crops are among the most studied applications (they constitute more than 75% of all planted acres of cropland across the USA [8]). In this scenario, research spans localization[9], path planning [10], navigation [11], monitoring[12], harvesting [13], spraying [14], and vegetative assessment [15], [16]. A particularly challenging situation occurs when standard localization methods, like GPS, fail to reach the desired precision due to unfavorable weather conditions or line-of-sight obstruction. That is the case, for example, of dense tree canopies, as shown in a simulated pear orchard in Figure 1.

Fig. 1. The proposed SegMin and SegMinD algorithms allow to precisely guide an autonomous mobile robot through a dense tree row solely using an RGB-D camera. A pear crop row in Gazebo is shown in the picture.

Previous works have proposed position-agnostic vision-based navigation algorithms for row-based crops. A first vision-based approach was proposed in [17] using mean-shift clustering and the Hough transform to segment RGB images and generate the optimal central path. Later, [18] achieved promising results using multispectral images and simply thresholding and filtering on the green channel. Recently, deep-learning approaches have been successfully applied to the task. [19] proposed a classification-based approach in which a model predicts the discrete action to perform. In contrast, [20] proposed combining a segmentation model and a proportional controller to align the robot to the center of the row. Finally, a different approach was tested in [11] with an end-to-end controller based on deep reinforcement learning. Although these systems proved effective in their testing scenarios, they have only been applied in simple crops where a full view of the sky favors both GPS receivers [21] and vision-based algorithms [22].

This work tackles a more challenging scenario in which dense canopies partially or totally cover the sky, and the GPS signal is very weak. We design a navigation algorithm based on semantic segmentation that exploits visual perception to estimate the center of the crop row and align the robot trajectory to it. The segmentation masks are predicted by a deep learning model designed for real-time efficiency and trained on realistic synthetic images. The proposed navigation algorithm improves on previous works being adaptive to different terrains and crops, including dense

[1] Department of Electronics and Telecommunications, Politecnico di Torino, 10129, Torino, Italy. {firstname.lastname}@polito.it

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

canopies. We conduct extensive experimentation in simulated environments for multiple crops. We compare our solution with previous state-of-the-art methodologies, demonstrating that the proposed navigation system is effective and adaptive to numerous scenarios.

The main contributions of this work can be summarized as follows:

- we present two variants of a novel approach for segmentation-based autonomous navigation in tall crops, designed to tackle challenging and previously uncovered scenarios;
- we test the resulting guidance algorithm on previously unseen plant rows scenarios such as high trees and pergola vineyards.
- we compare the new method with state-of-the-art solutions on straight and curved vineyards, demonstrating an enhanced general and robust behavior.

The next sections are organized as follows: Section II presents the proposed deep-learning-based control system for vision-based position-agnostic autonomous navigation in row-based crops, from the segmentation model to the controller. Section III describes the experimental setting and reports the main results for validating the proposed solution divided by sub-system. Finally, Section IV draws conclusive comments on the work and suggests interesting future directions.

II. METHODOLOGY

This work proposes a real-time control algorithm with two variants to navigate high-vegetation orchards and arboriculture fields and improve the approach presented in [20]. The proposed system avoids exploiting the GPS signal, which can lack accuracy due to signal reflection and mitigation due to vegetation.

The working principle of the proposed control algorithms is straightforward and exploits only the RGB-D data. Both the proposed solutions consist of four main steps:

1) Semantic segmentation of the input RGB frame.
2) Processing of the output segmentation mask using depth frame data.
3) Searching for the direction which leads the mobile platform towards the end of the row.
4) Generating linear and angular velocity commands to input the mobile robot.

Nonetheless, the two proposed methods differ only for steps 2 and 3 in employing the depth frame data and in the generation of the path which the robot should follow. In contrast, the segmentation technique 1 and the command generation 4 are carried out similarly. A schematic representation of the proposed pipeline is described in Figure 2.

As in [20] a first step, an RGB frame $\mathbf{X}^t_{rgb} \in \mathbb{R}^{h\times w\times c}$ and a depth map $\mathbf{X}^t_d \in \mathbb{R}^{h\times w}$ are acquired by a camera placed on the front of the mobile platform at each instant t, where h and w are the width of the frame and c is the number of channels. The received RGB data is then fed to a segmentation neural network model H_{seg}, which outputs a binary segmentation mask bringing the semantic information of the input frame.

$$\hat{\mathbf{X}}^t_{seg} = H\left(\mathbf{X}^t_{rgb}\right) \tag{1}$$

where $\hat{\mathbf{X}}^t_{seg}$ is the estimated segmentation mask. Moreover, the segmentation masks of the last N time instants $\{t-N,\dots,t\}$ are fused to obtain more robust information.

$$\hat{\mathbf{X}}^t_{CumSeg} = \bigcup_{j=t-N}^{t} \hat{\mathbf{X}}^j_{seg} \tag{2}$$

where $\hat{\mathbf{X}}^t_{CumSeg}$ is the cumulative segmentation mask and the operator $\bigcup$ represent the logical bitwise *OR* operation over the last N binary frames.

Additionally, the depth map $\mathbf{X}^t_d$ is now used to consider the segmented regions between the camera position and a given depth threshold d_{th} to remove useless information given by far vegetation, which is irrelevant to control the robot's movement.

$$\hat{\mathbf{X}}^t_{segDepth}{}_{\substack{i=0,\dots,h\\ j=0,\dots,w}}(i,j) = \begin{cases} 0, \text{ if } \hat{\mathbf{X}}^t_{CumSeg(i,j)} \cdot \hat{\mathbf{X}}^t_{d(i,j)} > d_{th} \\ 1, \text{ if } \hat{\mathbf{X}}^t_{CumSeg(i,j)} \cdot \hat{\mathbf{X}}^t_{d(i,j)} \leq d_{th} \end{cases} \tag{3}$$

where $\hat{\mathbf{X}}_{segDepth}$ is the resulting intersection between the cumulative segmentation frame and the depth map cut at a distance threshold d_{th}.

Henceforth the proposed algorithm forks in two variants, *SegMin* and *SegMinD*, respectively described in II-A and II-B.

A. SegMin

The first variant improves the approach proposed in [20]. After processing the segmentation mask, a sum over the column is performed to obtain a histogram $\mathbf{h} \in \mathbb{R}^w$, quantifying how much vegetation is present on each column. Hereafter, a moving average on a window of n elements is performed over the array to smooth the values and make the control more robust to punctual noise derived from the previous passages. Ideally, the minimum of this histogram x_h corresponds to the regions where less vegetation is present and, therefore, identifies the desired central path inside the crop row. If more global minimum points are present (i.e., there is a region where no vegetation is detected), the mean of the considered points is considered to be the global minimum and, in consequence, the continuation of the row.

B. SegMinD

The second proposed approach consists of a variant of the previous algorithm, devised for wide rows with tall and thick canopies, which in the previous case would generate an ambiguous global minimum due to the constant presence of vegetation above the robot. This variant multiplies the previously processed segmentation mask for the normalized inverted depth datum.

$$\hat{\mathbf{X}}^t_{depthInv} = \hat{\mathbf{X}}^t_{segDepth} \bigcap \left(1 - \frac{\mathbf{X}^t_d}{d_{th}}\right) \tag{4}$$

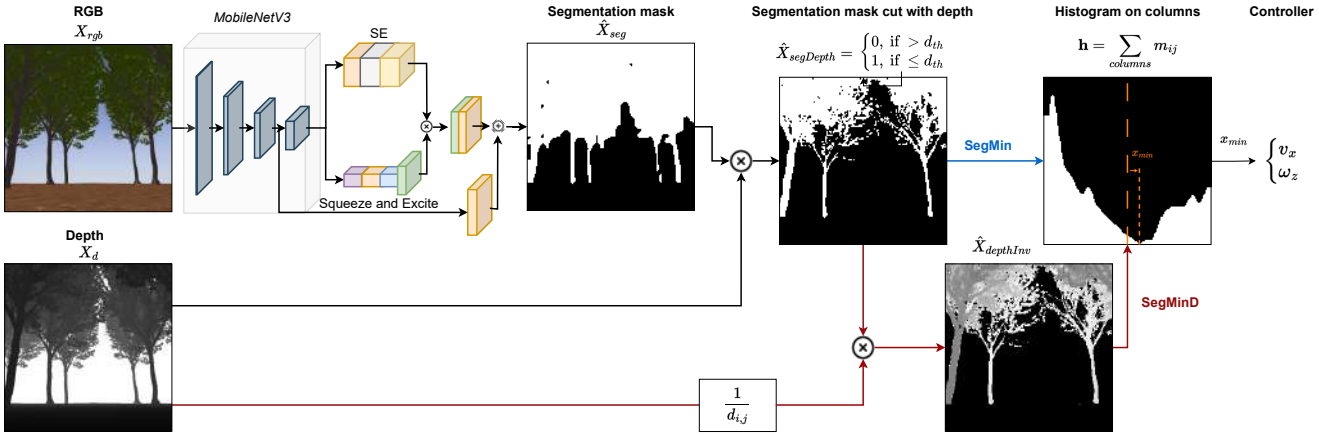


Fig. 2. Scheme of the overall proposed navigation pipeline. The RGB image is fed into the segmentation network, thus the predicted segmentation mask $\hat{\mathbf{X}}^t_{seg}$ is refined using the depth frame to obtain $\hat{\mathbf{X}}^t_{segDepth}$. The blue arrow refers to the SegMin variant, and red arrows refer to the SegMinD variant to compute the sum histogram over the mask columns. Images are taken from navigation in the tall trees simulation world.

where $\hat{\mathbf{X}}^t_{depthInv}$ is the result of the element-wise multiplication, represented by $\bigcap$, between the binary mask $\hat{\mathbf{X}}^t_{segDepth}$ and the depth frame $\hat{\mathbf{X}}^t_d$ normalized over the depth threshold d_{th}. As in the previous case, the sum over the column is performed to obtain the 1D array $\mathbf{h}$ and, later on, the smoothing through a moving average. The introduced modification allows the closer elements to exert a greater influence on identifying the row direction.

C. Segmentation Network

We adopt the same network used in previous works on real-time crop segmentation [20], [6]. The model consists of a MobilenetV3 backbone for feature extraction and an efficient LR-ASPP segmentation head [23]. In particular, the LR-ASPP leverages effective modules such as depth-wise convolutions, channel-wise attention, and residual skip connections to provide an effective trade-off between accuracy and inference speed. The model is trained with a similar procedure to [6] on the AgriSeg dataset[1]. Further details on the training strategy and hyperparameters are provided in Section III.

D. Robot heading control

The objective of the controller pipeline consists in keeping the mobile platform at the center of the row, which, in this work, is considered equivalent to keeping the row center in the middle of the camera frame. Therefore, as defined in the previous step, the minimum of the histogram should be centered in the frame width. The distance d from the center of the frame and the minimum is defined as:

$$d = x_h - \frac{w}{2} \tag{5}$$

The linear and angular velocities are then generated through custom functions similarly as in [24].

$$v_x = v_{x,max}\left[1 - \frac{d^2}{\left(\frac{w}{2}^2\right)}\right] \tag{6}$$

[1]https://pic4ser.polito.it/AgriSeg

$$\omega_z = -k_{\omega_z} \cdot \omega_{z,max} \cdot \frac{d^2}{w^2} \tag{7}$$

where $v_{x,max}$ and $\omega_{z,max}$ are respectively the maximum achievable linear and angular velocities and k_{ω_z} is the angular gain which regulates the speed of the response. In order to avoid abrupt changes in the robot's motion, the final velocities $\bar{v}_x$ and $\bar{\omega}_z$ commands are smoothed with an Exponential Moving Average (EMA) as:

$$\begin{bmatrix} \bar{v}^t_x \\ \bar{\omega}^t_z \end{bmatrix} = (1-\lambda)\begin{bmatrix} \bar{v}^{t-1}_x \\ \bar{\omega}^{t-1}_z \end{bmatrix} + \lambda \begin{bmatrix} v^t_x \\ \omega^t_z \end{bmatrix} \tag{8}$$

where t is the time step and λ is a chosen weight.

III. EXPERIMENTS AND RESULTS

A. Simulation Environment

The proposed control algorithm was tested through the use of Gazebo[2] simulation software. The software was selected because of its compatibility with ROS 2 and can incorporate plugins that simulate sensors, such as cameras. A Clearpath Jackal model was utilized to assess the algorithm's effectiveness. The URDF file, available through Clearpath Robotics, contains all the necessary information regarding the mechanical structure and joints of the robot. During the simulation, an Intel Realsense D435i plugin was utilized, positioned 20 cm in front of the robot's center, and tilted 15° upwards. This positioning gave the camera a better view of the upper branches of trees.

The navigation algorithm was tested in four different custom simulation environments: a common vineyard, a pergola vineyard characterized by vine poles and shoots above the row, a pear field constituted by small size trees, and a high trees field where canopies of the trees are merged above the row. Each simulated field adopts a different terrain, miming the irregularity of uneven terrain. The detailed measurements of the simulation world are described in Table I.

[2]https://gazebosim.org

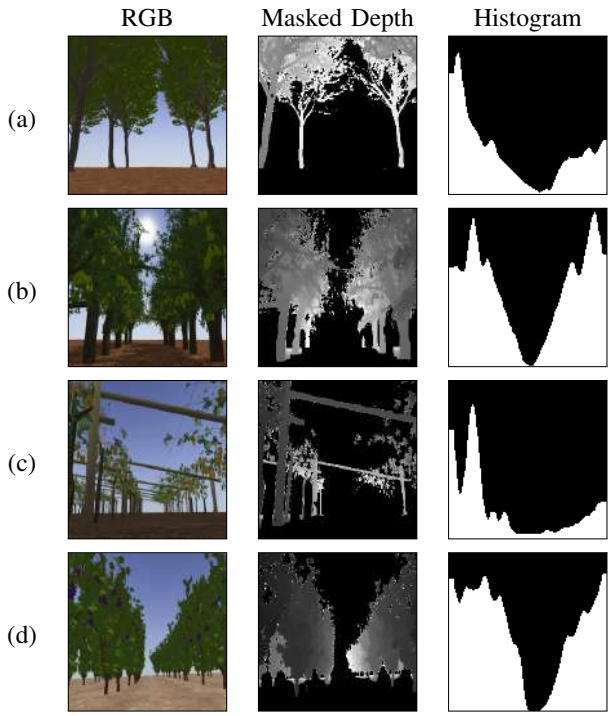


Fig. 3. Sample outputs of the proposed SegMinD algorithm for High Trees (a), Pear Trees (b), Pergola Vineyard (c), and Vineyard (d). Predicted segmentation masks are refined cutting values exceeding a depth threshold. The sum over mask columns provides the histograms used to identify the center of the row as its global minimum.

TABLE I

SIZE OF THE DIFFERENT SIMULATED CROPS, REFERRING TO THE AVERAGE VALUES OF THE DISTANCE BETWEEN ROWS, THE DISTANCE BETWEEN PLANTS ON THE ROW, AND THE HEIGHTS OF THE PLANTS.

Gazebo worlds	Rows distance [m]	Plant distance [m]	Height [m]
Common vineyard	1.8	1.3	2.0
Pergola vineyard	6.0	1.5	2.9
Pear field	2.0	1.0	2.9
High trees field	7.0	5.0	12.5

During the experimental part of this work, we consider frame dimensions equal to $(h,w) = (224,224)$, which is the same size as the input and the output of the neural network model, with the number of channels $c = 3$. The maximum linear velocity has been fixed to $v_{x,max} = 0.5m/s$, and the maximum angular velocity has been fixed to $\omega_{z,max} = 1rad/s$. The angular velocity gain $\omega_{z,gain}$ has been fixed to 0.01, and the EMA buffer size has been fixed to 3. The depth threshold has been changed according to the various crops. In particular, it has been empirically fixed to 5 m in the case of vineyards, while it was increased to 8 m for pear trees and pergola vineyards and 10 m for tall trees according to the average distance from the rows in the diverse fields.

B. Segmentation Network Training and Evaluation

We train the crop segmentation model using a subset of the AgriSeg segmentation dataset [6]. In particular, for the High Tree and Pear crops, we train on Generic Tree splits 1 and 2, and on Pear; for Vineyards, we train on Vineyard and Pergola Vineyard (note that the testing environments are different from the ones from which the training samples are generated). In both cases, the model is trained for 50 epochs with Adam optimizer and learning rate 3×10^{-4}. We apply data augmentation by randomly applying cropping, flipping, greyscaling, and random jitter to the images. Our experimentation code is developed in Python 3 using TensorFlow as the deep learning framework. We train models starting from ImageNet pretrained weights, so the input size is fixed to (224 × 224). All the training runs are performed on a single Nvidia RTX 3090 graphic card.

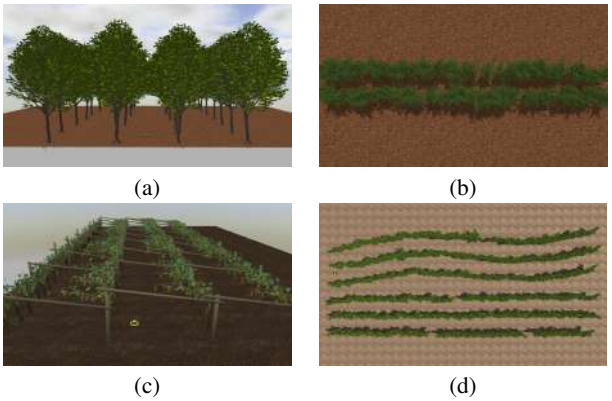


Fig. 4. Gazebo simulated environments used to test the SegMin approach in relevant different crops rows: wide rows composed of high trees (a), a narrow pear tree row (b), a pergola vineyard with asymmetric rows (c), straight and curved vineyard rows (d). In the last case, the tests were carried out in the second row from above and the second row from below.

C. Navigation Results

The overall navigation pipeline of SegMin and its variant SegMinD are tested in realistic crops fields in simulation using relevant metrics for visual-based control without precise localization of the robot, as done in previous works [20], [11]. The camera frames are published at a frequency of 30 Hz, while the inference is carried out at 20 Hz, and the controllers publish the velocity commands at 5 Hz. The evaluation has been performed using the testing package of the open-source PIC4rl-gym[3] in Gazebo [25]. The selected metrics aim at evaluating the effectiveness of the navigation (clearance time) as well as the precision, quantitatively comparing the obtained trajectories with a ground truth one through Mean Absolute Error (MAE) and Mean Squared Error (MSE). The ground truth trajectories have been computed by averaging the curve obtained by interpolating the plants' poses in the rows. For the asymmetric pergola vineyard case, the row is intended as the portion of the pergola without vegetation on top, as shown in Figure 4 (c). The response of the algorithms to terrain irregularity and rows geometry is also studied, including in the test significant kinematic information of the robot. The cumulative heading average $\gamma[rad]$ along the path is considered, together with the mean linear velocity $v_{avg}[m/s]$ and the standard deviation of the angular velocity $\omega_{stddev}[rad/s]$ commands predicted to keep

[3] https://github.com/PIC4SeR/PIC4rl_gym

TABLE II

NAVIGATION RESULTS OBTAINED IN DIFFERENT TEST FIELDS WITH THE SEGMIN, SEGMIND, AND PREVIOUS WORK SEGZEROS SEGMENTATION-BASED ALGORITHMS. THE METRICS TEST THE EFFECTIVENESS OF THE NAVIGATION (CLEARANCE TIME) AND ITS PRECISION WITH MEAN ABSOLUTE ERROR (MAE) AND MEAN SQUARED ERROR (MSE) BETWEEN OBTAINED AND GROUND TRUTH PATH. THE CUMULATIVE HEADING AVERAGE $\gamma[rad]$, THE MEAN LINEAR VELOCITY $v_{avg}[m/s]$, AND THE STANDARD DEVIATION OF THE ANGULAR VELOCITY $\omega_{stddev}[rad/s]$ COMMANDS PROVIDE RELEVANT KINEMATIC INFORMATION OF THE ROBOT WHILE NAVIGATING. SEGZEROS IS NOT APPLICABLE IN THE CASE OF HIGH TREES, PEAR TREES, AND PERGOLA VINEYARDS SINCE THE SKY MAY BE COVERED BY VEGETATION.

Test Field	Method	Clearance time [s]	MAE [m]	MSE [m]	Cum. γ_{avg} [rad]	$v_{avg}[m/s]$	$\omega_{stddev}[rad/s]$
High Trees	SegMin	**40.409 ± 0.117**	0.265 ± 0.005	0.084 ± 0.003	0.079 ± 0.001	0.487 ± 0.000	0.054 ± 0.002
	SegMinD	40.440 ± 0.515	**0.174 ± 0.006**	**0.036 ± 0.002**	0.048 ± 0.002	0.484 ± 0.006	0.063 ± 0.019
Pear Trees	SegMin	**42.058 ± 1.228**	0.034 ± 0.012	**0.002 ± 0.001**	0.013 ± 0.002	0.483 ± 0.003	0.108 ± 0.054
	SegMinD	42.259 ± 1.912	**0.031 ± 0.017**	**0.002 ± 0.002**	0.016 ± 0.004	0.477 ± 0.009	0.026 ± 0.004
Pergola Vineyard	SegMin	**40.859 ± 0.386**	**0.077 ± 0.011**	**0.011 ± 0.003**	0.030 ± 0.022	0.479 ± 0.003	0.174 ± 0.021
	SegMinD	41.135 ± 0.329	0.097 ± 0.052	0.015 ± 0.014	0.029 ± 0.011	0.475 ± 0.004	0.204 ± 0.032
Straight Vineyard	SegMin	**50.509 ± 0.305**	**0.105 ± 0.003**	**0.014 ± 0.001**	0.033 ± 0.002	0.487 ± 0.000	0.079 ± 0.011
	SegMinD	50.629 ± 0.282	0.110 ± 0.005	0.018 ± 0.003	0.026 ± 0.009	0.486 ± 0.001	0.088 ± 0.005
	SegZeros	53.695 ± 1.029	0.138 ± 0.025	0.024 ± 0.010	0.027 ± 0.004	0.457 ± 0.008	0.089 ± 0.008
Curved Vineyard	SegMin	53.321 ± 0.249	0.115 ± 0.008	0.017 ± 0.002	0.036 ± 0.008	0.487 ± 0.001	0.088 ± 0.021
	SegMinD	**51.444 ± 1.030**	**0.093 ± 0.005**	**0.012 ± 0.001**	0.015 ± 0.004	0.484 ± 0.007	0.065 ± 0.008
	SegZeros	71.048 ± 27.132	0.108 ± 0.044	0.019 ± 0.009	0.045 ± 0.008	0.395 ± 0.127	0.114 ± 0.039

the robot correctly oriented. The mean value of ω is always close to zero due to the consecutive correction of the robot orientation.

The complete results collection is reported in Table II. For each metric, an average value and the standard deviation are indicated since all the experiments have been repeated over 3 runs on a 20 m long track in each crop row. The proposed method demonstrates to solve the problem of guiding the robot through tree rows with thick canopies (high trees and pears) without a localization system, as well as in peculiar scenarios such as the pergola vineyards. The identification of plant branches and wooden supports hinders the usage of previously existing segmentation-based solutions that were based on the assumption of finding a free passage solely considering the zeros of the binary segmentation mask [20]. We refer to this previous method as SegZeros in the comparison of the results that we tested using the same segmentation neural network.

The SegMin approach based on histogram minimum search demonstrates to be a robust solution to guide the robot through tree rows. The introduction of the depth inverse values as a weighting function allows SegMinD to further increase the precision of the algorithm in following the central trajectory of the row in complex cases such as wide rows (high trees) and curved rows (curved vineyard). The different sum histograms obtained with SegMin and SegMinD are directly compared in Figure 5, showing the sharper trend and the global minimum isolation obtained, including the depth values. Moreover, the novel methods show competitive performance also with standard crop rows where a free passage to the end of the row can be seen in the mask without the disturbance of canopies. The histogram minimum approach significantly reduces the navigation time and the trajectory precision in vineyard rows (straight and curved) compared to the previous segmentation-based baseline method. The

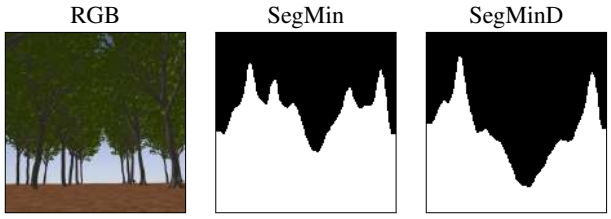


Fig. 5. Comparison of the two histograms obtained using the two different algorithms, given the RGB frame on the right. It can be noticed how SegMinD offers a narrower and less ambiguous global minimum point.

search of plant-free zero clusters in the map results in being less robust and efficient, leading the robot to undesired stops during the navigation and to an overall slower and more oscillating behavior. Moreover, the standard deviation of the angular velocity is coherent with the obtained results, being smaller in the cases when the trajectory is more accurate, and the cumulative heading shows larger values when the algorithms are more reactive.

Nonetheless, the trajectories obtained with the SegMin, SegMinD and SegZeros algorithms are also visually shown in Figure 6 inside representative scenarios: a cluttered, narrow row with small pear trees, a wide row with high trees, and curved vineyards with state-of-the-art method SegZeros.

IV. CONCLUSIONS

In this work, we presented a novel method to guide to a service-autonomous platform through crop rows where a precise localization signal is often occluded by the vegetation. Trees rows represented an open problem in row crop navigation since previous works based on image segmentation or processing failed due to the presence of branches and canopies covering the free passage for the rover in the image. The proposed pipeline SegMin and SegMinD overcome this limitation by introducing a global minimum search on the

A Map-Free LiDAR-Based System for Autonomous Navigation in Vineyards

Riccardo Bertoglio[1], Veronica Carini[1], Stefano Arrigoni[2], and Matteo Matteucci[1]

***Abstract*— Agricultural robots have the potential to increase production yields and reduce costs by performing repetitive and time-consuming tasks. However, for robots to be effective, they must be able to navigate autonomously in fields or orchards without human intervention. In this paper, we introduce a navigation system that utilizes LiDAR and wheel encoder sensors for in-row, turn, and end-row navigation in row structured agricultural environments, such as vineyards. Our approach exploits the simple and precise geometrical structure of plants organized in parallel rows. We tested our system in both simulated and real environments, and the results demonstrate the effectiveness of our approach in achieving accurate and robust navigation. Our navigation system achieves mean displacement errors from the center line of $0.049\,\mathrm{m}$ and $0.372\,\mathrm{m}$ for in-row navigation in the simulated and real environments, respectively. In addition, we developed an end-row points detection that allows end-row navigation in vineyards, a task often ignored by most works.**

I. Introduction

The increasing demand for food in the current climate-changing environment introduces new challenges, such as the necessity of increasing production and the sustainability of crop management while reducing costs [1]. Agricultural robots can help achieve these goals by performing repetitive and time-consuming tasks, allowing farmers to improve production yields. At the same time, for robots to be effective, they must be able to navigate autonomously in fields or orchards without human intervention. Navigation approaches can be broadly divided into two categories: those with or without a map of the environment. While map-based approaches can be helpful in unstructured environments, they require a more expensive sensor suite and incur increased computational effort. Additionally, localization on a pre-built map can fail due to the constantly changing nature of agricultural environments. Nevertheless, agricultural environments typically have a simple and precise geometrical structure, with crops organized in parallel rows. This structure can be exploited for navigation without the need for a map.

Autonomous navigation in agriculture often utilizes GNSS information for pre-planned routes or as supplementary information. Additionally, Differential GNSS technology provides higher localization accuracy of up to centimeters. However, the GNSS signal is not always available, especially for those cultivations with high plants and abundant vegetation. LiDAR and camera sensors are also utilized for navigation. LiDARs can be either 2D or 3D sensors, with the latter characterized by multiple scanning planes. LiDAR sensors provide a geometrical view of the environment, work at a reasonable frequency (over 10 Hz), and are precise. Cameras, such as RGB, stereo, or RGB-D, provide a more complex semantic interpretation of the environment, which is helpful for tasks like obstacle avoidance. Stereo and RGB-D cameras can also produce 3D renderings of the environment. Although LiDARs only provide geometrical data, they are less susceptible to lighting conditions than cameras, which is essential in agricultural environments where strong sunlight and shadows are typical.

Fig. 1. Our robotic platform navigating a real vineyard.

The VINBOT project [2] has developed a vineyard navigation system combining a line detection algorithm and GNSS navigation for in-row navigation. Two lines representing vineyard rows were identified using a 2D laser and RANSAC algorithm. The robot changed the corridor by rotating around one of two points representing the plant's end. Localization relied on IMU, GPS, and wheel odometry data, but tests have shown that plant holes should be manually managed to avoid misinterpretation.

The VineSLAM algorithm, described in [3], employed laser rangefinder data and known parameters to identify trunks and masts as landmarks for 2D SLAM. RFID tags were utilized to mark the corridor boundaries for topological mapping. However, the algorithm's accuracy relied on the detection of trunks and masts, and external factors such as grass and wind introduced substantial noise, compromising navigation reliability.

[1]Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milan, Italy {name.surname}@polimi.it

[2]Department of Mechanical Engineering, Politecnico di Milano, Milan, Italy stefano.arrigoni@polimi.it

979-8-3503-0704-7/23/$31.00

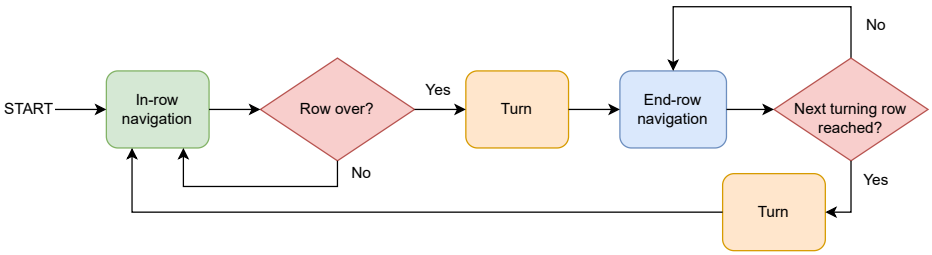


Fig. 2. The general navigation software architecture.

Bernad et al. [4] proposed three straightforward in-row navigation approaches using only 2D LiDAR data. The most effective algorithm involved calculating the average distance from both sides of the crop row and estimating an orientation correction based on the offset. They achieved an accuracy of $0.041\,\mathrm{m}\pm0.034\,\mathrm{m}$ from the center line when testing outdoors with potted maize plants.

Rovira-Más et al. [5] presented a multi-sensor navigation approach for inside-row guidance. The authors used a so-called Augmented Perception Obstacle Map (APOM) to store and evaluate readings from a 3D stereo camera, LiDAR, and ultrasonic sensors. The map is then analyzed to find specific situations representing the status of row detection. The next navigation target point is only computed if one or both rows are found.

Mengoli et al. [6], [7] proposed Hough Transform-based methods for orchard navigation, including in-row and row-change maneuvers. The authors enhanced robustness by incorporating vineyard geometry conditions and using GPS to identify corridor ends. The detected pivot point in row-change maneuvers had an RMSE of 0.3429 m in the x direction and 0.5840 m in the y direction.

Aghi et al. [8] introduced a vineyard in-row navigation algorithm with two components. The first component uses an RGB-D camera's depth map to detect the end of the row by fitting a rectangular area to the farthest pixels. In case of failure, a backup algorithm takes over, utilizing a neural network to identify and correct the robot's orientation if needed.

The Field Robot Event (FRE)[1] is a robotics competition that focuses on autonomous navigation in agricultural environments. We drew inspiration from the in-row navigation approach used by the Kamaro team [9] in the 2021 FRE competition for maize fields and adapted it for vineyard navigation. Our navigation system utilizes a single LiDAR and wheel encoders to reduce sensor requirements and costs. Additionally, we developed an end-row navigation algorithm to facilitate autonomous row changes. We proposed a straightforward evaluation benchmark for in-row navigation and end-row point detection, eliminating the need for external devices like laser tracking or Differential GNSS systems. The system was tested in both real vineyard (see Figure 1) and simulated environments. The complete algorithm code is available at this GitHub repository: `https://github.com/AIRLab-POLIMI/MFLB-vineyard-navigation`.

[1] `https://fieldrobot.nl/event`

II. Materials and Methods

We developed our navigation algorithm for a skid-steering mobile robot, although the general structure can also be adapted to other types of kinematics. The navigation software was implemented using the Robot Operating System (ROS) library, specifically the Melodic version on Ubuntu 18.04 LTS. The software architecture is presented in Figure 2.

Initially, the robot is assumed to reach the beginning of a row; the In-row navigation module guides the robot to follow the row until the end is detected. Then, the robot performs an open-loop turn managed by the End-Row navigation module, which guides the robot along the border of the vineyard until it reaches a specified row to turn into, where the In-row navigation module is reactivated. The following gives a more detailed description of each algorithm component.

A. Input Data

Our algorithm needs very few input data, namely, an odometry source and 2D laser scans. Since we used a robot with a skid-steering kinematic, we computed its odometry with the model presented in [10]. The kinematic relation is expressed as follows:

$$\begin{pmatrix} v_x \\ v_y \\ \omega_z \end{pmatrix} = A \cdot \begin{pmatrix} V_l \\ V_r \end{pmatrix} \tag{1}$$

where $v = (v_x, v_y)$ is the vehicle's translational velocity with respect to its local frame, ω_z is its angular velocity, V_l and V_r are the left and right linear tread velocities, and matrix A is defined by Equation (2). Following the experiments presented in [10] we have calibrated the matrix A that, in the case of an ideal symmetrical kinematic, takes the following form:

$$A = \frac{\alpha}{2x_{ICR}} \cdot \begin{bmatrix} 0 & 0 \\ x_{ICR} & x_{ICR} \\ -1 & 1 \end{bmatrix} \tag{2}$$

where, x_{ICR} is the $x-axis$ component of the Instantaneous Center of Rotation (ICR), and α is a correction factor to account for mechanical issues such as tire inflation conditions or the transmission belt tension. Both these parameters have been empirically estimated following the directions provided in [10].

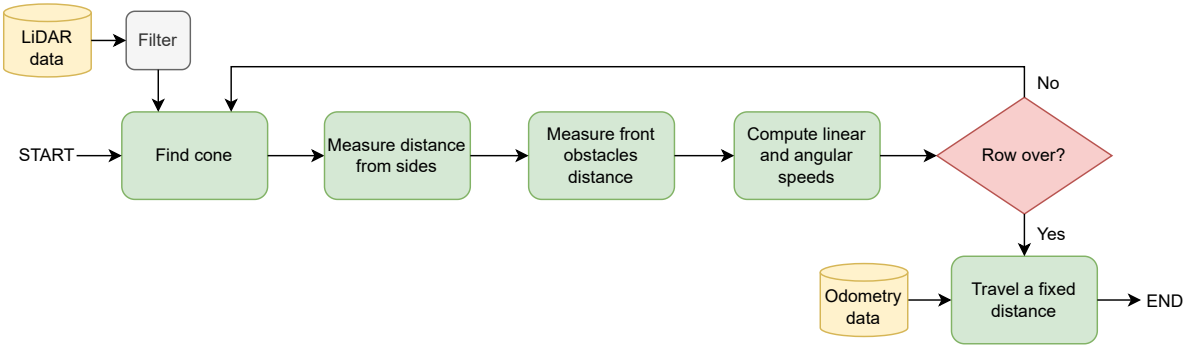


Fig. 3. In-row navigation algorithm.

Beyond odometry, our navigation system expects 2D laser scans to perceive the environment. We transformed LiDAR messages from an Ouster OS1 3D LiDAR sensor into 2D laser scans through the *pointcloud_to_laserscan* ROS package[2]. We set the sensor at 10 Hz and 1024 points for each of its 64 planes. We then filtered the laser scan messages to reduce their size. We first applied radius filtering to remove points outside a circle centered on the sensor and then downsampling to reduce the density of points. We also applied outlier filtering to remove noise from data.

B. In-row navigation

In the in-row navigation stage, the navigation system makes the robot traverse a corridor created by two lines of plants by maintaining an equal distance from them as much as possible. The approach we used for the in-row navigation has been adapted from that of the Kamaro team[3] which participated in the 2021 FRE competition.

The functioning of the In-row navigation module is graphically illustrated in Figure 3. The *find_cone* method analyzes the laser scan messages to find an obstacle-free cone in front of the robot. To do so, a cone centered on the moving robot direction is gradually grown by enlarging the apex angle until a certain number of points fall inside the cone. The two cone sides are moved independently, and they have a configurable length. Once the cone is found, we compute an angular offset between the cone center line and the robot center line. This angular offset is increased by an additional offset proportional to the distance between the robot and corridor center. The latter distance is computed by growing two rectangles on the side of the robot until a certain number of points fall into them. A graphical representation of the cone and rectangles is shown in Figure 4.

The final angular offset defines a new line pointing toward the steering direction. We use a PID controller to steer toward the point on this line that is $1\,\mathrm{m}$ in front of the robot. The linear speed is set to a constant value, and it is reduced if an object in front of the robot is detected. The algorithm uses

[2]https://github.com/ros-perception/pointcloud_to_laserscan

[3]https://github.com/Kamaro-Engineering/fre21_row_crawl

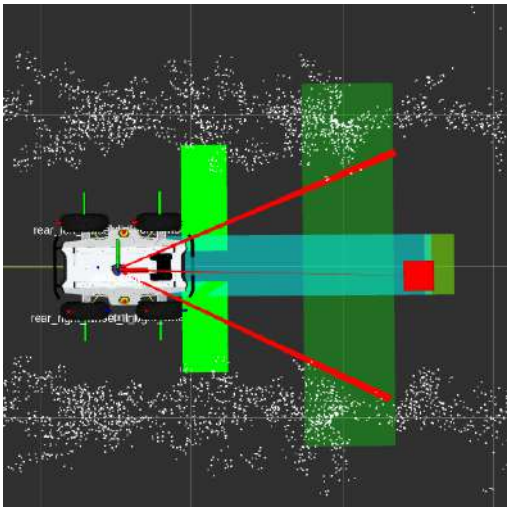


Fig. 4. The robot navigating inside a row in the simulated environment. The two thick red lines represent the sides of the cone, while the red square on the center line represents the new navigation point to follow. The light green rectangles are used to compute the distances from both sides. The semi-transparent rectangle in front of the robot is used to check if the end of the row is reached by counting the number of points inside it. The rectangle placed in the middle-front part of the robot is used to check an obstacle's distance and reduce speed accordingly.

a rectangle in front of the robot to calculate the target speed based on the distance between the robot and any obstacles.

At each linear and angular speed update, the In-row module checks if the end of the row has been reached. This procedure involves a rectangular area (colored light green in Figure 4) placed in front of the robot, spanning the entire corridor and part of both row sides. The corridor is over when the number of points in the rectangle approaches zero. The last step is to exit the row by a fixed distance measured through the robot odometry. Since the latter distance is usually of about $1\,\mathrm{m}$, the odometry guarantees a reasonable accuracy.

Once the robot has exited a row, it performs an in-place rotation by a fixed angle (usually $90°$). The user needs to set the direction of the first rotation, left or right. During the rotation, the odometry is monitored to halt the robot when the required angle has been performed. Note here that we expect the robot to skid, and because of this, the effective rotation

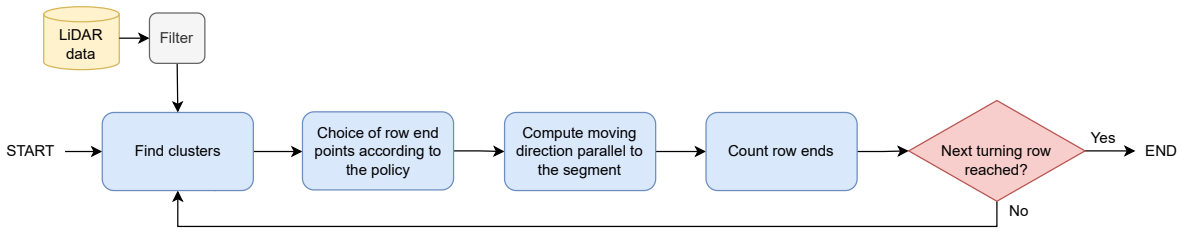


Fig. 5. End-row navigation algorithm.

might differ from 90°. However, the algorithm overcomes this problem by selecting two end points—one positioned in front and the other at the back of the robot. Subsequently, it rotates the robot to align its moving direction parallel to the line segment connecting these two points. It's also important to note that the robot does not need to be perfectly aligned with the row direction when it begins navigating at the beginning of the row. In both scenarios, the algorithm compensates for an incomplete rotation up to a specific angle. The maximum angle that can be recovered depends on factors such as the width of the row, the robot's distance from the row's starting point, and algorithm parameters like the length of the cone sides. Once the turn is completed, the navigation system activates the End-row navigation module.

C. End-row navigation

After completing the turn, the navigation system initiates the End-row algorithm. A schematic representation of the End-row navigation algorithm is presented in Figure 5. The primary objective of this algorithm is to enable the robot to travel perpendicularly to the field rows until it reaches the next corridor. The algorithm is specifically designed to leverage row ends, which typically consist of wooden support poles in vineyards. We employed the Euclidean Cluster Extraction technique [11] to identify row ends from the 2D point cloud data. This simple algorithm is highly effective in vineyards because the rows are widely separated by open areas to allow for human operations. Each obtained cluster represents a row end.

The subsequent task selects a point for each recognized cluster, representing the row end. We evaluated two policies to select such end point. The first policy, termed *Nearest*, involves selecting the nearest cluster point to the robot center, which is surrounded by a minimum number of points at a threshold distance. Therefore, the circular neighborhood's radius and the minimum number of points are parameters that need to be configured. The second policy, called *Line fitting*, involves a first step in which the end point is selected with the Nearest policy, then a line is fitted to the cluster of points, and finally, the end point is projected onto that line. We implemented line fitting using the random sample consensus (RANSAC) algorithm, finding that 100 iterations and a distance threshold of 0.1 m offer a good balance between speed and accuracy.

After detecting the points representing row ends, we use them to construct segments that indicate the navigation direction. Indeed, the navigation system keeps a fixed distance from row ends by maintaining a moving direction parallel to such fitted segments. Figure 6 displays the clustered row ends in various colors and the identified end points through the Nearest policy with red squares. Additionally, the current direction segment is shown with a red line. Figure 7 shows the clusters and end points obtained through the Line fitting policy.

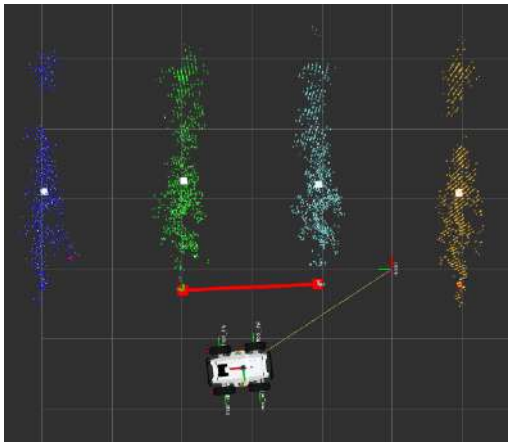


Fig. 6. A screenshot of the simulation environment with the clustered row ends. Each cluster is represented with a different color. With red squares are shown the selected end points according to the Nearest policy. The red line represents the segments the robot follows to navigate perpendicularly to row ends.

While the robot navigates parallel to end rows, it keeps track of the number of passed row ends and stops in the middle of the next corridor to enter. Then it will perform a 90° in-place rotation, and the system will activate the In-row navigation module again.

III. Results

We conducted experimental tests in both simulated and real environments. The simulation has been performed on the Gazebo simulator with vineyard models at different vegetative stages taken from the BACCHUS project repository[4] (see Figure 8). We also performed tests in a real vineyard located on the Piacenza (Italy) campus of the Università Cattolica del Sacro Cuore. The simulated environment consisted of three vineyard corridors approximately 36 m long and approximately 2 m large, characterized by three different vegetative stages: low, medium, and high. The results reported for

[4] https://github.com/LCAS/bacchus_lcas

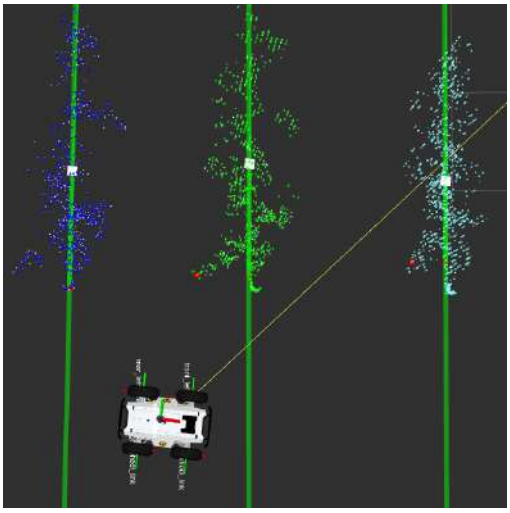


Fig. 7. A screenshot of the simulation environment when the robot is performing end-row navigation. End row points are clustered, and a line is fitted for each cluster (green lines). Then, each end point (red squares) is projected onto the line model of its cluster.

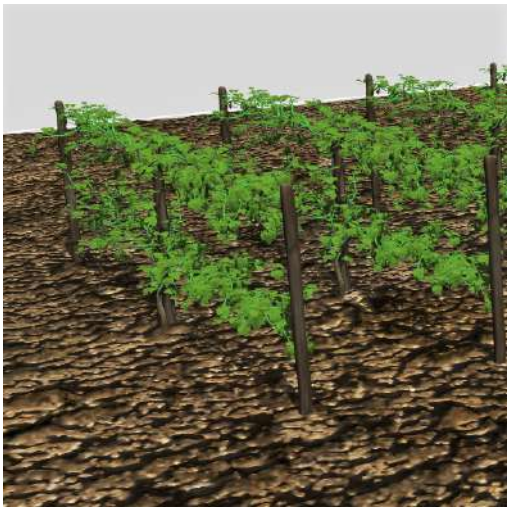


Fig. 8. A screenshot that depicts a portion of the simulated vineyard.

the simulated environment are thus an average over the three vegetative stages. The real environment was a single vineyard corridor with a length of approximately $40\,\mathrm{m}$ and a width of approximately $2.5\,\mathrm{m}$, which is one of the typical settings in Italy. The vegetative stage of the real vineyard was comparable to the high vegetative stage of the simulated one. During the tests, we reached a maximum linear speed of $2\,\mathrm{m\,s^{-1}}$ in the simulated environment and $1\,\mathrm{m\,s^{-1}}$ in the real environment for both in-row and end-row navigation. We mounted the Ouster OS1 LiDAR sensor at an approximate height of $1\,\mathrm{m}$ from the ground.

The navigation system ran on an onboard Shuttle XPC (model DS81L15) equipped with an Intel(R) Core(TM) i7-4790S CPU and 8 GB of RAM. The LiDAR sensors produced messages at a frequency of $10\,\mathrm{Hz}$, and the odometry was published at $50\,\mathrm{Hz}$. All the ROS nodes were capable of keeping up with the $10\,\mathrm{Hz}$ frequency of the LiDAR, except for the nodes responsible for clustering and end point detection, which proved to be the bottleneck of the system. Specifically, the node performing clustering with the Nearest end point picking policy operated at a minimum frequency of $9\,\mathrm{Hz}$, while the one using the Line fitting policy ran at a minimum frequency of $5\,\mathrm{Hz}$. Nevertheless, the bottleneck only affected the end-row navigation, which represents a small part of the total path traversed in a vineyard.

A. In-Row Navigation Evaluation

To evaluate the precision of the In-row navigation module, we measured the robot's displacement from the central row line. This displacement was determined by calculating the absolute distance between the robot's center and the central line of the row. In the simulated environment, we had access to the true robot position, whereas in the real-world test, we relied on the side distance measurements of the In-row algorithm performed via the LiDAR (which has a precision of $\pm 0.01\,\mathrm{m}$). Evaluating navigation accuracy in real agricultural environments is a challenging and ambiguous task currently addressed by agricultural robotics competitions such as that described in [12]. Alternatively, one could utilize an expensive yet highly accurate laser position tracking system, although determining the optimal target trajectory remains a nontrivial problem. In our case, we defined a perfectly row-centered trajectory as the optimal one. However, in both the simulation and the real vineyard, protruding vegetation and branches caused the robot to deviate from the central line, resulting in some average deviation from the center. Table I presents the outcomes of in-row navigation tests performed in simulation across three rows at varying vegetation stages and in two real vineyard rows.

Measurements	**Simulation**	**Real**
Mean center displacement	$0.049\,\mathrm{m}$	$0.372\,\mathrm{m}$
Max center displacement	$0.167\,\mathrm{m}$	$1.183\,\mathrm{m}$
Mean corridor width	$1.373\,\mathrm{m}$	$2.142\,\mathrm{m}$
Max corridor width	$2.300\,\mathrm{m}$	$2.620\,\mathrm{m}$
Min corridor width	$0.740\,\mathrm{m}$	$1.600\,\mathrm{m}$

TABLE I

IN-ROW NAVIGATION EVALUATION RESULTS.

The mean displacement from the central line was $0.049\,\mathrm{m}$ in the simulated environment, whereas in the real vineyard, we observed a mean displacement of $0.372\,\mathrm{m}$. In both scenarios, the robot successfully avoided protruding branches and never collided with the row sides. Table I also presents the row width measurements computed from LiDAR scans. The measurements indicate that protruding vegetation causes row width variations, impacting robot centering. In the real scenario, the minimum measurable row width of $1.6\,\mathrm{m}$ was reached, as our LiDAR has a minimum scanning distance of $0.8\,\mathrm{m}$.

B. Row Ends Detection Evaluation

To estimate the accuracy of the row ends detection, we computed the Euclidean distance between the true center

Pole distance	Simulation		Real	
error	Nearest	Line fitting	Nearest	Line fitting
Mean	0.205 m	0.155 m	0.23 m	0.26 m
Max	0.540 m	0.363 m	0.30 m	0.32 m
Min	0.038 m	0.013 m	0.15 m	0.20 m

TABLE II

ROW END POINTS DETECTION EVALUATION.

of row support poles and those detected by our row ends detection system. It is important to note that the assumption that the pole center is always the true row end point is not always valid, as vegetation can cover the pole and protrude outward. In the simulated environment, we computed the instantaneous Euclidean distance from the real pole center to the end point detected by our system during a full turn from one row to the next. We performed measurements for three different vegetative stages. In the real environment, obtaining multiple measurements of the real displacement of the pole center from the robot is laborious and time-consuming. Furthermore, without any absolute positioning system available, the only way to measure it was manually, which introduced measurement errors in the order of centimeters. Therefore, we statically positioned the robot in the middle of a row to detect the two side end points and compared them to manual measurements.

In both the simulated and real scenarios, we compared the two policies explained in section II-C: Nearest and Line fitting. Table II shows the mean, max, and min distances between the true center poles coordinates and those detected by our system. In the simulated scenario, the Line fitting policy was more accurate with a mean of $0.155\,\mathrm{m}$. The Nearest policy also showed an acceptable mean distance of $0.205\,\mathrm{m}$ while being less computationally intensive. In the real scenario, the accuracy of both policies was comparable since the difference in the order of centimeters could be attributable to the error of manual measurements. Nonetheless, our row ends detection system performed accurately in both scenarios.

IV. CONCLUSIONS

In this paper, we have presented a simple and efficient map-free LiDAR-based navigation system designed for vineyard applications. Our approach relies on the geometrical structure of the environment and does not require a prebuilt map or GNSS measurements. The navigation system is capable of in-row, turn, and end-row navigation and has been tested in both simulated and real vineyards. The results of our experiments indicate that the proposed navigation system achieves accurate and reliable navigation performance, even under challenging vineyard conditions with variations in row spacing and vegetative stages. The system can effectively detect protruding vegetation and adjust the trajectory accordingly, potentially reducing crop damage. The proposed navigation system is simple and cost-effective, relying only on odometry and LiDAR as sources of information, requiring low computational effort. Future work can explore testing with a 2D LiDAR to compare the navigation precision and extend the system's evaluation to other types of line-arranged crops. Additionally, the system could be integrated with a robust semantic obstacle detection algorithm to enhance the navigation system's safety.

ACKNOWLEDGMENT

We are grateful to our colleagues at Università Cattolica del Sacro Cuore (Piacenza, Italy), especially Professor Matteo Gatti, for allowing us to conduct experiments in their vineyard on the university campus. This study was conducted within the Agritech National Research Center, and received partial funding from the European Union NextGenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR), missione 4, componente 2, investimento 1.4, D.D. 1032 17/06/2022, CN00000022), the European Union's Digital Europe Programme under grant agreement N.101100622, and the European Union's H2020 grant N.101016577.

REFERENCES

[1] R. Bertoglio, C. Corbo, F. M. Renga, and M. Matteucci, "The digital agricultural revolution: a bibliometric analysis literature review," *IEEE Access*, vol. 9, pp. 134 762–134 782, 2021.

[2] R. Guzmán, J. Ariño, R. Navarro, C. Lopes, J. Graça, M. Reyes, A. Barriguinha, and R. Braga, "Autonomous hybrid gps/reactive navigation of an unmanned ground vehicle for precision viticulture-vinbot," *Intervitis Interfructa Hortitechnica-Technology for wine, juice and special crops*, 2016.

[3] F. N. Dos Santos, H. Sobreira, D. Campos, R. Morais, A. Paulo Moreira, and O. Contente, "Towards a reliable robot for steep slope vineyards monitoring," *Journal of Intelligent & Robotic Systems*, vol. 83, pp. 429–444, 2016.

[4] P. Bernad, P. Lepej, Č. Rozman, K. Pažek, and J. Rakun, "An evaluation of three different infield navigation algorithms," in *Agricultural Robots-Fundamentals and Applications*, J. Zhou and B. Zhang, Eds. IntechOpen, 2019, ch. 3.

[5] F. Rovira-Más, V. Saiz-Rubio, and A. Cuenca-Cuenca, "Augmented perception for agricultural robots navigation," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 11 712–11 727, 2020.

[6] D. Mengoli, R. Tazzari, and L. Marconi, "Autonomous robotic platform for precision orchard management: Architecture and software perspective," in *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. IEEE, 2020, pp. 303–308.

[7] D. Mengoli, A. Eusebi, S. Rossi, R. Tazzari, and L. Marconi, "Robust autonomous row-change maneuvers for agricultural robotic platform," in *2021 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. IEEE, 2021, pp. 390–395.

[8] D. Aghi, V. Mazzia, and M. Chiaberge, "Local motion planner for autonomous navigation in vineyards with a rgb-d camera-based algorithm and deep learning synergy," *Machines*, vol. 8, no. 2, p. 27, 2020.

[9] J. Bier, T. Friedel, J. Barthel, E. Bulovas, and L. Tuschla, "BETEIGEUZE - KAMARO," pp. 17–22, 2022. [Online]. Available: https://www.fieldrobot.com/event/wp-content/uploads/2022/02/Proceedings_FRE2021.pdf

[10] A. Mandow, J. L. Martinez, J. Morales, J. L. Blanco, A. Garcia-Cerezo, and J. Gonzalez, "Experimental kinematics for wheeled skid-steer mobile robots," in *2007 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2007, pp. 1222–1227.

[11] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[12] R. Bertoglio, G. Fontana, M. Matteucci, D. Facchinetti, M. Berducat, and D. Boffety, "On the design of the agri-food competition for robot evaluation (acre)," in *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2021, pp. 161–166.

Surgical fine-tuning for Grape Bunch Segmentation under Visual Domain Shifts

Agnese Chiatti[1], Riccardo Bertoglio[1], Nico Catalano[1], Matteo Gatti[2], and Matteo Matteucci[1]

***Abstract*— Mobile robots will play a crucial role in the transition towards sustainable agriculture. To autonomously and effectively monitor the state of plants, robots ought to be equipped with visual perception capabilities that are robust to the rapid changes that characterise agricultural settings. In this paper, we focus on the challenging task of segmenting grape bunches from images collected by mobile robots in vineyards. In this context, we present the first study that applies surgical fine-tuning to instance segmentation tasks. We show how selectively tuning only specific model layers can support the adaptation of pre-trained Deep Learning models to newly-collected grape images that introduce visual domain shifts, while also substantially reducing the number of tuned parameters.**

I. INTRODUCTION AND BACKGROUND

The climate change crisis has highlighted the importance of increasing the sustainability of food production, as prescribed in the European Commission's "Farm to Fork" strategy[1]. In this regard, digital technologies are playing a crucial role in reducing the amount of water and chemicals used in agriculture [1]. One of the key applications of digital technologies is the deployment of mobile robots, which can perform a range of tasks such as plant spraying [2], weeding [3], and harvesting [4]. To carry out these tasks effectively, robots need the ability to autonomously monitor plant traits and status, a task also known as *plant phenotyping*. For example, in vineyards, a robot must be capable of detecting plant organs for posing the appropriate cuts during winter pruning operations [5]. They also ought to accurately identify the presence of grape bunches, their level of ripeness, and promptly detect the emergence of any diseases that may compromise the fruit quality.

Robot's perception systems deployed in agricultural settings face particular challenges due to the significant weather and seasonal variations that characterise these environments. Thus, ensuring the effective reuse of visual patterns and features learned under specific environmental conditions (e.g., in terms of weather, lighting, and plant diversity) becomes crucial. This requirement stems from the need to guarantee accurate plant monitoring, even when the underlying conditions change. For instance, viewpoint changes caused by different sensor positions and occlusions caused by leaves are prominent factors that can hinder the accurate monitoring of fruit [6], [7].

[1] Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, Milan, Italy {name.surname}@polimi.it

[2] Department of Sustainable Crop Production, Università Cattolica del Sacro Cuore, Piacenza, Italy matteo.gatti@unicatt.it

[1] https://food.ec.europa.eu/horizontal-topics/farm-fork-strategy_en

The widespread application of Deep Learning (DL) methods has considerably accelerated the progress in various visual perception tasks, including plant phenotyping [8]. However, supervised DL methods typically require abundant training data and are susceptible to changes in the data distribution. Moreover, training all model parameters on new data is a costly process in terms of computational power and memory footprint, especially when working on edge devices and mobile platforms. To address these issues, one possible approach is to pre-train the model on a large-scale source domain and fine-tune the parameters on a few examples from the target domain. The aim of fine-tuning is to adapt the model to the target domain while retaining the information learned during pre-training, particularly in cases where the source and target distributions significantly overlap despite the shift. This process is commonly known as transfer learning. A traditional transfer learning practice known as linear probing involves fine-tuning only the last few layers of a Deep Neural Network (DNN) while reusing features from earlier layers. This approach was based on initial evidence suggesting that representations in earlier layers may be more transferable to new data and tasks than the specialised features learned in higher layers [9].

Recent research [10], [11] has explored effective alternatives to this consolidated fine-tuning practice. Indeed, Lee et al. [10] discovered that selectively tuning only the earlier, intermediate, or last layers of a DNN can counteract different types of distribution shifts and often even outperform cases where all model parameters are tuned. They have named this approach *surgical fine-tuning* (SFT). Their study concerned transfer learning across different image classification benchmarks, such as CIFAR and ImageNet. However, the authors' conclusions have yet to be validated on image segmentation tasks and data gathered in real-world application scenarios, e.g., from mobile robots.

This paper focuses on the task of grape bunch segmentation, which is a critical prerequisite for autonomous plant phenotyping and yield forecast in vineyards [12]–[14]. Our research investigates whether surgical fine-tuning can support grape bunch segmentation under visual domain shifts. To address this research question, we extend the study of surgical fine-tuning from image classification models to instance segmentation architectures in the specific case of viticulture. The work in [12] is most closely related to this study, because it evaluates the utility of linear probing for grape segmentation. However, the experiments in [12] did not examine the option of fine-tuning layers other than the classification head.

To facilitate the analysis of different types of visual domain shifts that characterise vineyards, we introduce the VINEyard Piacenza Image Collections (VINEPICs) [15], a comprehensive and novel grape image archive. In [14], Santos et al. presented the Embrapa Wine Grape Instance Segmentation Dataset (WGISD), which is a large-scale collection of vineyard images displaying high-resolution instances of grape bunches across five different grapevine varieties. Our dataset was gathered in a distinct geographic area and it encompasses different grapevine varieties from those in the WGISD dataset, including wine and table grapes. Crucially, the proposed VINEPICs dataset contains additional variations in terms of camera viewpoint, scene occlusion, and time of data collection. Moreover, we captured images using a consumer-grade camera mounted on a mobile robot, which presents additional challenges due to possible motion blur from the robot's movement. As such, the contributed dataset more closely resembles realistic setups in autonomous vineyard phenotyping compared to the WGISD benchmark.

Our results from applying the widely-adopted Mask R-CNN model [16]–[18] to challenging robot-collected images indicate that adopting a surgical fine-tuning strategy can significantly outperform both linear probing and full parameter tuning when novel samples that introduce distribution shifts are considered. The paper is structured as follows. In Section II, we present the reference datasets, ablation study, technical implementation, and evaluation metrics used in our experiments. We then discuss the experimental results in Section III. Concluding remarks and future extensions of this work are left to Section IV.

TABLE I: Domain shifts from source to target data.

Dataset	Changes introduced	Shift types[10]	Instances
Source: WGISD	-	-	2,020
Fine-tuning set: VINEPICs21	geographic area, vineyard, Red Globe camera setup	natural, feature-level input-level	668
Target sets: VINEPICs22R	temporal: different years	input-level	100
VINEPICs22RV	temporal, camera viewpoint	input-level	112
VINEPICs22RF	temporal, foliage occlusion	input-level	105
VINEPICs22C	temporal, grape variety (Cabernet S.: red)	input-level feature-level	138
VINEPICs22O	temporal, grape variety (Ortrugo: white)	input-level feature-level	135

II. Materials and methods

To test the performance of applying surgical fine-tuning to instance segmentation models, we ran a set of layered experiments. Consistently with [10], we set up the training in two stages. First, we pre-trained on the largest available set of examples for the grape segmentation task: namely WGISD in this case [19]. Then, we considered different target sets that introduce a distribution shift from the source set. The goal was evaluating the extent to which transfer learning can be achieved from source to target, with minimal adjustments, thanks to surgical fine-tuning. Differently from [10], where the evaluation set was held out from the same data used for fine-tuning, we ran inferences on a different dataset, collected one year after the fine-tuning set. This setup resembles the real-world challenges of viticulture applications. Indeed, grape images can be collected only at specific times of the year and adapting learning models from past years to newly-collected data becomes essential.

A. Datasets

Embrapa WGISD. The Embrapa Wine Grape Instance Segmentation Dataset (WGISD) [19] comprises 300 high-resolution images depicting 2,020 grape bunches from five *Vitis vinifera* L. grapevine varieties: Chardonnay, Cabernet Franc, Cabernet Sauvignon, Sauvignon Blanc, and Syrah. The images were captured at the Guaspari Winery (Espírito Santo do Pinhal, São Paulo, Brazil) in April 2018, with the exception of images of the Syrah dataset that was collected in April 2017. Grape bunches were photographed while keeping the camera principal axis approximately perpendicular to the vineyard row, using both a Canon EOS REBEL T3i DSLR camera and a Motorola Z2 Play smartphone and were resized and stored at a resolution of 2048x1365. At the time of data collection, no defoliation treatments were applied except for the routine canopy management for wine production adopted in the region. In the original data split used in [14], 110 images (accounting for 1612 grape instances) were jointly devoted to training and validation, whereas 27 images (i.e., 408 grape instances) were held out for testing. However, the actual split between training and validation was not provided. Therefore, we decided to use a 20% validation split stratified across grape varieties from the original training subset.

VINEPICs. The VINEyard Piacenza Image Collections (VINEPICs) dataset consists of grape images collected at the vineyard facility of Università Cattolica del Sacro Cuore in Piacenza, Italy. The VINEPICs dataset is publicly available under CC BY 4.0 (Attribution 4.0 International) license and accessible at this link `https://doi.org/10.5281/zenodo.7866442`. The acronym **VINEPICs21** refers to the first collection of images gathered in the summer of 2021 on Red Globe vines (*Vitis vinifera* L.) grafted on Selection Oppenheim 4 (SO4), i.e., the vine rootstock, growing outdoors in 25 L pots. This set includes 73 RGB images captured on three different dates: 26 images of resolution 480x848 were collected at beginning of grape ripening on

July 27th, 23 images of resolution 720x1280 on August 23rd when berries were fully coloured, and 24 images of resolution 720x1080 at harvest on September 9th. An Intel D435i RGB-D camera was used to capture the data, which was mounted on a SCOUT 2.0 AgileX robotic platform, a four-wheeled differential steering mobile robot[2]. The plants were arranged along two, vertically shoot-positioned, North-South oriented rows and hedgerow-trained for a canopy wall extending about 1.3 m above the main wire. Each vine had a $\sim$1 m cane bearing 10-11 nodes that was raised 80 cm from the ground. Between fruit-set (BBCH 71) and berry touch (BBCH 79) [20], the leaves around bunches were gradually removed for a resulting fully defoliated fruit zone with reduced incidence of berry sunburns [21]. Before veraison, eight vines were subjected to crop thinning to control for fruit occlusions caused by excessive fruit density. Accordingly, a basal bunch was kept every second shoot for about six retained bunches/vine; the remaining unthinned vines were clustered into two groups with about 10 and 4 bunches/vine. During data collection, the camera principal axis was rotated to form an angle of approximately 45° with the scanned plant row. The grape bunch regions were annotated using polygonal masks through the Computer Vision Annotation Tool (CVAT)[3], and the annotations followed the COCO annotation format[4].

A second and more extensive dataset, named **VINEPICs22**, was collected at the same vineyard facility of Università Cattolica del Sacro Cuore in Piacenza, Italy, on two separate dates in August and September 2022, approximately one year after the previous set. This dataset comprises 165 annotated images, representative of different types of domain shifts, including 1464 grape bunch instances. From this dataset, we extracted subsets of data to control for the incremental changes we expect from the fine-tuning domain (VINEPICs21) to the target domain, as detailed in Table I. Specifically, the VINEPICs22R set includes new images collected from the same grape variety (Red Globe), by maintaining the same camera viewpoint, and level of defoliation as VINEPICs21. VINEPICs22RV introduces a change in the camera viewpoint (i.e., the camera principal axis is perpendicular to the plant rows), while set VINEPICs22RF was captured first on non-defoliated canopies. Furthermore, sets VINEPICs22C and VINEPICs22O maintain the same camera viewpoint and defoliation level as VINEPICs21 but represent different grape varieties, namely Cabernet Sauvignon (red grape) and Ortrugo (white grape), growing in a experimental vineyard. Table I maps the changes introduced for each fine-tuning and target set to the taxonomy of shift types adopted in [10]. The selected target sets cover three shift types: i) *input-level shifts*, which occur due to variations in the visual appearance of the same environment (e.g., observing the same vineyard on different days introduces lighting variations); ii) *feature-level shifts*, where the source-target shift is caused by different populations of the same class, in our case, different grape varieties; and iii) *natural shifts*, which are due to collecting the source and target data in different environments, in our case, different growing conditions (potted vines vs. experimental vineyard). *Output-level shifts* do not concern our use-case, since the target class (grape bunches) remains unchanged throughout the experiments detailed in this paper.

[2]The analyses presented in this paper only concern RGB images, but we also collected depth data to support a wider range of applications, such as, e.g., estimating the volume of grape bunches.

[3]https://github.com/opencv/cvat

[4]https://cocodataset.org/

B. Surgical fine-tuning for instance segmentation

Given the focus on image classification tasks, the experiments described in [10] consider ResNet architectures [22] as a reference and utilize surgical fine-tuning to manipulate the different residual blocks. However, in the context of instance segmentation tasks, supplementary modules are introduced for detecting and segmenting object regions. Region-based segmentation architectures such as the widely utilized Mask R-CNN model [16] merge CNN feature extraction layers with a Region Proposal Network (RPN) that extracts Regions of Interest (ROI) from input images. Predicted object regions are then fed to three network heads that operate in parallel, generating predictions for the object class, bounding box, and polygonal mask (Figure 1). A popular implementation of this generalized architecture uses a combination of ResNets and Feature Pyramid Networks (FPN) as a backbone for the feature extraction step [17], [18].

To assess the efficacy of surgical fine-tuning in the context of region-based segmentation models, we also ought to examine the impact of selectively fine-tuning the FPN and RPN components, along with the residual blocks and classification heads. Hence, we conduct experiments that compare the following model ablations:

- **Tune All:** This configuration fine-tunes all model parameters.
- **Linear Probing:** In this classic configuration, only parameters in the three ROI heads are updated, while earlier layer parameters remain fixed at values learned during pre-training.
- **Res n:** This setup involves fine-tuning only the ResNet layers, specifically the residual block identified by the number n. We use the keyword "stem" to refer to the first residual block, and the notation "res n" for blocks numbered 2 and higher. This setup follows the rationale applied in [10].
- **Joint SFT: Res Block n + FPN at n:** This configuration is a variation of the previous setup, where the selected residual blocks are fine-tuned simultaneously with the related Feature Pyramid Network (FPN) operations.
- **RPN:** In this setup, we only apply surgical fine-tuning to the Region Proposal Network (RPN) in the Mask R-CNN model.

To the best of our knowledge, this is the first study on the application of surgical fine-tuning to instance segmentation tasks.

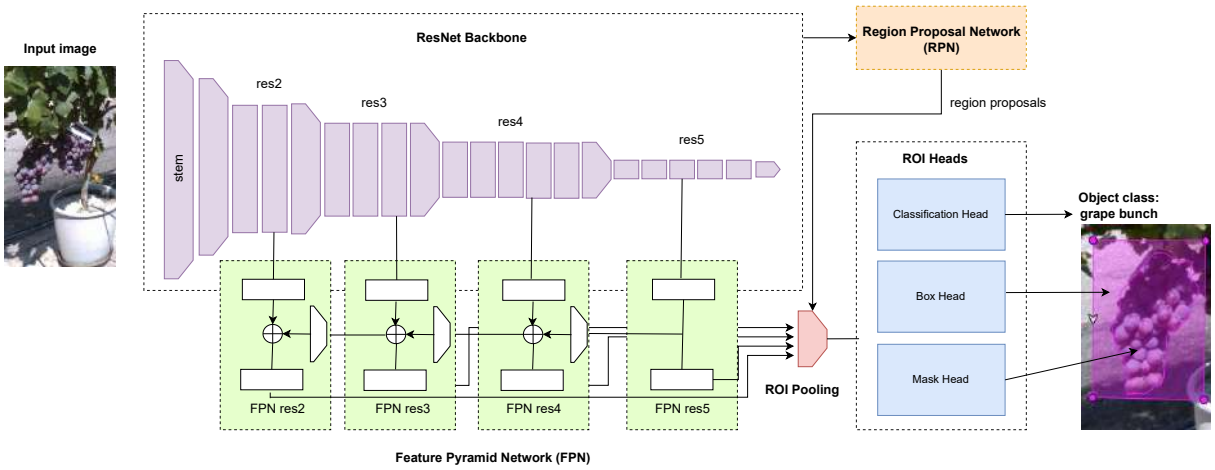


Fig. 1: Overview of the Mask R-CNN architecture. The backbone of the architecture is based on ResNet50, and features from blocks 2 to 5 are extracted and passed through a Feature Pyramid Network (FPN). The Region Proposal Network (RPN) generates region proposals, which are then combined with the upsampled features and input to three model heads, which predict object class, bounding box, and polygonal mask in parallel.

TABLE II: Inference results from pre-training baseline instance segmentation models on the WGISD dataset.

Baseline	$\mathbf{AP}_{0.3-0.9}$	$\mathbf{P}_{0.3-0.9}$	$\mathbf{R}_{0.3-0.9}$	$\mathbf{F1}_{0.3-0.9}$
Mask R-CNN ResNet101 (results from [14])	0.540	0.683	0.649	0.665
Mask R-CNN ResNet101 [18]	0.550	0.789	0.588	0.674
Mask R-CNN ResNet50 [18]	0.571	**0.806**	0.607	0.693
Mask R-CNN ResNet50 [17]	**0.623**	0.796	**0.663**	**0.724**

TABLE III: Number of fine-tuned parameters in the evaluated ablations.

Ablation	**Parameters**
tune all	∼ 45.3M
linear probing	∼ 17.8M
stem	∼ 9.5K
res2	∼ 215K
res2 + FPN	∼ 872K
res3	∼ 1.22M
res3 + FPN	∼ 1.94M
res4	∼ 7.1M
res4 + FPN	∼ 7.95M
res5	∼ 14.9M
res5 + FPN	∼ 16.1M
RPN	∼ 594K

C. Implementation details

To apply surgical fine-tuning as described in the previous section, we customised the Detectron2[5] implementation of the Mask R-CNN architecture. The code for reproducing these trials is available at `https://github.com/AIRLab-POLIMI/SFT_grape_segmentation`.

We augmented our training examples by applying various transformations such as Gaussian blur, additive Gaussian noise, random brightness, contrast, and saturation, pixel dropout, and random flipping transformations. During pre-training on the source domain, we utilized ResNet50 and ResNet101 backbones employing Group Normalization (GN). We experimented with different weight initializations following the Detectron2 Mask R-CNN baselines for the COCO instance segmentation task. In the first configuration, we used the weights obtained from the method introduced in [18], where the model was trained from scratch on COCO with an extended training schedule and an augmented jittering scale. In the second configuration, we initialized the model with the weights from the method presented in [17], where Mask R-CNN was trained on COCO instances from scratch, i.e., with random weight initialization, rather than reusing initialization values derived from ImageNet. All models were trained with a batch size of 2 images, and we used an early stopping criterion if the validation loss did not improve for 30 consecutive evaluation checks, with one evaluation check every 220 minibatch iterations. We optimized model parameters using stochastic gradient descent, with a constant learning rate set to 0.01.

[5]`https://github.com/facebookresearch/detectron2`

D. Evaluation metrics

We evaluate the instance segmentation performance by measuring the Average Precision (AP) of predicted object regions, as well as the standard Precision (P), Recall (R), and F1 score of predicted object instances. The metrics were averaged over Intersection over Union (IoU) values ranging from 0.3 to 0.9, to allow for comparison with the results presented in [14]. Consistently with [14], only predictions with confidence greater than 0.9 for the grape class are considered in the evaluation. We prioritize improvements in terms of F1 over individual P and R scores, as detecting

all true positives is as important as minimizing the false positives in the target use-case.

III. Results and Discussion

Before conducting the ablation study, we pre-trained three Mask R-CNN models on the WGISD dataset. Table II demonstrates that on our task, ResNet50 backbones generally delivered better results than ResNet101 backbones. Furthermore, initializing the model with weights obtained after training from scratch on the COCO dataset [17] yielded the best combination of segmented object region quality (in terms of AP) and grape class prediction quality (in terms of F1), compared to using weights from longer training schedules and large-scale jittering [18]. Therefore, we have chosen the "Mask R-CNN ResNet50 [17]" model as the baseline for fine-tuning on VINEPICs21.

During the fine-tuning stage, we applied the different ablations presented in Section II-B and evaluated the results on the five target sets selected from VINEPICs22. The top-performing methods in each set of trials, together with the "linear probing" and "tune all" alternatives, are summarised in Table IV. The complete evaluation results can be found in the extended version of this paper [23]. We also report the number of parameters tuned in each configuration in Table III.

Results on the **VINEPICs22R** sets approximate scenarios where the only change introduced is the date and time of data collection, while considering the same grape variety (Red Globe), camera viewpoint, and defoliation level as the fine-tuning set. In this case, fine-tuning the first four CNN layers individually, excluding the stem, ensured a higher AP than the scenario when all model parameters are tuned. In particular, tuning the third ResNet block led to the highest AP and F1 scores, outperforming linear probing.

Changing camera viewpoint, in **VINEPICs22RV**, led to generally higher scores than the previous set of trials. Notably, the AP scores are even higher than the AP achieved on the VINEPICs21 test set, for the majority of tested ablations. This result may be due to the fact that a perpendicular camera viewpoint is more similar to the setup adopted in the WGISD set, i.e., the source set. Moreover, it is worth noting that the VINEPICs21 test split comprises nearly twice as many grape instances as the VINEPICs22RV set. As a result, the average scores in the VINEPICs21 case provide more conservative performance figures than VINEPICs22, which accounts for approximately 100 instances for each subset (Table I). In this case, tuning the third and fourth ResNet blocks led to the most marked improvement over the the "tune all" and "linear probing" performance. In particular, tuning the fourth ResNet block in combination with its FPN layers led to the highest results with respect to the AP of region predictions, Recall and F1 of instance predictions. Interestingly, the top precision was achieved when tuning the Region Proposal Network in isolation, albeit generating a higher number of false positives, as indicated by the lower recall scores.

We then considered grape images captured in the presence of occluding foliage (**VINEPICs22RF**), under temporal and viewpoint conditions that are comparable to the tuning set. Similarly to the case of the temporal shifts introduced in VINEPICs22R, the top performance was achieved by tuning the third ResNet block. However, in this case, while the highest AP score was achieved in the "res3" configuration, the highest F1 was reached by jointly tuning res3 with FPN.

When we shift the target domain towards different grape varieties, the drop in performance from the fine-tuning set to the target sets is significant. Indeed, although the source set (WGISD) already included examples of both red and white grape bunches, the VINEPICs22C and VINEPICs22O sets are drastically more challenging than previously examined sets. First, the number of instances to be detected in each frame is significantly higher in this case, as exemplified in Figure 2. Moreover, images in these sets were captured at a lower resolution than WGISD and in lower lighting conditions than both the WGISD and the VINEPICs21 sets. Thus, this setup complicates not only the learning but also the manual annotation of grape instances. Under these challenging conditions, selectively tuning the stem and RPN was ineffective and prevented the model from providing any grape predictions [23]. Conversely, applying surgical fine-tuning to intermediate layers resulted in a significant improvement over the near-zero baseline performance. In the case of the Cabernet Sauvignon variety (**VINEPICs22C**) tuning only the parameters in the fourth ResNet block improved the AP by 10% and the F1 by 12%, compared to "linear probing". In the case of the Ortrugo variety (**VINEPICs22O**), jointly tuning res4 with FPN outperformed "linear probing" by 8%, in terms of AP, and by 14%, in terms of F1.

Overall, results from these experiments support the view that selecting intermediate network layers can outperform the common practice of only re-training the classification head of the model, when visual domain shifts are introduced. In particular, we found that selecting the third block for fine-tuning best supported temporal changes, as well as changes in the level of plant defoliation. Selecting the fourth ResNet block, instead, contributed to mitigating the impact of viewpoint and grape variety shifts. Importantly, adopting a surgical fine-tuning approach allowed us to substantially reduce the number of parameter updates, compared to the costly alternative of re-training the complete model from scratch: from over 45M total parameters to nearly 1M and 7M in the res3 and res4 cases (Table III).

IV. Conclusions

To effectively deploy mobile robots for agricultural applications, improving the adaptability of visual perception methods based on Deep Learning to rapidly-changing environments is essential. In particular, we have considered the task of autonomously segmenting grape instances from images collected in real vineyards. In this context, we showed that pre-training on large-scale, high-resolution training examples and fine-tuning only selected layers on more challenging robot-collected data can support knowledge transfer to newly-collected grape images that introduce changes in the camera viewpoint, foliage occlusion level, and grape variety.

(a) VINEPICs21 (b) VINEPICs22R (c) VINEPICs22RV (d) VINEPICs22RF (e) VINEPICs22C (f) VINEPICs22O

Fig. 2: Image examples from the VINEPICs sets. Examples from the WGIS set are available in [19].

TABLE IV: Inference results on test sets, after applying surgical fine-tuning on VINEPICs21.

Test set	Ablations	$AP_{0.3-0.9}$	$P_{0.3-0.9}$	$R_{0.3-0.9}$	$F1_{0.3-0.9}$
VINEPICs21 test	tune all	0.374	0.767	0.404	0.529
VINEPICs22R	tune all	0.254	0.682	0.273	0.390
	linear probing	0.226	**0.689**	0.234	0.350
	res3	**0.395**	0.602	**0.421**	**0.496**
VINEPICs22RV	tune all	0.387	0.634	0.436	0.517
	linear probing	0.409	0.660	0.454	0.538
	res4 + FPN	**0.463**	0.595	**0.515**	**0.552**
	RPN	0.305	**0.687**	0.325	0.442
VINEPICs22RF	tune all	0.342	0.696	0.371	0.484
	linear probing	0.290	**0.711**	0.305	0.426
	res3	**0.469**	0.577	**0.512**	0.542
	res3 + FPN	0.461	0.607	0.503	**0.550**
VINEPICs22C	tune all	0.007	0.571	0.004	0.008
	linear probing	0.003	0.286	0.002	0.004
	res2	0.013	**0.643**	0.009	0.018
	res4	**0.068**	0.534	**0.073**	**0.129**
	res4 + FPN	**0.068**	0.548	0.071	0.126
VINEPICs22O	tune all	0.022	**0.762**	0.017	0.033
	linear probing	0.021	0.449	0.023	0.044
	res4 + FPN	**0.102**	0.625	**0.111**	**0.189**

Notably, tuning intermediate network layers improves the robustness of the model to input-level and feature-level shifts. These findings complement the evidence gathered in [10] on image classification benchmarks, where input-level shifts were best supported by tuning the initial network layers. These results also withstand the popular practice of only tuning the last layers on a new target domain. Even in challenging scenarios where images of novel grape varieties are introduced at test time, surgical fine-tuning on intermediate network blocks allowed us to bootstrap the grape segmentation performance, while drastically reducing the number of parameters required for fine-tuning.

Our evaluation of the utility of surgical fine-tuning to support grape segmentation has been limited to methods derived from the widely-applied Mask R-CNN architecture. Thus, future research directions include the study of instance segmentation models that are based on Transformers, such as [24], for instance. Another transfer learning approach that we have not yet explored concerns the combination of linear probing with the selection of useful features from different layers, as proposed in [11]. The availability of the VINEPICs resource can facilitate the progress in tackling these unexplored research directions.

ACKNOWLEDGMENTS

This paper is supported by the Italian L'Oreal-UNESCO program "For Women in Science", the European Union's Digital Europe Programme under grant agreement Nº 101100622 (AgrifoodTEF). The study was conducted within the Agritech National Research Center and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4 – D.D. 1032 17/06/2022, CN00000022).

REFERENCES

[1] R. Bertoglio, C. Corbo, F. M. Renga, and M. Matteucci, "The digital agricultural revolution: A bibliometric analysis literature review," *IEEE Access*, vol. 9, pp. 134 762–134 782, 2021.

[2] H. Li, C. Guo, Z. Yang, J. Chai, Y. Shi, J. Liu, K. Zhang, D. Liu, and Y. Xu, "Design of field real-time target spraying system based on improved yolov5," *Frontiers in Plant Science*, vol. 13, p. 5192, 2022.

[3] R. Bertoglio, A. Mazzucchelli, N. Catalano, and M. Matteucci, "A comparative study of fourier transform and cyclegan as domain adaptation techniques for weed segmentation," *Smart Agricultural Technology*, p. 100188, 2023.

[4] G. Kootstra, X. Wang, P. M. Blok, J. Hemming, and E. Van Henten, "Selective harvesting robotics: current research, trends, and future directions," *Current Robotics Reports*, vol. 2, pp. 95–104, 2021.

[5] P. Guadagna, M. Fernandes, F. Chen, A. Santamaria, T. Teng, T. Frioni, D. Caldwell, S. Poni, C. Semini, and M. Gatti, "Using deep learning for pruning region detection and plant organ segmentation in dormant spur-pruned grapevines," *Precision Agriculture*, pp. 1–23, 2023.

[6] T. Zaenker, C. Lehnert, C. McCool, and M. Bennewitz, "Combining local and global viewpoint planning for fruit coverage," in *2021 European Conference on Mobile Robots (ECMR)*. IEEE, 2021, pp. 1–7.

[7] Y. Tang, J. Qiu, Y. Zhang, D. Wu, Y. Cao, K. Zhao, and L. Zhu, "Optimization strategies of fruit detection to overcome the challenge of unstructured background in field orchard environment: a review," *Precision Agriculture*, pp. 1–37, 2023.

[8] M. H. Saleem, J. Potgieter, and K. M. Arif, "Automation in agriculture by machine and deep learning techniques: A review of recent developments," *Precision Agriculture*, vol. 22, pp. 2053–2091, 2021.

[9] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *Advances in neural information processing systems*, vol. 27, 2014.

[10] Y. Lee, A. S. Chen, F. Tajwar, A. Kumar, H. Yao, P. Liang, and C. Finn, "Surgical fine-tuning improves adaptation to distribution shifts," in *Proceedings of the The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[11] U. Evci, V. Dumoulin, H. Larochelle, and M. C. Mozer, "Head2toe: Utilizing intermediate representations for better transfer learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 6009–6033.

[12] H. Cecotti, A. Rivera, M. Farhadloo, and M. A. Pedroza, "Grape detection with convolutional neural networks," *Expert Systems with Applications*, vol. 159, Nov. 2020.

[13] A. S. Aguiar, S. A. Magalhães, F. N. dos Santos, L. Castro, T. Pinho, J. Valente, R. Martins, and J. Boaventura-Cunha, "Grape Bunch Detection at Different Growth Stages Using Deep Learning Quantized Models," *Agronomy*, vol. 11, no. 9, Sept. 2021.

[14] T. T. Santos, L. L. de Souza, A. A. dos Santos, and S. Avila, "Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association," *Computers and Electronics in Agriculture*, vol. 170, p. 105247, 2020.

[15] R. Bertoglio, M. Gatti, S. Poni, and M. Matteucci, "VINEyard Piacenza Image Collections - VINEPICs," 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7866442

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2961–2969.

[17] K. He, R. Girshick, and P. Dollár, "Rethinking imagenet pre-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4918–4927.

[18] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2918–2928.

[19] T. T. Santos, L. L. de Souza, A. A. dos Santos, and S. Avila, "Embrapa Wine Grape Instance Segmentation Dataset – Embrapa WGISD," 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3361736

[20] D. Lorenz, K. Eichhorn, H. Bleiholder, R. Klose, U. Meier, and E. Weber, "Growth stages of the grapevine: Phenological growth stages of the grapevine (vitis vinifera l. ssp. vinifera)—codes and descriptions according to the extended bbch scale," *Australian Journal of Grape and Wine Research*, vol. 1, no. 2, pp. 100–103, 1995.

[21] M. Gatti, A. Garavani, A. Cantatore, M. G. Parisi, N. Bobeica, M. C. Merli, A. Vercesi, and S. Poni, "Interactions of summer pruning techniques and vine performance in the white vitis vinifera cv. o rtrugo," *Australian Journal of Grape and Wine Research*, vol. 21, no. 1, pp. 80–89, 2015.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[23] A. Chiatti, R. Bertoglio, N. Catalano, M. Gatti, and M. Matteucci, "Surgical fine-tuning for Grape Bunch Segmentation under Visual Domain Shifts," July 2023, arXiv: 2307.00837.

[24] Y. Fang, W. Wang, B. Xie, Q. Sun, L. Wu, X. Wang, T. Huang, X. Wang, and Y. Cao, "Eva: Exploring the limits of masked visual representation learning at scale," *arXiv preprint arXiv:2211.07636*, 2022.

Multi-camera GPS-free Nonlinear Model Predictive Control strategy to traverse orchards

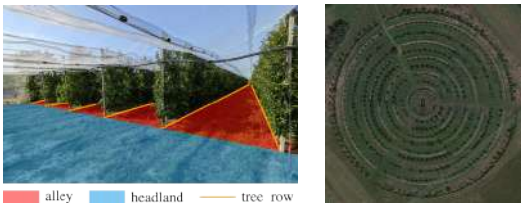
A. Villemazet[1,2], A. Durand-Petiteville[3] and V. Cadenat[1,2]

***Abstract*— This paper deals with autonomous navigation through orchards. It proposes a multi-camera GPS-free strategy relying on a Nonlinear Model Predictive Control (NMPC) scheme to follow a reference path. This latter, based on a Voronoi diagram for the row traversals or a spiral model for the headland maneuvers, is computed as a Non-Uniform Rational Spline (NURBS) curve making it possible to deal with multiple orchard layouts. The method has been implemented on our robot and validated through experimentation conducted in an orchard.**

I. Introduction

Robotics has been identified as one of the major solutions to promote truly sustainable agriculture where the necessary production increase matches environmental concerns [1]. In this work, we focus on orchard mechanization, and more specifically on the autonomous navigation system, which is mandatory to realize some agricultural tasks such as mowing, spraying, or harvesting. When moving through an orchard, a robot has to autonomously drive from the entrance of an alley to its exit, and then move to the next alley by navigating in the headlands, *i.e.,* the uncultivated area between the edge of the trees and the orchard boundary used for machinery maneuvers. It repeats these two steps to cover the whole area of interest (see Fig. 1(a)).

As the GPS signal is often blocked or perturbed by the dense canopy or nets protecting the trees [2], the existing navigation strategies rely on embedded sensors, either vision systems [3] [4] [5] [6] or LiDAR sensors [7] [8]. These works propose to compute and then follow a straight line passing through the middle of the alleys. The obtained line may be disturbed by the natural environment where branches and foliage are uneven and lighting conditions significantly vary. Moreover, these approaches do not allow coping with modern orchards whose circular layout is specifically designed to control pests thanks to ecological processes [9] (see Fig. 1(b)). Regarding maneuvers in the headland, the few existing works on this topic use dead reckoning because of the lack of sensory information in these zones. In such a case, the execution robustness and repeatability are significantly reduced [10]. We may nonetheless mention the following methods where dead reckoning is coupled with other techniques in an attempt to overcome this drawback: a slip compensation



(a) Straight orchard. (b) Gotheron circular orchard.

Fig. 1. Example of orchards.

solution [11], automatic detection of the rows extremities using either a laser [12] or dedicated artificial landmarks [13]. It then would be interesting to provide a navigation strategy able to cope with the different types of orchard layouts, while improving the headland maneuver robustness and avoiding any environmental instrumentation. Some of our earlier works have proposed to perform the U-turn using data provided by a 2d laser rangefinder. We have designed a sensor-based nonlinear controller following a spiral centered on the last row tree [14] [15]. This approach was later extended to unify both in-row and headland navigation in a unique spiral-based framework [16] in a straight orchard. Despite promising results regarding the use of a unique sensor-based framework for both parts of the navigation, the solution presented oscillation issues, especially when re-entering the alley. This was due to the idea of modeling the orchard navigation as a point regulation problem where the robot had to reach a sequence of waypoints, *i.e.,* without considering the robot orientation. Moreover, it is necessary to use a more robust perception method. Indeed, although allowing to validate the approach in simulation, a 2d laser rangefinder has a planar field of view, not allowing to detect trees in a robust way in an orchard.

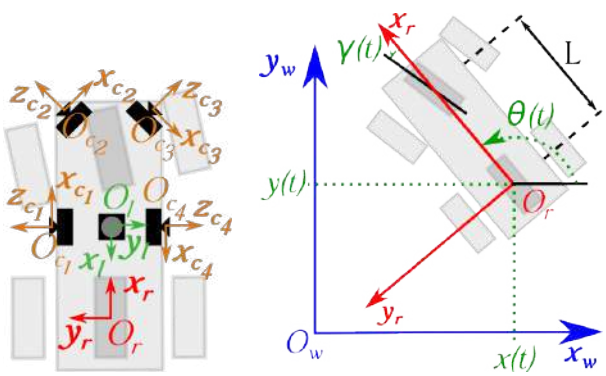
In this paper, we present a novel sensor-based framework allowing the robot to navigate through an entire orchard, *i.e.,* both the alleys and the headlands, without adding any landmark nor considering a particular layout. First, the robot has been equipped with a vision system made of four RGB-D low-cost cameras. On the one hand, it offers a relatively inexpensive solution to acquire 3D data, thus increasing the tree detection capabilities with respect to 2D laser rangefinder-based solutions. On the other hand, it allows benefiting from an overall large field of view to perceive trees both in the row and in the headland, making sensor-based control possible. Second, instead of defining the orchard navigation in terms of point regulation, we now state it as a path-following

[1]Univ. de Toulouse, CNRS, UPS, Toulouse, France
[2]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France {villemaze, cadenat}[at]laas.fr
[3]Departamento de Engenharia de Mecânica, Universidade Federal de Pernambuco UFPE, Recife, PE, Brazil adrien.durandpetiteville[at]ufpe.br
979-8-3503-0704-7/23/$31.00 ©2023 IEEE

problem, to reduce the previous oscillations. The reference path is a local path iteratively updated using the position of the trees computed by the vision system. To do so, we use a Voronoi diagram for row traversals and a spiral model for the headland maneuvers. Next, a Non-Uniform Rational Spline (NURBS) curve is computed to unify the different sections of the path and provides a smooth reference to follow. This problem formulation presents three advantages: (i) the in-row and headland navigation are unified during the path computation and are not merged at the controller level as in [16]; (ii) it allows dealing with numerous orchard layouts (and not only rectangle-shaped ones); and (iii) it offers a more consistent reference than the straight-line following approach and takes into account the robot orientation. The path following is performed using a Nonlinear Model Predictive Control (NMPC) scheme coupled with a Frenet-based formulation of the problem. It provides an efficient minimization of the error between the computed and desired paths over the whole prediction horizon while taking into account specific constraints such as actuator saturation. Finally, in order to evaluate the relevancy of the proposed approach, it is first compared with [16] using the Gazebo simulator, and next implemented on the Hunter 2.0 robotic platform to navigate in an orchard.

The rest of the paper is organized as follows. We first present the robotic system before focusing on the proposed navigation framework. The simulated and experimental results are then presented to show the approach's efficiency.

II. Modelling



(a) The perception system and the related frames.

(b) The robot model.

Fig. 2. The robotic system models.

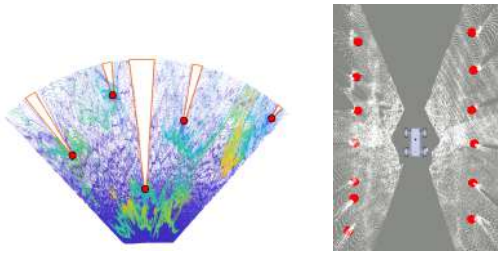
The considered platform is the Agilex Hunter 2.0 car-like robot equipped with a laser rangefinder and four RGB-D cameras (see Fig. 2(a)). To obtain the necessary wide field of view, two cameras are placed at the front of the robot and respectively oriented left forward and right forward, while the two other ones are placed on the sides of the platform (see Fig. 2(a)). To model the system, we define $F_w = (O_w, \mathbf{x_w}, \mathbf{y_w}, \mathbf{z_w})$ as the world frame, $F_r = (O_r, \mathbf{x_r}, \mathbf{y_r}, \mathbf{z_r})$ as the robot frame, $F_l = (O_l, \mathbf{x_l}, \mathbf{y_l}, \mathbf{z_l})$ as the laser frame, and $F_{c_i} = (O_{c_i}, \mathbf{x_{c_i}}, \mathbf{y_{c_i}}, \mathbf{z_{c_i}})$ as the frame of the i^{th} camera, with $i \in [1, 4]$. We rely on the Ackermann model to represent the robot and therefore its pose is given by $\chi(t) = [x(t), y(t), \theta(t), \gamma(t)]$, where $x(t)$ and $y(t)$ are the coordinates of O_r in F_w, $\theta(t)$ represents the angle from $\mathbf{x_w}$ to $\mathbf{x_r}$, and $\gamma(t)$ is the angular position of the steering angle (see Fig. 2(b)). Moreover, we define the control vector by $\mathbf{U}(t) = [\upsilon(t), \gamma(t)]$ where $\upsilon(t)$ is the linear velocity along $\mathbf{x_r}$. For such a system, considering L the distance between the front and rear wheels, the kinematic model is:

$$\begin{cases} \dot{x}(t) = \upsilon(t)\cos(\theta(t)) \\ \dot{y}(t) = \upsilon(t)\sin(\theta(t)) \\ \dot{\theta}(t) = \frac{\upsilon(t)}{L}\tan(\gamma(t)) \end{cases} \tag{1}$$

III. Orchards traversal strategy

To navigate in the orchard, the robot has to cross an alley, maneuver in the headlands to switch from an alley to the next one, and repeat these two steps until its navigation is completed. In this section, we detail the different processes involved in the proposed navigation framework. We first present the vision system and data processing. Next, we introduce our solution to compute the local path to follow both in the alleys and in the headlands. Finally, our NMPC-based path-following strategy is detailed.

A. Data processing



(a) Top view of a point cloud containing shadows due to the presence of trees [6].

(b) Example of four point clouds expressed in F_l.

Fig. 3. Data processing examples.

The presented navigation strategy relies on the position of the tree trunks in the current robot frame. The positions are computed using the point clouds provided by the four onboard RGB-D cameras. To do so, we rely on the algorithm [6] which estimates the tree trunk positions by detecting shadows in the point cloud due to the presence of trees (see Fig. 3(a)). The algorithm processes, therefore, the four point clouds separately and provides the position of the detected trees in each camera frame F_{c_i}.

The tree coordinates must then be expressed in a common frame, which is the laser frame F_l. Indeed, the laser field of view overlaps the one of the four cameras, allowing computing the extrinsic parameters between F_l and the four F_{c_i}. The calibration process between F_l and F_{c_i}, *i.e.,* the computation of the homogeneous transformation matrix $H_{l|c_i}$ is performed using [17]. An example of the result is shown in Fig. 3(b).

B. Path generation

In this section, we present how the tree coordinates in the current robot frame are used at each iteration to generate a new path to follow. The proposed path generation is a two-step process: first, we calculate a set of waypoints, and next, we compute a path based on these waypoints. The waypoints computation is done differently for the alley traversing and the headland navigation. For the alleys, we generate a Voronoi diagram [18] using the tree coordinates. The vertices of the diagram, which approximately lie in the middle of the row, will then be used to compute the path to follow (see steps 1 and 5 in Fig. 5).

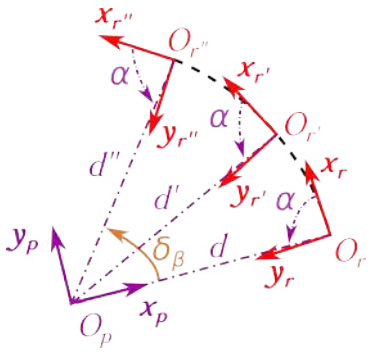


Fig. 4. Several robot frames while describing a spiral.

For the headland maneuver, we propose to compute waypoints lying on a spiral centered on the last tree of the row, called the pivot point and denoted O_p (see step 3 in Fig. 5). It is used as the origin of the frame F_p, whose orientation is arbitrary. We rely on the spiral model presented in [19] where O_p, the pivot point, is considered as the spiral center, $d(t)$ is the distance between the robot and the pivot point, *i.e.,* between O_p and O_r, and $\alpha(t)$ is the oriented angle from the $\mathbf{x_r}$ vector to the $\mathbf{O_rO_p}$ one (see Fig. 4). Finally $\beta(t)$ is the angle between $\mathbf{x_p}$ and $\mathbf{O_pO_r}$. It is shown in [19] that if both $v(t)$ and $\alpha(t)$ are constant, then O_r describes a spiral, and the following equations hold:

$$\dot{d}(t) = -v\cos\alpha \tag{2}$$

$$d(\beta) = d_0 e^{\cot\alpha(\beta_0-\beta)} \tag{3}$$

Eq. (2) shows that the type of spiral only depends on parameter α. Indeed, if $\alpha \in [-\pi;0]$, then O_r turns clockwise with respect to O_p and counter-clockwise if $\alpha \in [0;\pi]$. Moreover, if $\alpha \in]-\pi;-\frac{\pi}{2}[\cup]\frac{\pi}{2};\pi[$, then the spiral is outward and inward if $\alpha \in]-\frac{\pi}{2};0[\cup]0;\frac{\pi}{2}[$. It becomes a circle if $\alpha = \pm\frac{\pi}{2}$, with a radius equal to d. Thus, the design of the spiral first consists in selecting a value for α and an initial distance d_0. Finally, the set of waypoints belonging to the spiral is computed over an angular horizon δ_β using (3). Note, that the frame F_p is readjusted at each iteration to align the $\mathbf{x}_p$ and $\mathbf{O_pO_r}$ vectors. This approach allows maneuvering in the headlands on the sole basis of the current exteroceptive data and does not require any localization process.

The waypoints having been computed for both alleys and headlands, it is then necessary to connect them to make the robot navigate in the orchard. In other words, we have to connect the spiral to the last vertex of the Voronoi diagram to make the robot exit the alley and to connect the spiral to the first vertex of the new diagram when the robot enters a new alley. First, when the robot exits the alley, d_0 is defined as the distance between the pivot point O_p and the last vertex of the diagram in order to connect the two parts of the path. Moreover, we set up $\alpha = \pm\frac{\pi}{2}$ to make the robot follow a circle of radius d_0 centered on the pivot point (see step 2 in Fig. 5). This approach initially makes it possible to safely turn around the pivot point but does not guarantee that the spiral will connect with the first vertex of the next alley diagram. Thus, once the next alley is visible and it is possible to compute the next Voronoi diagram, the spiral parameters are modified. First, α is adjusted to make the spiral pass via the vertex (from here the path is no more a circle, but a spiral), and the angular horizon δ_β is modified to make coincide the end of the spiral with the vertex (see step 4 in Fig. 5). Setting up the spiral parameters as described guarantees the continuity between the different parts of the path.

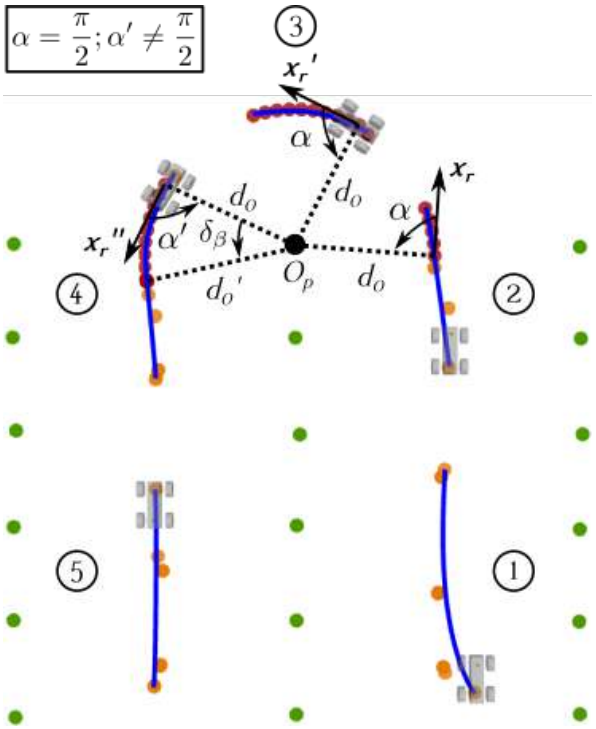


Fig. 5. Examples of path generation. Green circle: tree - Black circle: pivot point - Orange circle: Voronoi vertex - Dark red circle: Spiral point - Blue curve: NURBS - Step 1/5: alley crossing - Step 2: path connecting the alley crossing to the headland maneuver - Step 3: headland maneuver - Step 4: path connecting the headland maneuver to the alley crossing.

Finally, we propose to use a NURBS (Non-Uniform Rational B-Spline) [20] curve to compute a smooth path passing through the waypoints. To summarize, this particular type of curve is defined by a set of weighted control points that

locally influence its curvature. Mathematically, its general form is given by [20]:

$$C(u) = \frac{\sum_{i=1}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=1}^{n} N_{i,p}(u) w_i}, u \in [0,1] \quad (4)$$

where n is the number of control points $\mathbf{P}_i$, w_i are the corresponding weights and $N_{i,p}$ are the B-Spline basis function of p^{th} degree. More details are available in [20]. In our case, the control points are either the endpoints of the Voronoi segments (see orange circles in Fig. 5) or the points belonging to the spiral (see dark red circles in Fig. 5). The NURBS curve was chosen because of its three properties which are useful in our application: the degree of the curve which depends on the number of control points, the knot vector (used by the B-Spline basis function), and the weighted control points. First, the high degree of the curve allows for generating a path for both straight and curved tree rows as well as the circular path for the headland maneuver. Next, the knot vector ensures that the curve passes through the first and the last control points allowing to avoid an abrupt re-alignment of the robot on the reference path. Finally, the weights allow us to adjust the influence of the control points on the curve to make a smooth path and thus obtain a better robot trajectory. We propose to define them as follows: $[1, w_1, ..., w_{n-2}, 1]$. First, the first and last weights are set to 1 with respect to the second property. $w_1, ..., w_{n-2}$ must thus be chosen as a compromise to obtain the most stable path over the iterations.

C. Path following

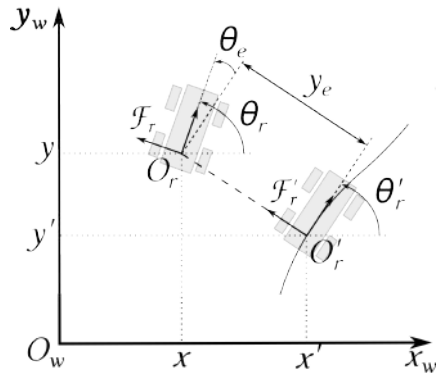


Fig. 6. Principle of the path tracking. [21]

We now present our approach for following a given path using an NMPC controller. As shown in Fig. 6, it consists in orthogonally projecting the center of the robot O_r on the reference path to define a Frenet frame F'_r associated with O_r. It is then possible to define θ_e as the orientation error and y_e as the lateral error. The path following is then performed by minimizing the error vector $\mathbf{e_{pf}} = [y_e, \theta_e]$ over a prediction horizon. This approach does not require including the linear velocity in the minimization problem as it is not aiming at reaching a set of points at a given instant sampled from the path, such as in [16]. The linear velocity can be fixed at a constant value or computed accordingly to a different criterion, such as terrain traversability. Thus, in this work, the linear velocity $v(t)$ is considered constant so that $v(t) = v, (v \neq 0)$. The only control input is thus the steering angle γ. The path following is performed via an NMPC scheme considering the following optimization problem:

$$\overline{\gamma}^*(k) = \min_{\overline{\gamma}(k)} (J_{N_p}(\mathbf{e_{pf}}(k), \overline{\gamma}(k))) \quad (5)$$

with

$$J_{N_p}(\mathbf{e_{pf}}(k), \overline{\gamma}(k)) = \sum_{p=k+1}^{k+N_p} \hat{\mathbf{e}}_{pf}(p)^T \hat{\mathbf{e}}_{pf}(p) + \lambda_\gamma (\gamma(p) - \gamma(p-1))^2 \quad (6)$$

subject to

$$\hat{\mathbf{e}}_{pf}(p+1) = f(\hat{\mathbf{e}}_{pf}(p), \gamma(p)) \quad (7a)$$

$$\hat{\mathbf{e}}_{pf}(k) = \mathbf{e_{pf}}(k) \quad (7b)$$

$$C(\overline{\gamma}^*(.)) \leq 0 \quad (7c)$$

It computes an optimal steering angle sequence $\overline{\gamma}^*(k)$ of $\overline{\gamma}(k)$, with $\overline{\gamma}(k) = [\gamma(k), ..., \gamma(k+N_p)]$ which minimizes the cost function J_{N_p} over a prediction horizon of N_p steps while taking into account the physical boundaries of the robot actuators as constraints $C(\overline{\gamma}^*(k))$. The values of both the prediction and control horizons are considered equal.

Cost function: J_{N_p} is divided in two parts. The first one is defined as the sum of the quadratic predicted configuration $\hat{\mathbf{e}}_{pf}$, and is intended to track the reference path. The second one is the sum of the quadratic differences between two consecutive commands, weighted by the parameter λ_γ, which allows smoothing of the control inputs and limiting velocities variations between two instants.

Remark: To project the predicted positions onto the reference path, we discretize the NURBS curve and search for the closest position belonging to the path for each prediction. The search relies on the k-d tree structure [22] which proposes an efficient nearest neighbor search based on a space-partitioning data structure.

Prediction model: Assuming that the steering angle $\gamma(t_1)$ is constant between the instant t_1 and $t_2 = t_1 + T_s$, where T_s is the sampling time, the robot predicted pose is computed by integrating (1) with a Runge-Kutta method of order 4.

Input constraints: The input constraints take into account the physical limits of the mobile base. They are given by:

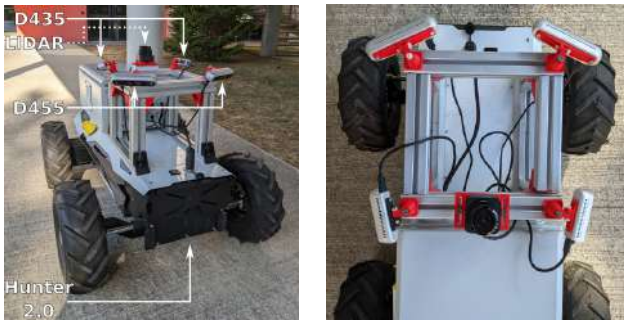
$$\begin{bmatrix} \gamma(i) - \gamma_u \\ \gamma_l - \gamma(i) \end{bmatrix} \leq 0 \quad (8)$$

where $i \in [1, N_p]$, γ_l and γ_u are respectively the lower and upper boundaries.

IV. Results

In this section, we present the obtained results, first using a simulator, then using a robotic platform. In both cases, the considered robot is the Hunter 2.0 car-like mobile base. The robot is equipped with a vision system consisting of

four Intel Realsense RGB-D cameras, two D455 and two D435, positioned as shown in Fig. 7(b) to enlarge the field of view as explained earlier. The robot has also been endowed with Slamtech's RPLIDAR S1 range-finder for the camera calibration step. The physical boundaries of the Hunter 2.0 actuators as well as the optimal ranges of the cameras are shown in Table I.



(a) Side view. (b) Top view.

Fig. 7. Robotic platform.

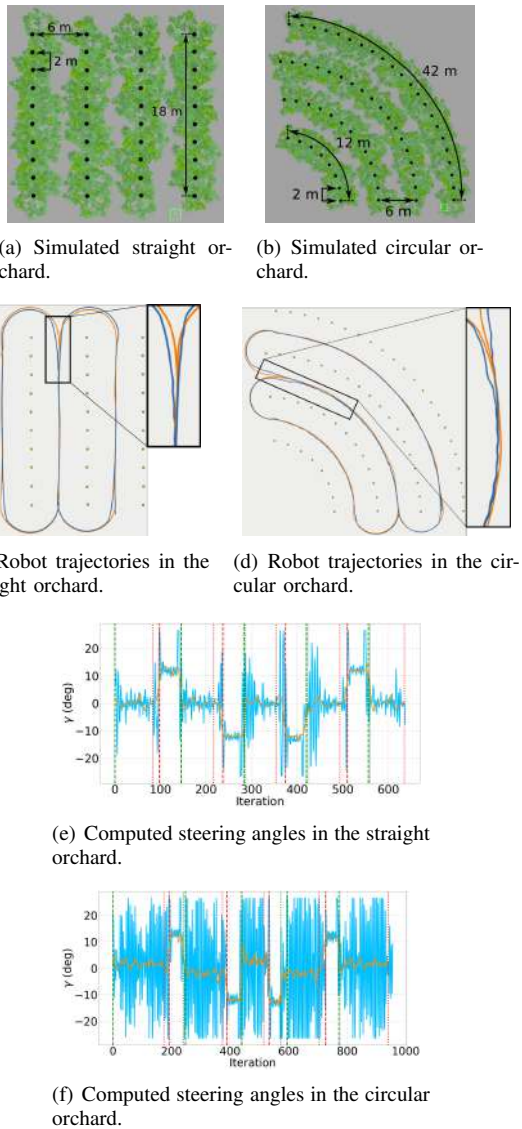
TABLE I
SYSTEM SPECIFICATIONS.

	minimum range	maximum range
Linear velocity	−1.5 m/s	1.5 m/s
Steering angle	−0.461 rad	0.461 rad
D455	0.6 m	6 m
D435	0.3 m	3 m

Furthermore, the robot is equipped with an NVIDIA Jetson Xavier NX GPU and an Intel Core i7-1165G7 48 GB RAM CPU. The former is dedicated to data processing while the latter calculates the control inputs. The implementation relies on the C++ 14 language and the ROS middleware. The data processing part uses the OpenCV and PCL libraries and is partially implemented using the CUDA language. The NMPC part is based on several libraries allowing to implement the following features: the clustering method, the Voronoi diagram, the NURBS curve, the k-d tree structure and the SQP solver.

A. Simulation

We first compare the proposed approach, the NURBS-based method, with the one described in [16], the spiral-based method. We recall that the NURBS-based method relies on a path following while the spiral-based one consists in reaching a sequence of positions. The simulations are performed with the straight and circular orchards shown in Fig. 8(a)) and Fig. 8(b)) where the trees' position and orientation were randomly modified to obtain a more realistic layout. The parameters for both methods are listed in Table II. For the spiral-based method, the set of parameters is similar to the one used in [16] with the exception of the solver tolerance values which are slightly modified to increase performance in the circular orchard. In addition to using a different cost function, path-following vs. positioning, the methods differ in their use of a terminal constraint. Indeed, the spiral-based method requires a terminal constraint to guarantee the stability of the positioning process while it is not required for the path-following approach. Finally, the lower/upper limits of the input constraints γ_l and γ_u in the NURBS-based method are no longer the physical limits of the steering angle of the Hunter 2.0 actuators, as in the spiral-based method, but the maximum positions reachable in T_s second ($\pm$ 2 degrees for the Hunter 2.0). This allows only feasible commands to be calculated for the robot, thus reducing solver disturbances between iterations and improving robot behavior.



(a) Simulated straight orchard. (b) Simulated circular orchard.

(c) Robot trajectories in the straight orchard. (d) Robot trajectories in the circular orchard.

(e) Computed steering angles in the straight orchard.

(f) Computed steering angles in the circular orchard.

Fig. 8. Navigation results in simulation. (c-d-e-f) Blue plots: spiral-based method results - Orange plots: NURBS-based method results - (e-f) Green vertical lines: Start of the alley crossing for the spiral-based method (dashed lines) and the NURBS-based one (dotted lines) - Red vertical lines: Start of the headland maneuver for the spiral-based method (dashed lines) and the NURBS-based one (dotted lines).

Figure 8 presents the results obtained for both methods and orchard layouts. In Fig. 8(c) and Fig. 8(d), it can be seen that the robot successfully achieves the navigation

TABLE II
SIMULATION PARAMETERS.

Approach	v	T_s	N_p	Maximum time[a]	Absolute tolerance[a]	ϕ_{ZEC} [b]	Number of points NURBS	w_i	λ_γ
Spiral	1	0.2	12	0.16	10^{-3}	10^{-4}	N/A	N/A	N/A
NURBS	1	0.1	20	0.09	10^{-6}	N/A	3000	10^{-2}	10

[a]Stopping criterion of the SQP solver.
[b]Zero terminal equality constraint tolerance.

task in the straight and curved orchards relying on both the spiral-based and the NURBS-based methods. Indeed, it successfully drives through the three alleys and maneuvers in the headlands to switch from one alley to the next one performing a 126 meters long path in the straight orchard and a 187 meters long one in the circular one. Thus, from a task point of view, both approaches are capable of navigating in different orchard layouts unlike other works focusing on straight lines. However, from a control perspective, it can be noticed that the spiral-based method tends to generate oscillations when the path is curved (entrance of a new alley or in the alleys of the circular orchard). This is due to the fact that it is a positioning approach not taking into account the robot's orientation. Thus, as long as the robot is oriented toward the next goal point, it navigates without oscillating, *e.g.,* when crossing the straight alleys. However, when it is not initially oriented toward the point to reach, it has a tendency to oscillate *e.g.,* when entering a new alley or driving through a curved alley. On the contrary, the NURBS-based method presented in this paper does not lead to such oscillations. Indeed, using a path-following formulation of the problem allows for taking into account the robot's orientation. Thus, the quality of the robot path is consistent when navigating in a straight or curved alley or when maneuvering in the headlands. This analysis is supported by the evolution of the steering angles shown in Fig. 8(e) and 8(f). Indeed it can be seen that the value of the steering angle varies more for the spiral-based approach than for the NURBS-based one, for both orchard layouts. Thus, despite the interest in the spiral-based method, the NURBS-based method significantly improves the quality of the navigation system.

B. Experimentation

To show the efficiency of the NURBS-based method, we conducted an experiment at the agricultural high school[1] in Auzeville-Tolosane, France. The considered orchard has 40 meters long by 4 meters wide tree rows with a space of 1 meter between two consecutive trees. At the time of the experiments, only four tree rows were usable. The following parameters were chosen: $v = 0.5m/s$, $T_s = 0.1s$, $N_p = 20$, which corresponds roughly to a prediction window of 1 meter, and $\lambda_\gamma = 5$. The other parameters remain identical to the simulation.

The orchard navigation is presented in the attached video. Some additional snapshots, completed with an RVIZ view of the detected trees and the computed path, display the main key steps of the navigation in Fig. 9. As shown in the video, the navigation task is correctly achieved. The robot successfully moves along the three alleys twice and performs two clockwise and two counter-clockwise U-turn maneuvers in the headlands. It has thus realized a 222 meters long path in 480 seconds. Now, let us go into further details and analyze the main steps of the navigation: the sequence of row followings and U-turn maneuvers. First, the system successfully computes a path based on the tree positions allowing to drive through the alley (see Fig. 9(a)). The path computing/following process is repeated at each iteration

(a) Alley crossing.

(b) Beginning of the clockwise headland maneuver.

(c) Clockwise headland maneuver.

(d) Counter-clockwise headland maneuver.

(e) Entering the next alley.

Fig. 9. Navigation snapshots - left: robot centered RVIZ data visualization (green circle: detected trees - blue circle: selected pivot point - orange circle: NURBS control points - blue curve: NURBS) - right: video screenshots.

[1]"Lycée Général et Technologique Agricole des Sciences Vertes"

until the row crossing is achieved. Once the robot gets closer to the end of the alley, one of the last trees is selected as the pivot point and the generated path is composed of both a row crossing section and a spiral one (see in Fig. 9(b)). The pivot point is chosen so that the robot makes a loop in the orchard and thus sequences the four U-turns. Next, the robot performs the clockwise/counter-clockwise headland maneuver following a spiral computed on the sole basis of the pivot point as seen in Fig. 9(c) and 9(d). Finally, in Fig. 9(e) the robot is about to reach the next alley. The spiral parameters are adjusted to connect the spiral section of the computed path to the row-crossing section. By doing so, the robot manages to enter the next alley and then restart the crossing step.

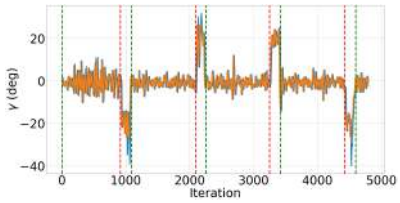


Fig. 10. Computed and applied steering angles. Blue line: computed steering angles - Orange line: Applied steering angles - Green vertical dashed line: beginning of the alley crossing - Red vertical dashed line: beginning of the headland maneuver.

Finally, the computed and applied commands are displayed in Fig. 10. As shown in this figure, the computed steering angle tends towards 0 degrees during the alley crossings and towards $\pm$ 18 degrees during the headland maneuvers, which is consistent with the orchard layout. The variations are mainly due to the variations of the computed tree coordinates. Indeed, these latter are computed on the sole basis of the current data and the results may differ from one iteration to the other. As the command frequency rate is higher than the steering angle capabilities, the robot path is not impacted by these oscillations. This leads to appropriate overall behavior and thus validates the control strategy.

V. Conclusion

This paper presents a novel multi-camera-based NMPC strategy allowing autonomously navigating through various shaped orchards without instrumentation. The proposed method relies on an original fully vision-based computation and update of the reference path and does not require any map. The path following problem is expressed using the NMPC framework, making easier the transition between in-row and headland navigation and the constraints handling. The approach has been implemented and validated through an experimental campaign conducted in an orchard. The obtained results show the relevance and efficiency of the approach. Regarding future works, we plan to increase the perception system robustness by adding a particle filter able to track the trees and coupling the point processing to an image-based tree detection. We also aim at integrating new constraints in NMPC to avoid obstacles and to reduce undesired vibrations due to rough terrains.

References

[1] J. F. Reid, "The impact of mechanization on agriculture," *Bridge*, vol. 41, no. 3, pp. 22–29, 2011.
[2] M. Li, K. Imou, K. Wakabayashi, and S. Yokoyama, "Review of research on agricultural vehicle autonomous guidance," *International Journal of Agricultural and Biological Engineering*, vol. 2, no. 3, pp. 1–16, 2009.
[3] J. Radcliffe, J. Cox, and D. M. Bulanon, "Machine vision for orchard navigation," *Computers in Industry*, vol. 98, pp. 165–171, 2018.
[4] S. Opiyo, C. Okinda, J. Zhou, E. Mwangi, and N. Makange, "Medial axis-based machine-vision system for orchard robot navigation," *Computers and Electronics in Agriculture*, vol. 185, p. 106153, 2021.
[5] J. Gai, L. Xiang, and L. Tang, "Using a depth camera for crop row detection and mapping for under-canopy navigation of agricultural robotic vehicle," *Computers and Electronics in Agriculture*, vol. 188, p. 106301, 2021.
[6] A. Durand-Petiteville, E. Le Flecher, V. Cadenat, T. Sentenac, and S. Vougioukas, "Tree detection with low-cost three-dimensional sensors for autonomous navigation in orchards," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3876–3883, 2018.
[7] P. M. Blok, K. van Boheemen, F. K. van Evert, J. IJsselmuiden, and G.-H. Kim, "Robot navigation in orchards with localization based on particle filter and kalman filter," *Computers and Electronics in Agriculture*, vol. 157, pp. 261–269, 2019.
[8] A. Danton, J.-C. Roux, B. Dance, C. Cariou, and R. Lenain, "Development of a spraying robot for precision agriculture: An edge following approach," in *2020 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2020, pp. 267–272.
[9] A. Favery. The gotheron orchard: when biodiversity becomes circular. [Online]. Available: https://www.inrae.fr/en/news/gotheron-orchard-when-biodiversity-becomes-circular
[10] S. G. Vougioukas, "Agricultural robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 365–392, 2019.
[11] V. Subramanian and T. F. Burks, "Autonomous vehicle turning in the headlands of citrus groves," in *2007 ASAE Annual Meeting*. American Society of Agricultural and Biological Engineers, 2007, p. 1.
[12] G. Bayar, M. Bergerman, A. B. Koku, and E. Ilhan Konukseven, "Localization and control of an autonomous orchard vehicle," *Computers and Electronics in Agriculture*, vol. 115, pp. 118–128, 2015.
[13] J. Zhang, S. Maeta, M. Bergerman, and S. Singh, "Mapping orchards for autonomous navigation," in *2014 Montreal, Quebec Canada July 13–July 16, 2014*. American Society of Agricultural and Biological Engineers, 2014, p. 1.
[14] A. Durand-Petiteville, E. Le Flecher, V. Cadenat, T. Sentenac, and S. Vougioukas, "Design of a sensor-based controller performing u-turn to navigate in orchards," in *International Conference on Informatics in Control, Automation and Robotics*, vol. 2, 2017, pp. 172–181.
[15] E. Le Flecher, A. Durand-Petiteville, F. Gouaisbaut, V. Cadenat, T. Sentenac, and S. Vougioukas, "Nonlinear output feedback for autonomous u-turn maneuvers of a robot in orchard headlands," in *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2019.
[16] A. Villemazet, A. Durand-Petiteville, and V. Cadenat, "Autonomous navigation strategy for orchards relying on sensor-based nonlinear model predictive control," in *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 05 2022.
[17] Y. Li, Y. Ruichek, and C. Cappelle, "3d triangulation based extrinsic calibration between a stereo vision system and a lidar," *Conference Record - IEEE Conference on Intelligent Transportation Systems*, pp. 797–802, 10 2011.
[18] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations - Concepts and Applications of Voronoi Diagrams*. John Wiley, 2000.
[19] K. N. Boyadzhiev, "Spirals and conchospirals in the flight of insects," *The college mathematics Journal*, vol. 30, no. 1, p. 23, 1999.
[20] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed. New York, NY, USA: Springer-Verlag, 1996.
[21] V. Cadenat, P. Souéres, and T. Hamel, "A reactive path-following controller to guarantee obstacle avoidance during the transient phase," *International Journal of Robotics and Automation*, vol. 21, no. 4, pp. 256–265, 2006.
[22] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, p. 509–517, sep 1975. [Online]. Available: https://doi.org/10.1145/361002.361007

Learned Long-Term Stability Scan Filtering for Robust Robot Localisation in Continuously Changing Environments

Ibrahim Hroob*, Sergi Molina, Riccardo Polvara, Grzegorz Cielniak and Marc Hanheide

***Abstract*—In field robotics, particularly in the agricultural sector, precise localization presents a challenge due to the constantly changing nature of the environment. Simultaneous Localization and Mapping algorithms can provide an effective estimation of a robot's position, but their long-term performance may be impacted by false data associations. Additionally, alternative strategies such as the use of RTK-GPS can also have limitations, such as dependence on external infrastructure. To address these challenges, this paper introduces a novel stability scan filter. This filter can learn and infer the motion status of objects in the environment, allowing it to identify the most stable objects and use them as landmarks for robust robot localization in a continuously changing environment. The proposed method involves an unsupervised point-wise labelling of LiDAR frames by utilizing temporal observations of the environment, as well as a regression network called Long-Term Stability Network (LTS-NET) to learn and infer 3D LiDAR points long-term motion status. Experiments demonstrate the ability of the stability scan filter to infer the motion stability of objects on a real agricultural long-term dataset. Results show that by only utilizing points belonging to long-term stable objects, the localization system exhibits reliable and robust localization performance for long-term missions compared to using the entire LiDAR frame points.**

I. Introduction

Accurate and reliable localization is essential for the autonomous navigation of robots and self-driving vehicles, particularly in environments that undergo significant changes [1], such as agricultural fields (Fig. 1). The Real-Time Kinematic Global Positioning System (RTK-GPS) is a widely used localization technique in field robotics, where high precision is required. RTK-GPS can provide localization accuracy of up to a few centimeters by using a combination of satellite-based positioning and ground-based reference stations [2]. However, RTK-GPS systems come with certain limitations such as high cost, reliance on subscriptions and external infrastructure, susceptibility to environmental factors and weather conditions, which can lead to degraded performance and reliability.

As an alternative, localization methods based on map building from onboard sensors, such as cameras [3] or LiDAR [4], may offer more robust and accurate solutions at a lower cost. LiDAR sensors, in particular, provide precise range information and has been found to be more robust than camera-based localization, as it is immune to changes in illumination and can provide accurate range information even in low-light conditions [6]. In this work, we focus on LiDAR-based localization and its potential as a more reliable for autonomous navigation in continuously changing environments such as agricultural fields.

Fig. 1: The images show a visual comparison of the significant changes in an outdoor agricultural orchard throughout the season. The image on the left represents the initial stage of the environment when deploying the robot in March, and the right image represents the fully grown stage in June [5].

In agricultural environments or outdoor fields in general, map-based localization systems often fail to perform reliable localization for long-term missions over extended periods of time (such as months or even years) due to constant changes in the environment [7], resulting in the initial map becoming quickly outdated. Simultaneous Localization and Mapping (SLAM) algorithms have demonstrated robust performance in estimating the robot pose in dynamic environments [8]. However, they are often prone to failure when used for long-term operations due to false positives in data association between localization sessions. False positive cases occur when an incorrect match is made between an object in the environment and a sensor measurement, which is due to significant changes between the map and the measurements. This problem has been well documented in literature, for example in [9].

Despite the changes that could occur in the environment, some parts of it remain static, which could be used as a landmark for achieving accurate and reliable long-term localization. We hypothesise that a learned filter, capable of distinguishing between static and dynamic parts of a scene, will significantly enhance long-term localization accuracy by reducing the impact of dynamic objects. To achieve this, we propose an unsupervised scan filter learning for robust robot localisation in long-term changing environments. This method leverages previous observations to recognize objects

*Corresponding author: ihroob@lincoln.ac.uk
All the authors are within the Lincoln Centre for Autonomous Systems (LCAS), University of Lincoln, Lincoln, UK, LN6 7TS.
979-8-3503-0704-7/23/$31.00 ©2023 IEEE

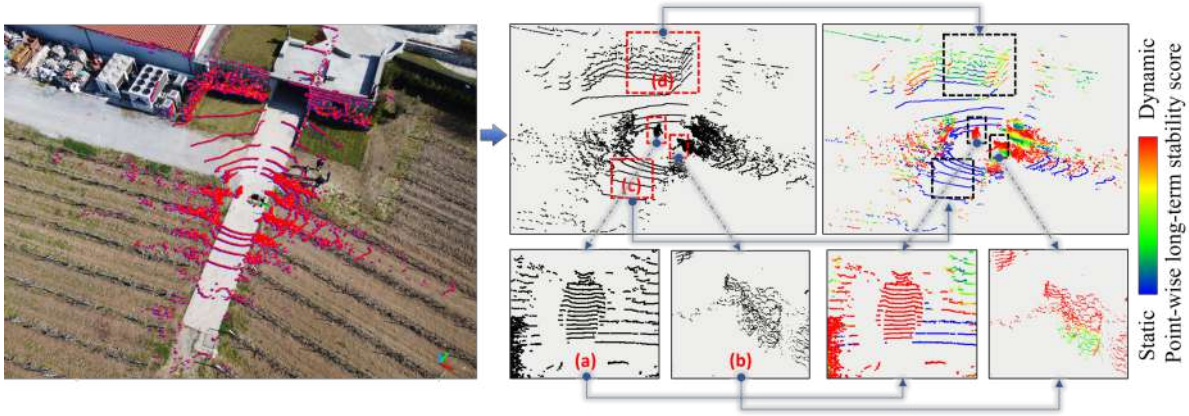


Fig. 2: The BLT-Dataset was used to create a LiDAR scan visualization. The left image shows an aerial view of the scan location, highlighting the stable structures in the environment. The middle image is the raw scan. The right image displays the predicted spatial-temporal stability labels from LTS-NET, with points inferred into these categories: (a) fast-moving objects such as humans, (b) visual appearance changing objects like seasonal vegetation changes, (c) ground plane points, and (d) long-term stable points like those belonging to a building.

that maintain stability over time and generates training data for a deep learning model. This model can then be used on future scans to directly identify stable objects within 3D LiDAR frames as illustrated in Fig. 2, allowing us to filter out dynamic points and rely on stable structures for extended localization.

Our key contributions in this paper include: (1) an automated point-wise labelling method for LiDAR scans. (2) The Long-Term Stability Network (LTS-NET), a regression network designed to infer LiDAR points spatiotemporal stability. (3) An analysis of a real-world long-term vineyard dataset shows that using filtered scans enhances the accuracy and robustness of 3D LiDAR localization algorithms that use scan matching techniques, compared to using raw scans. (4) We show that the trained regression model can directly predict objects' stability in a new environment without the need for a prior sequence of temporal observations. The code of this paper as well as our pre-trained model is available as a docker image at https://github.com/LCAS/lts_filter.

II. Related work

There have been numerous approaches proposed for long-term localization, ranging from traditional methods based on odometry and map-based approaches to more recent approaches that utilize deep learning techniques.

One common approach to long-term localization is to use a pre-built map of the environment and match current sensor readings to the map to determine the robot's position. This approach can be effective in environments with stable, distinct features, but can be less reliable in environments with more dynamic or changing features. To address this limitation, some studies have explored the use of additional structural information in the initial map to increase robustness, such as Gaussian Mixture probabilistic maps [10]. Others have extracted pole-like landmarks for long-term localization [11].

In some approaches to long-term localization, the map is updated to reflect changes in the environment. Researchers have addressed this in several ways. For instance, some studies have accumulated visual experience of the same place over time and used it for localization [12], [13]. This approach has been shown to handle some level of environmental change, but can lead to very large map sizes. Other solutions have employed mathematical models to predict changes in the environment [14], [15]. For example, FreMEn [14] is a frequency map enhancement approach that considers regular feature points in visual SLAM as a combination of harmonic functions. This method has been shown to improve robot localization in indoor and outdoor environments.

Another approach to long-term localization is the use of temporal mapping, which involves building a temporary map when global matching is unreliable and merging it with a pre-built map for use in later localization runs [16]. However, this approach may not be suitable for continuously changing environments as the appearance can change dramatically, requiring the method to enter the mapping phase repeatedly.

Recent advancements in long-term localization rely on the utilization of deep learning models for extracting long-term stable features, which could be either from 3D point cloud data [17], [18] or visual data [19], [20]. The models can be trained to identify stable features in the environment, such as tree trunks or light posts, and use them as landmarks for achieving long-term localization. However, these methods primarily target urban structures and require manual annotation of data for model training. The agricultural domain has seen the development of visual localization and mapping techniques to identify specific environmental features such as tree trunks [21], [22]. These methods facilitate long-term operations but are limited by the need for manual data annotation which is prone to human error and time-consuming [23]. Additionally, visual data is vulnerable to changes in lighting conditions [6].

In this paper, our method differs from existing state of the art methods by eliminating the need for costly manual

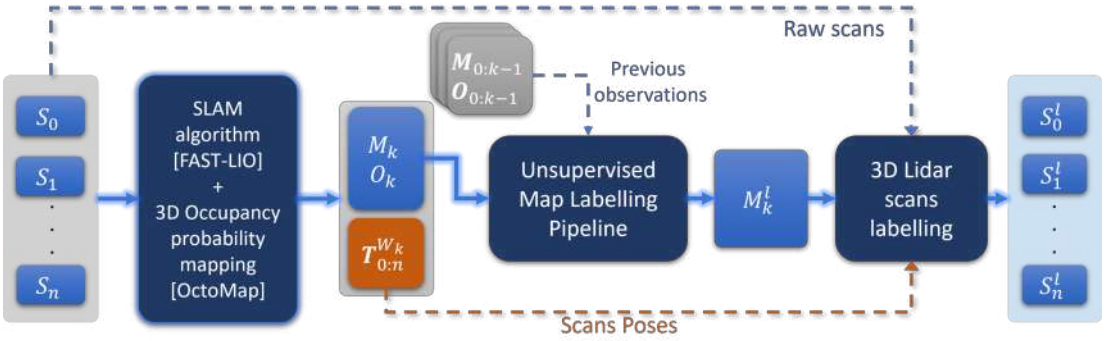


Fig. 3: The unsupervised data labelling pipeline for 3D LiDAR frames $\mathbf{S} : \{\mathcal{S}_0, \ldots, \mathcal{S}_n\}$. First we build a point cloud map $\mathbf{M}_k$ using a SLAM system, then we assign the long-term stability labels to $\mathbf{M}_k$ by exploring the previous maps $\mathbf{M}_{0:k-1}$. Finally the stability labels are projected back to the LiDAR frames by using there poses $T^{W_k}_{0:n}$.

annotation in labelling LiDAR frames. Our approach is data-driven and capable of adapting to any environment. It leverages the temporal characteristics of the environment's history to train a deep learning model, enabling it to learn about the long-term stability of objects directly from LiDAR frames.

III. Proposed Method

We propose a generic learnable stability scan filter to learn and extract the inherent stable structure (i.e. landmarks) of a given environment from 3D LiDAR scans, then filter other objects to achieve robust localization over extended periods of time. To accomplish this, we utilize a temporal sequence of 3D point cloud maps $\mathbf{M}_{0:k}$, and their associated LiDAR frames $\{\mathbf{S}\}_{0:k}$. Our framework consists of an algorithm for assigning spatiotemporal features for $\mathbf{M}_k$ with respect to previous observations $\mathbf{M}_{0:k-1}$, then projecting those features back to their associated LiDAR frames $\{\mathbf{S}\}_k$. Second, a neural network, $f(.)$, that learns the spatiotemporal features (i.e. stability score) from the labelled LiDAR frames to predict the spatiotemporal features of the next session $\{\mathbf{S}\}_{k+1}$.

A. Automated labelling of 3D LiDAR frames

Manual labelling of 3D LiDAR frames can be a challenging and time-consuming task, especially when dealing with large volumes of data [23]. It requires a significant amount of human effort, and there is always the risk of human error. Moreover, the type of labels applied to the point cloud of LiDAR frames varies depending on the targeted application. For instance, semantic segmentation tasks may benefit from full class segmentation, while tasks such as distinguishing moving from stable objects may only require binary labels. In this work, we investigate the utilization of continuous labels to represent the spatiotemporal stability of the point cloud.

The continuous labels are assigned based on the points of a spatiotemporal dependency across multiple time slices of the environment. The spatiotemporal information can capture objects' long-term motion status. To label the LiDAR frames, we first build a point cloud for all the observations, perform the labelling on them, then we project the labels back to the frames. The pipeline is summarized in Fig. 3, here the process in more detail.

Maps labelling To accurately label points in the map based on their long-term stability, our approach requires at least two observations of the environment. Given the LiDAR frames $\mathbf{S} : \{\mathcal{S}_0, \ldots, \mathcal{S}_n\}$ (n is the number of the frames) and IMU data for each observation, we first build a point cloud map using mapping system (such as FAST-LIO [8]), and we save the transformation matrix $T^{W_k}_{0:n} \in \mathbb{R}^4$ for each LiDAR frame, where W_k is the world frame of the point cloud map k.

Second, while building the point cloud map, we also construct an occupancy probability map using OctoMap [24]. This step allows for the representation of uncertainty about the state of the environment, which is useful when calculating the points' features at later stages. The resulted point cloud map with its occupancy probability is represented as follows:

$$\begin{aligned} \mathbf{M}_k &= \{\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_m\}, \\ \rho(\boldsymbol{P}) &= p(\boldsymbol{P}|\mathbf{O}_k), \end{aligned} \tag{1}$$

where $\boldsymbol{P}_i = (x_i, y_i.z_i), \boldsymbol{P}_i \in \mathbb{R}^3$ is the 3D coordinate of the i-th point, m is the total number of points, $\mathbf{O}_k$ is the octree data structure that represents the 3D occupancy grid and $\rho(\boldsymbol{P})$ is the probability of a point location exists in a given state (occupied, free or unknown) based on $\mathbf{O}_k$, where unknown in this context means that the location of the point was either not scanned or occluded by other objects.

Then we segment the ground plane using the Cloth Simulation Filter (CSF) [25], where $\mathbf{M}^{OG}_k, \mathbf{M}^{G}_k = CSF(\mathbf{M}_k)$ that is **O**ff-**G**round and **G**round maps respectively. We assign a value of 0 to all ground points to ensures the points are labelled with the same value (the labelled ground is denoted as $\mathbf{M}^{G,L}_k$), and segmenting $\mathbf{M}^{G}_k$ increases the disparity when calculating the points' features at later stages for the off ground maps.

To ensure robust data association between the temporal observations, we geometrically align all the off-ground point cloud maps w.r.t the initial off-ground map. To achieve that we utilize the Iterative Closest Point (ICP) [26] algorithm to perform the registration (i.e. alignment) process. The resulting

transformation matrix $T^{M_0^{OG}}_{M_k^{OG}}$, where M_k^{OG} and M_0^{OG} denote the off-ground point cloud maps of the k-th and initial point cloud map respectively, is then used to transform the nodes of the associated octomap $\mathbf{O}_k$ and the labelled ground plane $\mathbf{M}_k^{G,L}$ to the new coordinates of the transformed map.

In the next step, we extract the stability features, referred to as labels, from the off-ground point cloud maps. The process of feature extraction, also known as labelling, is outlined in Algorithm. 1. Initially, we select a map to be labelled ($\mathbf{M}_k^{OG}$). For each point $\boldsymbol{P} \in \mathbf{M}_k^{OG}$, we first find the occupancy probability $\rho(\boldsymbol{P})$ of the point location in occupancy grid $\mathbf{O}_i$ of the query map $\mathbf{M}_i^{OG}$, if the location is not occluded we find the closest point $\boldsymbol{q}$ using k-Nearest Neighbors (KNN) algorithm, then we compute the spatial distance (d) between $\boldsymbol{P}$ and $\boldsymbol{q}$ and append it to the distance vector $\mathbf{d} = [\mathbf{d}, d]$. On the other hand, if the point location was occluded, we append -1 to the distance vector $\mathbf{d} = [\mathbf{d}, -1]$. After querying all other maps, the point label l of $\boldsymbol{P}$ is set using the maximum spatial distance of $\mathbf{d}$ as a feature, then we map the value using the cumulative distribution function of an exponential function as follows:

$$\boldsymbol{P}.l = 1 - e^{-\max(\mathbf{d})}, \tag{2}$$

where the final label value is a score bounded between 0 and 1 indicating the point spatiotemporal stability.

At the end of the labelling process, the labeled map may contain some noise due to occlusion or mislabeled points. To fix this, we introduce a Voting Median Filter (VMF) based on the labels of nearest neighbors points:

$$f_{med}(i) = \text{median}\{l_j \mid j \in \mathcal{NN}_i(kn)\}, \tag{3}$$

$f_{med}(i)$ is the filtered label for the point $\boldsymbol{P}_i$, l_j is the label of the point $\boldsymbol{P}_j$, $\mathcal{NN}_i(kn)$ is the set of kn nearest neighbors of point $\boldsymbol{P}_i$, and median is the median function that returns the middle value of a set of values. An illustration of the impact of applying VMF on the labelled map is presented in Fig. 4.

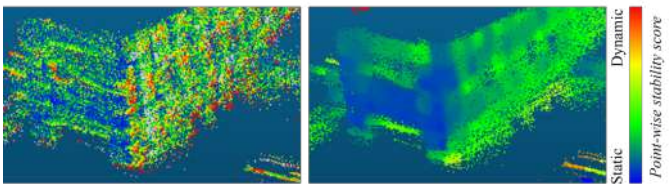


Fig. 4: The impact of applying the VMF on the raw spatiotemporal feature map can be seen on the right, resulting from the input on the left which depicts part of a building.

Finally, the filtered labeled off-ground map will be combined with the corresponding ground point cloud map $\mathbf{M}_k^{G,L}$, to form the final labelled map $\mathbf{M}_k^L = \mathbf{M}_k^{OG,L} \cup \mathbf{M}_k^{G,L}$, which will be used for labelling the LiDAR frames. In summary our approach for labelling the points is based on the assumption that long-term stable objects should appear in the same geometrical location across all temporal observations, thus the associated label value should be smaller compared to dynamic objects.

Algorithm 1: Unsupervised point-wise labelling algorithm

inputs : A set of filtered and aligned maps $\mathbf{M}_{0:k}^{OG}$
output: A labeled map $\mathbf{M}_k^L$
foreach $\boldsymbol{P} \in \mathbf{M}_k^{OG}$ **do**
  Initialize closest distance vector: $\mathbf{d} = \{\}$
  foreach $\mathbf{M}_i^{OG} \in \mathbf{M}_{0:k-1}^{OG}$ **do**
    if $p(\boldsymbol{P}|\mathbf{O}_i) \neq Unknown$ **then**
      $\boldsymbol{q} \leftarrow KNN(\mathbf{M}_i^{OG}, \boldsymbol{P})$
      $\mathbf{d}.append(\sqrt{\sum_{i=1}^{3}(q_i - p_i)^2})$
    else
      $\mathbf{d}.append(-1)$ ▷ $\boldsymbol{P}$ is occluded in $\mathbf{M}_i^{OG}$
  Point label: $\boldsymbol{P}.l = 1 - e^{-\max(\mathbf{d})}$
Filter $\mathbf{M}_k^{OG,L}$ using median filter Eq. 3
$\mathbf{M}_k^L = \mathbf{M}_k^{OG,L} \cup \mathbf{M}_k^{G,L}$

LiDAR frames labelling: To propagate the features/labels from the labelled map $\mathbf{M}_k^L$ back to its LiDAR frames $\mathbf{S} : \{\mathcal{S}_0, \ldots, \mathcal{S}_n\}$, we first transform the frame $\mathcal{S}_i$ coordinate to its associated map $\mathbf{M}_k^L$ coordinate using the transformation matrix $T^{M_0^{OG}}_{M_k^{OG}}.T_i^{W_k}$, where $i \in n$ is the frame number. Then, we use nearest-neighbour interpolation to propagate the labels back to the frame. Finally, we transform the frame back to its original coordinates.

B. LTS-NET

We present the Long-Term Stability NETwork (LTS-NET) Fig. 5, a regression network that is capable of learning the spatiotemporal labels from the auto labelling algorithm directly on point cloud data. LTS-NET utilize the PointNet++ architecture [27], which has been shown to be effective in processing point cloud data directly. The input to the LTS-NET is a 3D LiDAR frame, represented as sets of 3D point coordinates $\mathcal{S}_i : \{(x_0, y_0, z_0) \ , \ldots, (x_{nn}, y_{nn}, z_{nn})\}$, where nn is the number of points in the frame (we only utilize the points coordinates as a features).

LTS-NET Encoder: In the encoder component, we employ 4 abstraction layers (down-sampling and feature concatenation layers) to aggregate local features from the previous layer into a global feature representation for each point set. To maintain consistent abstraction across layers and prevent information loss or distortion due to varying levels of down-sampling, we use equal numbers of input points in the first two layers (2048 points) and in the last two layers (1024 points). The features sampling radius for each input point across the layers is set to 0.2, 0.4, 0.8, and 1 m, respectively, to facilitate the learning of local geometry and its spatiotemporal stability

LTS-NET Decoder: In the decoder, we use 4 feature propagation layers (up-sampling layers) to restore point features from a down-sampled point cloud to its original form. This layer takes the global features from the previous layer and updates the features of each point by considering its local neighborhood.

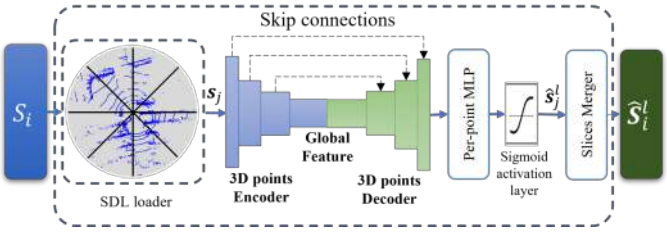


Fig. 5: The LTS-NET is a point-wise network designed to infer spatiotemporal stability of 3D LiDAR frames. The input frames $\mathcal{S}_i$ are processed by SDL, which divides the frame into a predefined number of slices, $\mathcal{N}$, as expressed in Eq. 5. The SDL passes each slice, $\mathbf{s}_j$, to the network for processing. The slices are then merged at the output of the network to form the original frame. This allows the network to infer the spatiotemporal stability of each point in the frame accurately.

The final output layer uses a Sigmoid function to bound the output values between 0 and 1. To supervise the training process, we used the Root Mean Square Error as a cost function, given by:

$$\mathcal{L} = (\frac{1}{N}\sum_{i=1}^{N}(l_i - \hat{l}_i)^2)^{\frac{1}{2}}, \tag{4}$$

here N is the number of points that goes into the first layer of LTS-NET multiplied by the training batch size, l_i and $\hat{l}_i$ are the true and predicted stability values of the i-th point.

LiDAR frames data loader: to process the LiDAR frame effectively by the network, we introduce the Slices Data Loader (SDL). The SDL divides the LiDAR frame into $\mathcal{N}$ slices, with a slice angle of $\psi = 2\pi/\mathcal{N}$, where $\mathcal{N} \geq 1$. The slice points are found by using the azimuthal angle θ of the spherical coordinates of the LiDAR frame, which is calculated as $\theta = arctan2(\mathbf{y}, \mathbf{x})$, where $\mathbf{y}$ and $\mathbf{x}$ are the Cartesian coordinates of the frame points. The slice points $\mathbf{s}_j$ are then obtained using the following equation:

$$\mathbf{s}_j = (\mathcal{S} | (\theta \geq \phi(j)) \ \cap \ (\theta < \Phi(j))), \tag{5}$$

where $j \in \mathcal{N}$ is the slice number, $\mathcal{S}$ is the LiDAR frame and $\phi(j) = j \times \psi$ and $\Phi(j) = (j+1) \times \psi$. Using the SDL enables the network to effectively learn regression by capturing most frame features. It prevents sub-sampling issues in the initial layer and avoids enlarging model layers, making it less resource-intensive and easier to train on mobile platforms.

IV. Experiments

A. Dataset

To demonstrate the effectiveness of our system in learning stable objects and achieving robust localization in a seasonally changing environment, we conducted experiments using the Bacchus Long-Term (BLT) dataset [5]. This dataset was collected in a semi-structured agricultural setting, specifically a vineyard, and includes data captured over a period of several months. During this time, the robot traversed a set of paths that intersected with each other, making it an ideal testbed for evaluating long-term localization algorithms. The traversed paths in the dataset are presented as a topological map in Fig. 6. The environment in the dataset includes a variety of objects that change at different rates, including static objects like buildings and other structures, slow dynamic objects such as vegetation, and fast dynamic objects like people who accompanied the robot during the data collection sessions.

The mobile robot in the BLT dataset is equipped with various sensors, including an RTK-GPS and an OS1-16 LiDAR sensor. The full list of sensors can be found in [5]. During the experiments, the RTK-GPS was used as the ground truth signal for the robot's pose, while data from the OS1-16 LiDAR sensor was used to test algorithms. The test path is $B \rightarrow G \rightarrow H \rightarrow C$ as shown in Fig. 6, which have a total length of 105 m. The experiments were conducted on six different sessions in 2022: *April 6th*, *April 20th*, *June 1st*, *June 8th*, *June 29th*, and *July 13th*. We use *April 6th* data to create the base map for localizing in the subsequent session.

Fig. 6: The topological map for the traversed paths in BLT-Dataset. The traversed path that we used for our experiments is highlighted in red.

The data labelling is performed as explained in Sec. III-A, then the labelled data is used to train a deep regression model to be used later to infer the stable points of the data (LiDAR frames) of the upcoming session. For instance, the labelled data of *June 1st* is labelled w.r.t previous two sessions that are *April 6th* and *April 20th*, then we train the network, in which we call LTS-JUNE-1 (based on the training data), to infer the long term stable objects of the next session *June 8th* LiDAR frames.

To evaluate the performance of the network, we used two metrics: root mean squared error loss ($\mathcal{L}$) and the coefficient of determination, also known as R-Squared ($\mathcal{R}^2$), which is often considered a more meaningful metric for evaluating regression models [28]:

$$\mathcal{R}^2 = \frac{\sum_{i=1}^{N}(\hat{l}_i - \bar{l})^2}{\sum_{i=1}^{N}(l_i - \bar{l})^2}, \tag{6}$$

here N represents the number of labels, l_i is the label value for the i-th point, $\hat{l}_i$ is the predicted label value from the network, and $\bar{l}$ is the mean of the ground truth labels.

LTS-NET training: To train the network, we split the LiDAR frames of the sessions into 80% training and 20% validation data. The training was conducted on a workstation with an Intel Core i7-6850K CPU, 64GB of RAM, and two

NVidia GTX 1080ti GPUs with 12GB of RAM each. The model was implemented using the PyTorch framework. We used an initial learning rate of 0.001, momentum of 0.9, and trained the model for 90 epochs for each network. The training time for the models was approximately 40 hours. The training results are summarized in Tab. I.

TABLE I: Networks training performance on different sessions. The metrics represent the average value.

Network	Training session	$\mathcal{L}$	$\mathcal{R}^2$
LTS-APRIL-20	*April 20th*	0.124	0.784
LTS-JUNE-1	*June 1st*	0.127	0.897
LTS-JUNE-8	*June 8th*	0.131	0.888
LTS-JUNE-29	*June 29th*	0.114	0.916

LTS-NET inference: Table II summarizes the network's inference performance on different sessions. As shown, LTS-APRIL-20 network had a poor performance with a negative R-Squared value, indicating that it was unable to accurately explain the data from *June 1st*. This may be due to the significant time gap between *April 20th* and *June 1st*, which resulted in a significant change in the appearance of the environment due to plant growth. However, the performance improved for the subsequent sessions as the appearance of the environment remained similar.

TABLE II: Networks inference performance on different sessions. The metrics represent the average value.

Network	Inferred session	$\mathcal{L}$	$\mathcal{R}^2$
LTS-APRIL-20	*June 1st*	0.427	-0.200
LTS-JUNE-1	*June 8th*	0.238	0.618
LTS-JUNE-8	*June 29th*	0.231	0.642
LTS-JUNE-29	*July 13th*	0.221	0.680

B. Evaluating localization performance

To evaluate the effect of filtering dynamics from LiDAR scans on long-term localization performance, we compare the localization performance of the filtered scans with that of the raw scans. All localization experiments were performed on an off-ground static map from *April 6th* as a base/reference map. The off-ground map used because the ground plane does not provide unique features for achieving long-term localization and can potentially lead to incorrect convergence of the localization package due to its size compared to the rest of the map. Therefore, we filter it out of the base map.

For localization, we use the HDL localization package [1], which is a 3D localizer based on the Normal Distribution Transform (NDT) [29] method. The filtered scans were obtained by thresholding the network predictions. During our experiments, we found that a threshold of $\epsilon_1 = 0.9$ is sufficient to filter out slow and fast dynamic points.

To evaluate the performance of the localizer, we use the Mean and the Root Mean Square Error (RMSE) of the Absolute Trajectory Error (ATE), which measures the difference between a device's true and estimated trajectories in a global coordinate system [30]. The true (ground truth) position in our setup is the pose from RTK GPS. Table III summarize the localization performance.

The metrics presented in Table III provide information on the accuracy of the system, but do not reflect its robustness. To evaluate the robustness of the system, we use the empirical Cumulative Distribution Function (CDF). This metric is commonly used to assess the registration accuracy between a reference scan and an input scan, as explained in [31]. However, it can also be used to assess the robustness of a localization system, as demonstrated in [32]. The CDF plots for the translational and rotational errors are presented in Fig. 7.

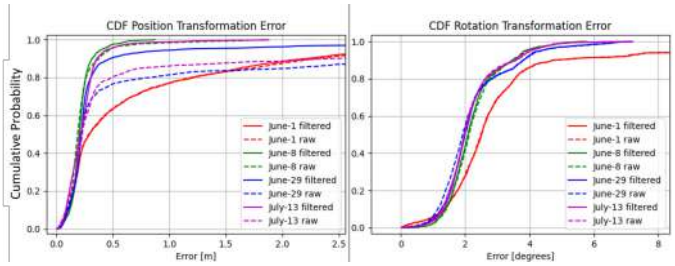


Fig. 7: The CDF plot is comparing the localization translation and rotational error of raw scans to the localization translation error of filtered scans for different sessions.

C. Results discussion

The localization performance for the *April 20th* session was evaluated using raw scans only as it was the second session, and there were not enough observations to train a network to infer *April 20th* scans. The localization performance was still the best among all sessions because the appearance of the environment did not change much between *April 20th* and *April 6th*. For the reset of the sessions the filtered scans resulted in improved robustness and performance for the localizer, as presented in Tab. III and in figure 7. However, on *June 1st* the localization performance was similar for both raw and filtered scans due to the failure of the LTS-APRIL-20 network to infer dynamic points in the LiDAR scans of *June 1st*, with identical CDF plots for both scans types. We attribute this to the lack of examples of dynamic objects/structures in the April data, as the vineyard was only in the early stages of vegetation at that time.

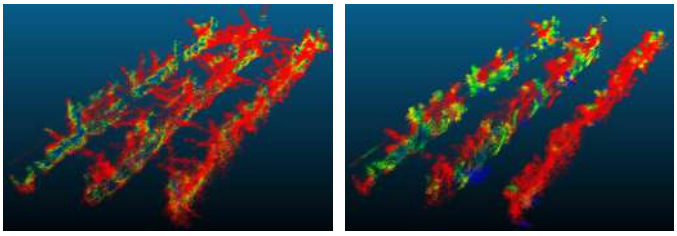
For the *June 8th* localization session, the LTS-JUNE-1 model demonstrated slight improvement in localization robustness and accuracy, with a mean error of 0.216 m for the entire estimated trajectory compared to 0.228 m for the raw scans. The raw scans showed good performance, which can be explained by a trimming process that occurred between *June 1st* and *June 8th*, resulting in fewer dynamic objects in the environment as shown in Figure 8. On *June 29th*, the filtered scans demonstrated superior performance and robustness compared to the raw scans. For example, the mean

[1]https://github.com/koide3/hdl_localization

TABLE III: Localization performance for robot position (pos) estimation of the raw scans compared the filtered scans. The metrics used are RMSE, Mean and std of the ATE, the units are all in m.

Session	Raw		Network	Filtered	
	ATE_{pos}^{RMSE}	ATE_{pos}^{Mean} (std)		ATE_{pos}^{RMSE}	ATE_{pos}^{Mean} (std)
April 20th	0.186	0.179 (0.048)	—	—	—
June 1st	1.206	0.743 (0.951)	LTS-APRIL-20	1.176	0.731 (0.921)
June 8th	0.300	0.228 (0.195)	LTS-JUNE-1	0.239	0.216 (0.103)
June 29th	2.994	1.250 (2.721)	LTS-JUNE-8	0.890	0.395 (0.795)
July 13th	2.199	0.903 (2.005)	LTS-JUNE-29	0.295	0.234 (0.179)

localization error for the entire trajectory was less than 0.4 m for the filtered scans, while it was 1.25 m for the raw scans. Similar results were observed on *July 13th*, indicating that the LTS-NET was able to successfully identify and filter in the long-term stable objects.



(a) Vine rows for *June 1st* (b) Vine rows for *June-8th*

Fig. 8: Comparison between the orchard rows in *June 1st* and *June-8th* indicating that a pruning process occurred between the two sessions, which resulted in fewer dynamic elements.

Despite the fact that the translational error was smaller for the filtered scans, the rotation error was similar for both types of scans, as illustrated in Fig. 7. This can be attributed to the robot's movement pattern, which consisted of traversing in straight lines within the rows of vines and only performing rotations at the end of the rows where stable structures were more pronounced. This enabled the localizer to robustly estimate the robot's heading angle for both types of scans.

D. Evaluating LTS-NET inference in a new environment

The motivation behind this experiment is to evaluate the performance of LTS-NET which has been trained on the temporal stack data from the BLT dataset in a completely new vineyard environment. The aim is to demonstrate that once the LTS-NET has learned long-term spatiotemporal stability, it can be applied directly in a new environment that lacks prior observations. To this end, data were collected in an initial-state vineyard (as shown in Fig. 9), over two sessions, primarily to generate labels for evaluating the LTS-NET's inference performance. The results show that the network exhibits an acceptable evaluation loss ($\mathcal{L}$ = 0.245) and coefficient of determination ($\mathcal{R}^2$ = 0.216) in this new environment. This suggests that the model is capable of providing plausible estimates of object stability, as demonstrated in Fig. 9-d, where the model correctly identifies the human as a dynamic object and the poles in Fig. 9-e as static/stable objects.

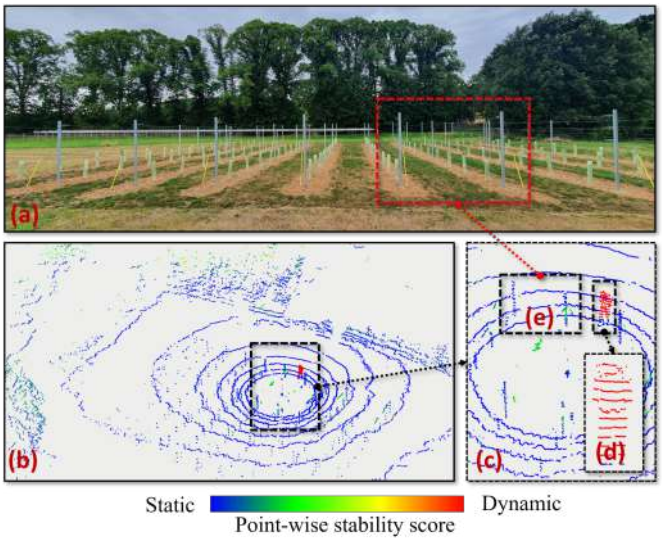


Fig. 9: Using the regression model learned from the BLT dataset in a new vineyard setting. (a) The image of the new field. (b) Displays a LiDAR frame with its predicted points stability labels. (c) A zoomed-in photo highlights some of the inferred features. (d) The individual accompanying the robot is considered dynamic based on inference. (e) The poles in the environment are considered stable objects through inference.

V. Conclusion

In this paper, we have proposed a novel spatiotemporal data-driven point-wise filter for learning long-term stability landmarks for robust localization in a continuously changing environment. The system utilizes an unsupervised labelling algorithm for 3D LiDAR scans to generate spatiotemporal point-wise stability scores based on multiple observations of the environment, and a point-wise regression network called LTS-NET to infer the stability of objects from 3D LiDAR frames. Through experimental evaluation, we have demonstrated the effectiveness of our approach in filtering dynamic elements from the scans and achieving robust, long-term localization performance. Furthermore, LTS-NET showed good performance when inferring object stability on LiDAR data from a completely new environment.

While the system demonstrated the ability to learn long-term stable features and use them to achieve robust localization over time, there are still some limitations that could potentially compromise the overall system. These limitations

can be summarized as follows: (1) Our unsupervised labelling algorithm relies on the accuracy of ICP map alignment; thus if this step fails, incorrect features may be associated with the points, which will impact the learning and filtering process. This issue could be addressed by introducing some manual intervention by a human operator. (2) The LiDAR resolution is another factor that affects the robustness of the system. A higher resolution allows the system to extract and learn more stable features, resulting in increased robustness. (3) The current implementation of LTS-NET has an inference time of approximately 2 frames per second.

As for future work, we aim to optimize the LTS-NET network architecture and code to enable real-time performance. In addition to that, further qualitative analysis is required to verify the transferability of the model to different domains, particularly with regard to seasonal changes, as the necessary data is currently unavailable.

ACKNOWLEDGEMENTS

This work has been supported by the European Commission as part of H2020 under grant number 871704 (BACCHUS).

REFERENCES

[1] T. Duckett, S. Pearson, S. Blackmore, B. Grieve, W.-H. Chen, G. Cielniak, J. Cleaversmith, J. Dai, S. Davis, C. Fox, *et al.*, "Agricultural robotics: The future of robotic agriculture," 2018.

[2] I. Hroob, R. Polvara, S. Molina, G. Cielniak, and M. Hanheide, "Benchmark of visual and 3d lidar slam systems in simulation environment for vineyards," in *Annual Conference Towards Autonomous Robotic Systems*. Springer, 2021, pp. 168–177.

[3] C. Toft, W. Maddern, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, T. Pajdla, *et al.*, "Long-term visual localization revisited," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[4] F. Boniardi, T. Caselitz, R. Kümmerle, and W. Burgard, "Robust lidar-based localization in architectural floor plans," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3318–3324.

[5] R. Polvara, S. Molina, I. Hroob, A. Papadimitriou, T. Konstantinos, D. Giakoumis, S. Likothanassis, D. Tzovaras, G. Cielniak, and M. Hanheide, "Blt dataset: acquisition of the agricultural bacchus long-term dataset with automated robot deployment," *Journal of Field Robotics, Agricultural Robots for Ag 4.0*, 2023, under Review.

[6] P. Mühlfellner, M. Bürki, M. Bosse, W. Derendarz, R. Philippsen, and P. Furgale, "Summary maps for lifelong visual localization," *Journal of Field Robotics*, vol. 33, no. 5, pp. 561–590, 2016.

[7] A. S. Aguiar, F. N. dos Santos, J. B. Cunha, H. Sobreira, and A. J. Sousa, "Localization and mapping for robots in agriculture and forestry: A survey," *Robotics*, vol. 9, no. 4, p. 97, 2020.

[8] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, 2022.

[9] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, "Lifelong localization in changing environments," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1662–1678, 2013.

[10] R. W. Wolcott and R. M. Eustice, "Robust lidar localization using multiresolution gaussian mixture maps for autonomous driving," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 292–319, 2017.

[11] N. Steinke, C.-N. Ritter, D. Goehring, and R. Rojas, "Robust lidar feature localization for autonomous vehicles using geometric fingerprinting on open datasets," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2761–2767, 2021.

[12] C. Linegar, W. Churchill, and P. Newman, "Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation," in *2015 IEEE International conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 90–97.

[13] M. Bürki, M. Dymczyk, I. Gilitschenski, C. Cadena, R. Siegwart, and J. Nieto, "Map management for efficient long-term visual localization in outdoor environments," in *2018 IEEE Intelligent vehicles symposium (IV)*. IEEE, 2018, pp. 682–688.

[14] T. Krajník, J. P. Fentanes, J. M. Santos, and T. Duckett, "Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 964–977, 2017.

[15] L. Wang, W. Chen, and J. Wang, "Long-term localization with time series map prediction for mobile robots in dynamic environments," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1–7.

[16] B. Peng, H. Xie, and W. Chen, "Roll: Long-term robust lidar-based localization with temporary mapping in changing environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 2841–2847.

[17] W. Wang, B. Wang, P. Zhao, C. Chen, R. Clark, B. Yang, A. Markham, and N. Trigoni, "Pointloc: Deep pose regressor for lidar point cloud localization," *IEEE Sensors Journal*, vol. 22, no. 1, pp. 959–968, 2021.

[18] H. Yin, R. Chen, Y. Wang, and R. Xiong, "Rall: end-to-end radar localization on lidar map using differentiable measurement model," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[19] V. Peretroukhin and J. Kelly, "Dpc-net: Deep pose correction for visual localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2424–2431, 2017.

[20] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, "Dynamic-slam: Semantic monocular visual localization and mapping based on deep learning in dynamic environment," *Robotics and Autonomous Systems*, vol. 117, pp. 1–16, 2019.

[21] J. Mendes, F. N. Dos Santos, N. Ferraz, P. Couto, and R. Morais, "Vine trunk detector for a reliable robot localization system," in *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2016, pp. 1–6.

[22] A. Papadimitriou, I. Kleitsiotis, I. Kostavelis, I. Mariolis, D. Giakoumis, S. Likothanassis, and D. Tzovaras, "Loop closure detection and slam in vineyards with deep semantic cues," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2251–2258.

[23] K. Wong, S. Wang, M. Ren, M. Liang, and R. Urtasun, "Identifying unknown instances for autonomous driving," in *Conference on Robot Learning*. PMLR, 2020, pp. 384–393.

[24] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[25] W. Zhang, J. Qi, P. Wan, H. Wang, D. Xie, X. Wang, and G. Yan, "An easy-to-use airborne lidar data filtering method based on cloth simulation," *Remote sensing*, vol. 8, no. 6, p. 501, 2016.

[26] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 2001, pp. 145–152.

[27] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[28] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation," *PeerJ Computer Science*, vol. 7, p. e623, 2021.

[29] M. Magnusson, "The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection," Ph.D. dissertation, Örebro universitet, 2009.

[30] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7244–7251.

[31] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[32] P. Barsocchi, S. Chessa, A. Micheli, and C. Gallicchio, "Forecast-driven enhancement of received signal strength (rss)-based localization systems," *ISPRS International Journal of Geo-Information*, vol. 2, no. 4, pp. 978–995, 2013.

GAFAR: Graph-Attention Feature-Augmentation for Registration A Fast and Light-weight Point Set Registration Algorithm

Ludwig Mohr*, Ismail Geles and Friedrich Fraundorfer†
Institute of Computer Graphics and Vision, Graz University of Technology
Inffeldgasse 16, 8010 Graz, Austria
Email: *ludwig.mohr@icg.tugraz.at, †fraundorfer@icg.tugraz.at

***Abstract*—Rigid registration of point clouds is a fundamental problem in computer vision with many applications from 3D scene reconstruction to geometry capture and robotics. If a suitable initial registration is available, conventional methods like ICP and its many variants can provide adequate solutions. In absence of a suitable initialization and in the presence of a high outlier rate or in the case of small overlap though the task of rigid registration still presents great challenges. The advent of deep learning in computer vision has brought new drive to research on this topic, since it provides the possibility to learn expressive feature-representations and provide one-shot estimates instead of depending on time-consuming iterations of conventional robust methods. Yet, the rotation and permutation invariant nature of point clouds poses its own challenges to deep learning, resulting in loss of performance and low generalization capability due to sensitivity to outliers and characteristics of 3D scans not present during network training.**

In this work, we present a novel fast and light-weight network architecture using the attention mechanism to augment point descriptors at inference time to optimally suit the registration task of the specific point clouds it is presented with. Employing a fully-connected graph both within and between point clouds lets the network reason about the importance and reliability of points for registration, making our approach robust to outliers, low overlap and unseen data. We test the performance of our registration algorithm on different registration and generalization tasks and provide information on runtime and resource consumption. The code and trained weights are available at `https://github.com/mordecaimalignatius/GAFAR/`.

I. Introduction

Rigid registration of point clouds is the task of simultaneously inferring both pose and correspondences between two sets of points [1]. As soon as either pose or correspondences are known, estimation of the respective other is straight forward, yet doing both simultaneously is posing challenges in computer vision and robotics. Its importance in tasks such as pose estimation [2], [3], map-building and SLAM [4], [5] as well as localization tasks geared towards autonomous driving [6] fuel the research interest in registration algorithms.

ICP and its many variants [7], [8], while able to provide exceptional results for good initializations, tend to get stuck in local minima if the initialization is insufficient, in the presence of high outlier rates, or in cases with low overlap. Attempts to resolve this range from methods using branch-and-bound to infer a globally optimal solution [9], methods based on feature matching between key-points followed by robust matching strategies [10], [11] and in recent years deep neural networks for learning feature descriptors and matching [12], [13], [14], [15]. Yet both, branch-and-bound as well as robust matching, suffer from speed and accuracy issues in real-life application due to the high number of iterations necessary in cases of high outlier ratios. Deep-learning based methods usually fare better with regard to outliers, yet still struggle due to the contradiction between low distinctiveness of local point-features caused by topological similarities and low match recall of global features in low-overlap cases. A further drawback of algorithms using deep neural networks often is their high requirements concerning compute resources, limiting their use in mobile applications.

979-8-3503-0704-7/23/$31.00 © 2023 IEEE

Fig. 1. Registration example on high quality real world scans captured with a handheld 3D scanner (left) and examples of the (meshed and textured) object scans available in the custom dataset used for testing generalization ability (right).

To tackle these challenges, we propose GAFAR: Graph-Attention Feature-Augmentation for Registration, which employs deep-learning techniques not only for extraction of meaningful local features from point sets, but also for learning an adaptive augmentation network for online transformation of local features for robust matching. We achieve this by exploiting structural information from between point sets as well as from within a single one thorough an architecture of interleaved self- and cross-attention layers [16], [17]. While achieving state-of-the-art registration performance, our method is light-weight and fast.

We demonstrate this in a series of experiments, testing not only registration performance on the dataset used for training, but also robustness and generalization ability in two further

experiments on vastly different datasets of real world scans, one captured with a handheld 3D scanner producing precise scans, the other being the Kitti Odometry Dataset [18] showing street scenes captured by a LiDAR scanner. Furthermore, we provide insight into runtime and resource needs. The main contributions of our method are:

- We demonstrate the use of transformer networks and the attention mechanism to build a fast and light-weight, yet accurate registration algorithm.
- We present an online feature augmentation strategy in registration which proves to be superior in terms of robustness to partial overlap and geometries not seen during training.
- We show how certain design-choices enable us to estimate the registration success without knowledge of the true transformation, enabling its use in applications that require fail-safes.
- We demonstrate state-of-the art performance and superior generalization capability in a light-weight package.

II. Related Work

One of the oldest, yet still relevant methods for registration of point clouds is ICP [19]. Starting from an initial alignment, ICP iteratively updates the registration parameters by establishing point correspondences using Euclidean distance, rejecting far away point pairs. Due to this design it is prone to get stuck in local minima, the final registration accuracy heavily depends on the initialization. Many variants have been proposed over the years [7] to mitigate these issues, yet the dependence on the initialization has remained.

Several registration algorithms trying to solve the dependence on initialization have been proposed [9], [20], alongside of handcrafted feature descriptors trying to capture local geometry of point clouds in a meaningful way, such as PPF [21] and FPFH [22], among others [23]. Yet, they never managed to reach the performance and robustness of their 2D counterparts.

Recent advances in deep-learning extend deep neural networks to 3D point clouds and have resulted in methods for learned local feature descriptors like PointNet [24], [25], FCGF [12], Graphite [26] and DGCNN [27], learned filtering of putative point matches [28] and complete learned registration pipelines. 3DSmoothNet [29] extracts a local reference frame and voxelizes the point cloud around key points, yet reference frame estimation is susceptible to outliers, voxelization tends to loose information due to spatial discretization. PointNetLK [14] estimates registration parameters to match the deep representations from PointNet of complete point clouds, DCP [15] uses DGCNN to extract point features and the attention mechanism [16] to predict soft correspondences, restricting their application to registration of point clouds with high overlap. Research into Pillar-Networks [6], [30] is driven mainly by automotive applications for processing of LiDAR point clouds from mobile mapping systems, assuming the input point clouds to share a common z=up orientation. They extract cylindrical point pillars along the z-axis around key point locations for further processing, and are therefor not applicable to general registration problems or when the assumption of z-axis alignment can not be guaranteed. Keypoint based methods like [31] aim at detecting repeatable keypoints across scans, and registering them using powerful descriptors. In contrast [32] uses a detection-free approach with a local-to-global detection strategy using superpoints. IDAM [33] tackles inaccuracies arising from inner product norms for feature matching with an iterative distance-aware similarity formulation. DeepGMR [34] recovers registration parameters from Gaussian Mixture Models, parameterized using pose-invariant correspondences. RPM-Net [35] predicts annealing parameters and predicts correspondences with annealing in feature matching and the Sinkhorn Algorithm [36] as solver for linear assignment, predicting soft correspondences. RGM [37] explicitly builds and matches graphs within point clouds to resolve ambiguity issues between locally similar patches and predicts hard correspondences using the Hungarian Algorithm.

In contrast to [37], we use graph matching for feature augmentation before matching, but do not match graphs extracted from point clouds explicitly. Similarity between the internal point cloud structures is handled by our method implicitly using cross-attention modules. Our method predicts hard correspondences by thresholding of the assignment matrix after running sinkhorn iterations, interpreting the correspondence estimation as optimal transport problem of the feature correlation matrix.

III. Problem Formulation

Rigid registration of two 3D point sets is the task of finding a transformation consisting of a rotation matrix $\mathbf{R} \in \mathbf{SO}^3$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$ aligning input point set $\mathcal{P}_S = \{p_i \in \mathbb{R}^3 | i = 1, ..., M\}$ to the reference point set $\mathcal{P}_R = \{p_j \in \mathbb{R}^3 | j = 1, ..., N\}$. Here M and N denote the respective sizes of the point sets.

The underlying assumption is, that both point sets are sampled on the same surface or the same object and share at least some common support (i.e., the physical location where the object has been sampled does actually overlap). In the most general case, point sets $\mathcal{P}_S$ and $\mathcal{P}_R$ may not have any true correspondences between them, may suffer from outliers and additive noise and they may only share parts of their support, resulting in only partial overlap.

Given a set of corresponding points between two point sets, the rigid transformation aligning both sets can be recovered using SVD. This approach relaxes the task of estimating a rigid transformation to that of finding pairs of corresponding points between both sets. Since the transformation obtained using SVD aligns the point pairs in a least-squares sense, this formulation directly lends itself to the case where no exact matches exist.

Hence, the task of rigid point set registration can be formulated mathematically as:

$$\mathbf{C}^* = \underset{\mathbf{C}}{\operatorname{argmin}} \left(\sum_{j}^{N} \sum_{i}^{M} c_{i,j} \| \mathbf{R}_C p_i + \mathbf{t}_C - p_j \|^2 \right), \quad (1)$$

where $\mathbf{C} \in \{0,1\}^{M,N}$ is a permutation matrix subject to row and column constraints $\sum_i^M C_i = \mathbf{1}^N$ and $\sum_j^N C_j = \mathbf{1}^M$, associating the points between both point sets. The transformation parameters $\mathbf{R}_C$ and $\mathbf{t}_C$ refer to those recovered by SVD using the point pairs designated by permutation matrix $\mathbf{C}$. To handle the case of partial overlap, the permutation matrix is augmented by a row and column to $C \in \{0,1\}^{M+1,N+1}$, while relaxing the constraints on rows and columns of $\mathbf{C}$ to

$$\sum_i^M C_i \leq \mathbf{1}^N, \qquad \sum_j^N C_j \leq \mathbf{1}^M. \tag{2}$$

In practice, this formulation can be solved by augmenting an initial full point feature correlation matrix with an additional row and column and solving the relaxed optimization problem as the optimal transport problem [38], [39], using the Sinkhorn Algorithm as differentiable implementation of the linear assignment problem [36], [17].

IV. The Making of GAFAR

The key idea behind our network architecture is to adapt initial local per-point feature descriptors $\mathcal{F}_S$ of a source point set $\mathcal{P}_S$ for correspondence matching in an online fashion by injecting information of the reference point set $\mathcal{P}_R$. The reasoning behind this is, that for successful point matching neither only local geometric structure (which may be repetitive or non-distinctive) nor fully-global information (which in case of partial overlap may encode information of areas which are not shared) is sufficient. The relevant information for successful point matching lies solely within the topology of the overlapping area as well as the relative position of points within this area. Our architecture takes two point sets $\mathcal{P}_S$ and $\mathcal{P}_R$, represented as point locations in Euclidean coordinates together with their respective point normals, as input. Internally, the network architecture consists of a feature head generating per-point features for both point sets independently, as well as an augmentation stage inspired by [17], consisting of interleaved self- and cross-attention layers. This allows the network to reason jointly over both sets of feature descriptors, adapting them iteratively into representations optimally suited for finding high-quality correspondences between those two specific point sets. Matching is done by calculating the dot-product similarity between all possible pairings of the resulting feature descriptors $\hat{\mathcal{F}}_S$ and $\hat{\mathcal{F}}_R$, relaxing the match matrix by adding a slack row and slack column and running the Sinkhorn Algorithm a predefined number of iterations, as in [17], [35]. The network weights are shared between the two branches processing $\mathcal{P}_S$ and $\mathcal{P}_R$, turning the architecture into a fully-siamese network [40]. Figure 2a depicts an overview of the architecture, the different building blocks are explained in greater detail in the following subsections.

A. Local Feature Descriptor Head

Our feature head, depicted in Figure 2b, consists of two main building blocks, a local feature encoder with a neighbourhood size $\mathcal{N}$ and a point-wise location encoder Multi-Layer Perceptron (MLP). Both take point locations within the unit-circle and their respective normal vectors as input. As point-feature network we employ an architecture derived from DGCNN [27], extended by an MLP functioning as a bottleneck to reduce the feature dimensionality to a more suitable size. A basic layer of this architecture embeds the lower-dimensional representation into a higher dimensional local representation with a nonlinear transformation by applying a MLP on point patches consisting of the $\mathcal{N}$ nearest neighbours of each point p_i, followed by max-pooling over the patch and normalization. Information is aggregated via multiple layers and concatenation until a high-dimensional internal representation $\mathcal{F}_I \in \mathbb{R}^d$ of the local point neighbourhood is reached. Our point-wise location encoder is implemented as a pure point-wise MLP, for each point p in point set $\mathcal{P}$ embedding its position in Euclidean space into a high-dimensional feature space, again of size $\mathbb{R}^d$. The output of both, the feature encoder and the position encoder, are then concatenated and projected back to $\mathbb{R}^d$ by a small point-wise MLP.

B. Graph-Attention Feature-Augmentation Network

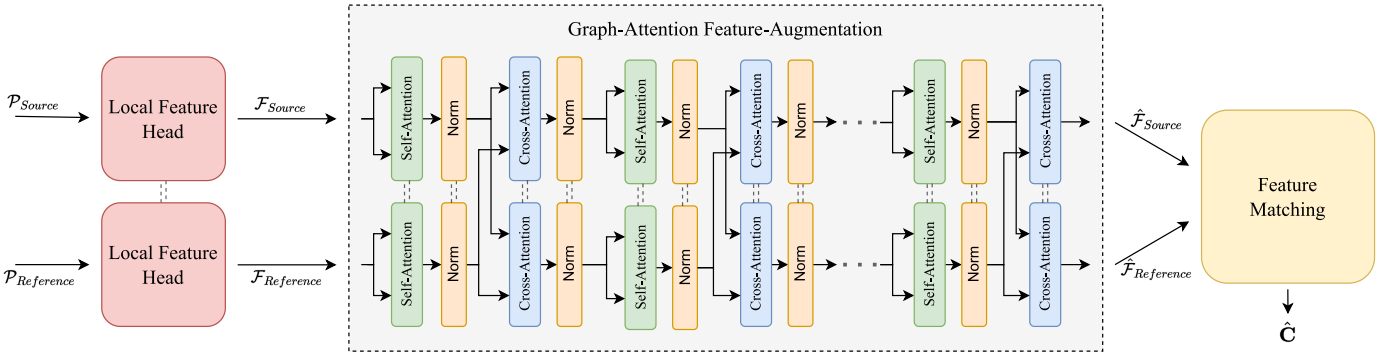
The purpose of the graph attention network for feature augmentation is to optimize the feature representations $\mathcal{F}_S$ of the input point set $\mathcal{P}_S$ at inference time for correspondence search by infusing knowledge of the reference point set $\mathcal{P}_R$, and vice versa. To this end, we build the feature augmentation sub-network as a stack of alternating self- and cross-attention layers, interleaved with normalization layers. The architecture of the attention layers is depicted in Figure 2c, implementing a residual block with message passing for feature update. We set the feature-augmentation network up as a stack of fully-connected graph-attention layers, thereby letting the network learn which connections are relevant for the current point feature from all possible connections and to only attend to those via Multi-Head Softmax-Attention. This allows to embed information of the relevant topology from both within and between point-sets in an iterative fashion into the feature descriptors, resulting in two sets $\hat{\mathcal{F}}_S : \{f_i \in \mathbb{R}^d, i = 1, ..., M\}$ and $\hat{\mathcal{F}}_R : \{f_j \in \mathbb{R}^d, j = 1, ..., N\}$ of point features for matching.

C. Feature Matching

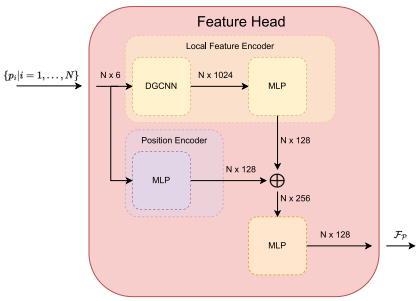
After feature augmentation, matching is done by calculating the similarity score matrix $\mathbf{S} \in \mathbb{R}^{M,N}$ between the point feature descriptors $\mathcal{F}_S^m$ and $\mathcal{F}_R^m$ of all possible point pairs $\mathbf{p}_{i,j} = \{p_i \in \mathcal{P}_S, p_j \in \mathcal{P}_R\}$ using dot-product similarity:

$$\mathbf{S} : s_{i,j} = \langle f_i, f_j \rangle . \tag{3}$$

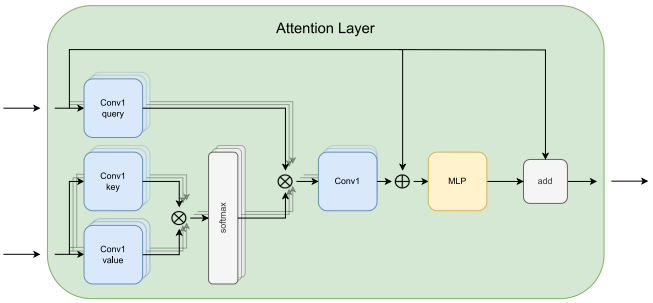
Since we are interested in finding point-correspondences, we interpret the optimization problem of equation (1) in terms of the optimal transport problem [39], using the similarity score $\mathbf{S}$ as its cost. We find an approximate solution $\mathbf{C}^*$ by adding a row and column of slack variables to $\mathbf{S}$ as detailed in equation 2 and applying a few iterations of the Sinkhorn-Algorithm as a differentiable approximation to the Hungarian Algorithm for the solution of optimal transport [36], [38], [41]. Finally, we threshold the resulting approximate permutation



(a) Overview of the network architecture of GAFAR.



(b) Feature head configuration.



(c) Detailed architecture of the attention layers.

Fig. 2. Architectural details of GAFAR. Figure (2a) shows an overview of the network architecture, sub-figures (2b) and (2c) the structure of the feature head and the attention layers, respectively. $\bigotimes$ denotes matrix multiplication, $\bigoplus$ concatenation.

matrix $\mathbf{C}^*$ by threshold $t_m \in [0, 1]$ and take mutual row- and column-wise maxima as point correspondences for calculation of the rigid transformation $\{\mathbf{R}, \mathbf{t}\}$ aligning the point sets using SVD.

D. Loss

As loss for network training we employ the binary cross entropy loss between the predicted permutation matrix $\mathbf{C}^*$ and the ground truth correspondence matrix $\mathcal{G}_{gt}$:

$$\mathcal{L}_{BCE} = -\sum_{i,j} g_{i,j} \log \hat{c}_{i,j} + (1 - g_{i,j}) \cdot \log(1 - \hat{c}_{i,j}). \quad (4)$$

V. Experiments

In order to evaluate the performance of our proposed registration method, we perform two experiments. The first experiment V-A tests the performance on synthetic data of ModelNet40 [42] for different settings of noise and overlap. The second experiment described in section V-B tests the generalization ability using LiDAR point clouds of the Kitti Odometry Benchmark [18] and custom high-quality real-world object scans, using only models trained on synthetic data in the experiment of section V-A.

Throughout the experiments, we have chosen the following parameters for our network: The feature dimension is chosen as $d = 128$, the number of layers and layer dimensions in the feature encoder of the feature head follows the parameterization of DGCNN [27] with a neighbourhood size of $\mathcal{N} = 20$. The location encoder is chosen as a 4 layer MLP with layer dimensions $[16, 32, 64, 128]$. Our feature-augmentation graph-attention network consists of 9 stacks of consecutive self- and cross-attention layers with 2 attention heads. For normalization, batch-norm is chosen throughout the network. The number of Sinkhorn-iterations is set to 10 for both, training and inference. We train the network on a single registration iteration per example, testing is done with a second iteration, feeding the source point cloud aligned by the result of the first iteration again through the network.

Model training usually converges after training for two days using AdamW optimizer with learning rate $1e^{-4}$ on a Nvidia GeForce RTX3090 (between 800 and 1000 epochs).

A. Experiments on ModelNet40

ModelNet40 consists of $12,311$ meshed CAD models in 40 object categories, spanning a vast array of scales from chairs to airplanes. Consistent with previous work, we use the pre-sampled point clouds provided by Shapenet [43], consisting of 2048 points per model to conduct the experiments. For easy comparison we follow the setup of [37] and perform the same experiments. All experiments with exemption of subsection V-A4 follow the official training and testing split, with an additional 80:20 split of the official training set for training and validation. The experiment described in subsection V-A4 uses the first 20 object classes of the training set for training, the first 20 object classes of the test set for validation and the remaining 20 classes for testing. The point clouds already come scaled to fit within the unit circle, therefor all measures related to point distance are given in a normalized scale. As in [37], we sample 1024 points at random from the point

clouds and apply random rotations within $[0°, 80°]$ around a random axis and random translations within $[-0.5, 0.5]$ in normalized units.

Registration performance in measured using the same metrics as [37], that is residual transformation errors of $\{\mathbf{R}, \mathbf{t}\}$ as mean isotropic errors (MIE) as proposed by [35], as well as clipped chamfer distance (CCD) between reference point cloud $\mathbf{Y}$ and transformed source point cloud $\hat{\mathbf{X}}$ after registration:

$$\begin{aligned} CCD(\hat{\mathbf{X}}, \mathbf{Y}) = &\sum_{\hat{x}_i \in \hat{\mathbf{X}}} \min(\min_{y_j \in \mathbf{Y}} (||\hat{x}_i - y_j||_2^2), r) \\ &+ \sum_{\hat{y}_j \in \mathbf{Y}} \min(\min_{\hat{x}_i \in \hat{\mathbf{X}}} (||\hat{x}_i - y_j||_2^2), r), \end{aligned} \tag{5}$$

with clip distance $r = 0.1$. Furthermore, we report registration recall (RR), defined as percentage of registration results with residual errors $\text{MAE}(\mathbf{R}) < 1°$ and $\text{MAE}(\mathbf{t}) < 0.1$. To keep consistent with previous research, we also state the residual transformation errors in terms of mean absolute errors (MAE) as proposed by [15], which is anisotropic. Errors related to rotations are given in degrees, errors related to distance are normalized to object size (since the data in ModelNet40 does not have a common scale and is normalized to the unit circle).

The design of our registration method provides us directly with information on the reliability and success of a matching attempt. Using the value of the matching score $s_{i,j}$ matching point p_i to point p_j as well as the number of found matches, we can reject invalid registrations. To this end, we provide results for matching thresholds $t_m = 0.5$ and rejecting registrations with less than 3 correspondences. Evaluation of registration errors is done on successful registrations only, stating the percentage of successful registrations in braces after the method name. Registration recall for our method is provided with respect to the full number of examples in the testing set, thereby making it directly comparable. In practical applications, failed registrations can easily be rectified by either performing batched registrations with different samplings for a single registration task or repeating the registration with a different subset of points in case of failure. Please note that the main competing methods do not allow any insight like this without knowledge of the underlying true registration, since RPMNet [35] works on soft-correspondences, RGM [37] only provides hard correspondences without associated score and returned in our experiments always more than 3 matches. Results of the comparing methods are reproduced from [37].

1) Full and clean data: The first experiment can be considered a baseline in registration performance, since the transformation has to be recovered from a full set of 1024 exact and noise-free correspondences and is mainly reproduced for completeness. From Table I we can see that basically all methods are able to almost perfectly register the point clouds with $\text{MAE}(\mathbf{R})$ below or around $1°$. Only ICP struggles in comparison.

2) Additive gaussian noise: In this experiment, source and reference point sets are sampled independently, so only a few perfect correspondences may exist. Additionally, we add gaussian noise sampled from $\mathcal{N}(0, 0.01)$ and clipped to the range $[-0.05, 0.05]$ to the point locations independently, thereby eliminating all perfect correspondences. Point correspondences and point normals are then re-established, following the procedure of [37], first finding mutual nearest neighbours and then adding remaining nearest neighbours, all within a maximum distance of 0.05 between corresponding points.

TABLE I
REGISTRATION PERFORMANCE ON CLEAN POINT CLOUDS.

method	MIE(R)	MIE(t)	MAE(R)	MAE(t)	CCD	RR
ICP [19]	6.4467	0.05446	3.079	0.02442	0.03009	74.19 %
FGR [20]	0.0099	0.00010	0.006	0.00005	0.00019	99.96 %
IDAM [33]	1.3536	0.02605	0.731	0.01244	0.04470	75.81 %
DeepGMR [34]	*0.0156*	*0.00002*	*0.001*	*0.00001*	*0.00003*	**100.00** %
RPMNet [35]	0.2464	0.00112	0.109	0.00050	0.00089	98.14 %
RGM [37]	**0.0103**	**<0.00001**	**<0.001**	**<0.00001**	**<0.00001**	**100.00** %
Ours (100.00%)	0.0150	0.00009	0.007	0.00004	0.00014	99.92 %

As can be expected, the performance degrades to a certain degree. The results listed in Table II show that the learning based methods still hold up rather well with $\text{MAE}(\mathbf{R})$ around or below $3°$. RPMNet, RGM as well as our method still achieve a RR of more than 90%. Interestingly, the performance of ICP does not degrade, showing its robustness to outliers.

TABLE II
REGISTRATION PERFORMANCE WITH ADDITIVE GAUSSIAN NOISE.

method	MIE(R)	MIE(t)	MAE(R)	MAE(t)	CCD	RR
ICP [19]	6.5030	0.04944	3.127	0.02256	0.05387	77.59 %
FGR [20]	10.0079	0.07080	5.405	0.03386	0.06918	30.75 %
IDAM [33]	3.4916	0.02915	1.818	0.01516	0.05436	49.59 %
DeepGMR [34]	2.2736	0.01498	1.178	0.00716	0.05029	56.32 %
RPMNet [35]	*0.5773*	*0.00532*	*0.305*	*0.00253*	*0.04257*	*96.68 %*
RGM [37]	**0.1496**	**0.00141**	**0.080**	**0.00069**	**0.04185**	**99.51** %
Ours (98.82%)	0.8560	0.00635	0.518	0.00296	0.04297	93.64 %

3) Registration of noisy, partially overlapping sets: In this experiment, in addition to additive gaussian noise, both source and reference point clouds are independently cropped along a random plane to 70% of their original size, resulting in variable overlap of at least 40%. This experimental setup corresponds closest to general real-world applications. From Table III we see that, with exception of RPMNet [35], RGM [37] and ours, the registration performance degrades beyond anything what can be deemed usable in any applications. Notably, the registration performance on recovered registrations of our method is the same as in the previous experiment with full overlap, albeit losing in successful registrations and in registration recall. Comparing the registration recall of 77.2% to the percentage of recovered registrations of 84.3%, we see the merit of our architecture and the ability to predict whether a registration attempt was successful.

4) Partial overlap of unseen object categories: The difference to the experiment outlined in section V-A3 is that now we only train on the first 20 object categories of ModelNet40, but evaluate on the remaining 20 categories. Thereby we can explore to what extent the learned registration networks are able to generalize to geometries not present in training. An

TABLE III
REGISTRATION PERFORMANCE WITH ONLY PARTIAL OVERLAP AND ADDITIVE GAUSSIAN NOISE.

method	MIE(R)	MIE(t)	MAE(R)	MAE(t)	CCD	RR [%]
ICP [19]	24.8777	0.26685	12.456	0.12465	0.11511	6.56 %
FGR [20]	42.4292	0.30214	23.185	0.14560	0.12118	5.23 %
IDAM [33]	16.9724	0.19209	8.905	0.09192	0.12393	0.81 %
DeepGMR [34]	70.9143	0.45705	43.683	0.22479	0.14401	0.08 %
RPMNet [35]	1.6985	0.01763	0.864	0.00834	0.08457	*80.59 %*
RGM [37]	*0.9298*	*0.00874*	*0.492*	*0.00414*	*0.08238*	**93.31** %
Ours (84.32%)	**0.8854**	**0.00721**	**0.484**	**0.00347**	**0.08119**	77.19 %

interesting fact evident in the results listed in Table IV is that the performance of all methods except RPM [37] does not decline much relative to the experiment done on known categories, whereas RPM almost doubles its residual errors. Although very powerful in establishing good correspondences, the neural network architecture in RPM seems to learn geometries by heart, hampering its generalization ability, whereas our method performs as strong as it did before, outperforming RGM in all measures. This again exemplifies the merit of feature augmentation at test time for optimal matching success. Furthermore we would like to point out that although our method is not able to successfully register all examples in the first attempt, using the match threshold t_m and the number of found matches, we can precisely predict unsuccessful attempts. In all experiments, RR is close to the number of valid examples within a margin of about 5%.

TABLE IV
REGISTRATION PERFORMANCE ON UNSEEN CATEGORIES, PARTIAL OVERLAP AND GAUSSIAN NOISE.

method	MIE(R)	MIE(t)	MAE(R)	MAE(t)	CCD	RR
ICP [19]	26.6447	0.27774	13.326	0.13033	0.11879	6.71 %
FGR [20]	41.9631	0.29106	23.950	0.14067	0.12370	5.13 %
IDAM [33]	19.3249	0.20729	10.158	0.10063	0.12921	0.95 %
DeepGMR [34]	71.0677	0.44632	44.363	0.22039	0.14728	0.24 %
RPMNet [35]	1.9826	0.02276	1.041	0.01067	0.08704	75.59 %
RGM [37]	*1.5457*	*0.01418*	*0.837*	*0.00674*	*0.08469*	*84.28 %*
Ours (89.10%)	**0.8695**	**0.00871**	**0.434**	**0.00432**	**0.08299**	**85.78** %

B. Generalization to real-world 3D scans

For real-world application, the ability of 3D registration methods to generalize to new and different geometries as well as capturing modalities is crucial. To this end, we compare the registration performance of the best performing methods trained on ModelNet40 as detailed in section V-A on two datasets, a custom dataset (publication is planned) as well as the the well known Kitti Dataset [18]. The custom dataset consist of objects scans of 10 objects taken with an Artec Leo [44] handheld 3D scanner, for each object up to 10 overlapping partial scans exist, with between 10.000 and 50.000 points each. Figure 1 shows a registration example of this dataset. Transformations are generated within the same constraints as in the experiments on ModelNet40. We report registration accuracy in terms of MIE($\mathbf{R}$), MIE($\mathbf{t}$) and registration recall. Since the objects in this dataset have a common scale, MIE($\mathbf{t}$) is reported in millimeters, registration recall is defined as percentage of registration results with residual errors MIE($\mathbf{R}$) $< 1^\circ$ and MIE($\mathbf{t}$) $< 5mm$. Again, the number of in brackets behind versions of our method states the respective percentage of *valid* registrations. For the experiments on Kitti, we follow the established praxis [32], [12], [31] of testing on sequences 8-10, testing registration performance of point cloud pairs at least 10m apart. As in [32], [12], [31], we use ground truth poses refined by ICP, MIE($\mathbf{t}$) is reported in meters, and registration recall is defined as percentage of registration results with residual errors MIE($\mathbf{R}$) $< 5^\circ$ and MIE($\mathbf{t}$) $< 2m$. Note that for fairness we applied an additional data normalization step for RPM-Net and RGM, scaling the data to fit into the unit circle for registration, thus making the input points span the same range as the training data of ModelNet40. From Table V we can see that our algorithm generalizes well to high quality 3D scans, the models trained on partial overlapping data outperform both RGM [37] and RPMNet [35] by a large margin in all metrics. For registration of large-scale outdoor scenes of Kitti, a domain-gap for all methods is noticeable. Nonetheless, our method still performs reasonably well given the circumstances, with registration recall of around 50% and mean errors of 3.1° and $3.5m$ for the best generalizing models trained with only partial overlap, again showing its robustness to different data modalities. Furthermore, the strong ability to predict which registrations were successful is visible from comparing the number of 51.1% valid registrations to the RR of 49.7% for the model trained on unseen categories. Again, we can observe that while a powerful registration method, RGM seems to overfit on the training modalities, being beaten even by RPM-Net trained for the experiment on noisy data and unseen categories, whereas our method is rather robust to changes in sampling, overlap and geometry. Interestingly, for both, RGM and RPM-Net, models trained on the harder cases of only partial overlap often lead to a decrease in generalization performance, whereas our methods ability to generalize to different data improves with the difficulty of the training task.

TABLE V
GENERALIZATION TO REAL WORLD OBJECTS SCANNED USING A HANDHELD 3D SCANNER, USING THE MODELS TRAINED ON MODELNET40 FROM THE EXPERIMENTS IN SECTION V-A. HERE, THE NAME IN THE COLUMN *experiment* REFERS TO THE EXPERIMENT IN WHICH THE METHOD WAS TRAINED, NO FURTHER DATA AUGMENTATION HAS BEEN DONE BESIDES RANDOM SUB-SAMPLING. FOR RR, THRESHOLDS ARE SET AS MIE($\mathbf{R}$) $< 1^\circ$ AND MIE($\mathbf{t}$) $< 5mm$.

method	experiment	MIE(R) [°]	MIE(t) [mm]	RR [%]
RPMNet [35]	clean	23.9	88.2	0.8 %
	noise	1.8	6.2	62.8 %
	unseen	4.4	14.9	65.0 %
RGM [37]	clean	6.1	22.8	10.5 %
	noise	3.3	12.3	32.5 %
	crop	5.8	24.8	25.4 %
	unseen	7.0	26.6	26.0 %
Ours	clean (100.0%)	6.0	22.4	5.0 %
	noise (97.6%)	1.2	5.0	58.6 %
	crop (99.1%)	*0.6*	*1.9*	*74.3 %*
	unseen (98.7%)	**0.6**	**1.7**	**76.7** %

TABLE VI
GENERALIZATION TO DATA FROM KITTI ODOMETRY BENCHMARK, AGAIN USING THE MODELS TRAINED ON MODELNET40 FROM THE EXPERIMENTS IN SECTION V-A. FOR RR, THRESHOLDS ARE SET AS MIE($\mathbf{R}$) $< 5°$ AND MIE($\mathbf{t}$) $< 2m$.

method	experiment	MIE(R) [°]	MIE(t) [m]	RR [%]
RPMNet [35]	clean	100.1	9.8	0.0 %
	noise	5.6	8.3	0.5 %
	unseen	4.7	7.5	0.7 %
RGM [37]	clean	6.5	9.4	0.0 %
	noise	6.0	8.1	0.2 %
	crop	6.7	8.4	0.7 %
	unseen	9.2	8.7	0.0 %
Ours	clean (100.0%)	7.2	9.6	0.0 %
	noise (88.8%)	14.4	10.3	1.6 %
	crop (59.1%)	*3.4*	*3.5*	*47.0* %
	unseen (51.1%)	**3.1**	**3.5**	**49.7** %

C. Resource Consumption and performance

Registration performance is not the only relevant criterion for the usability of an algorithm. Execution time as well as compute resource needs are limited especially in mobile applications and are therefor a further relevant measure in algorithm selection. To this end, we compare our algorithm in terms of complexity and resource needs to the two best competing methods. Model complexity is measured in the number of trainable parameters. Compute resource needs are given in GB of GPU memory use for batch sizes of 20, 5, and 1, as well as registration speed measured in registrations per second. We can see from Table VII, that our method is both more light-weight and faster while still providing competitive results.

TABLE VII
RESOURCE CONSUMPTION OF THE BEST PERFORMING METHODS ON A NVIDIA GEFORCE RTX3090. MEMORY USE IS PROVIDED FOR BATCH SIZES 20, 5, AND 1.

method	param [#]	mem@20	mem@5	mem@1	rate [#/s]
RPMNet [35]	$0.91e^6$	6.50 GB	3.2 GB	2.2 GB	45.9
RGM [37]	$25.0e^6$	7.20 GB	3.4 GB	2.4 GB	6.6
Ours	$4.4e^6$	4.47 GB	2.6 GB	2.2 GB	62.0

D. Ablation Study

In order to evaluate the benefit of different parts in our feature head, we test the following configurations. Networks are trained on the task of partial overlap, as in section V-A3 using the same random seed, with the following architectural differences:

- Location encoder: the feature head only uses the location encoder.
- Feature only: the feature head only uses the local point feature network.
- additive fusion: the MLP fusing position encoding and local point feature is replaced by a simple addition of feature vectors.
- MLP fusion: this is the full network architecture, consisting of the feature head with location encoder, point feature network and MLP for feature fusion.

Please note that the networks have not been trained to full convergence, since only a qualitative difference is required. For testing, the same modalities as for the experiments in section V-A have been employed. From the results in Table VIII we can see, that each additional structure improves the overall performance, the method works best if we let the network learn how to combine both feature vectors.

TABLE VIII
ABLATION STUDY TESTING THE INFLUENCE OF THE DIFFERENT PARTS OF OUR FEATURE HEAD. THE FULL HEAD WITH BOTH, LOCAL FEATURE ENCODER AND POSITION ENCODER FUSED BY AN SMALL MLP, PERFORMS BEST. NOTE THAT THE NEURAL NETWORKS WERE NOT TRAINED TO FULL CONVERGENCE IN THIS STUDY.

variant	MIE(R) [°]	MIE(t)	RR [%]
location encoder	2.90	0.021	72.5 %
feature encoder	1.99	0.016	*78.9* %
additive fusion	*1.76*	*0.014*	77.0 %
MLP fusion	**1.29**	**0.012**	**79.0** %

VI. CONCLUSION

In this paper, we presented GAFAR, a novel, light-weight algorithm for point set registration using an end-to-end learnable deep neural network for feature encoding and correspondence prediction. Its performance is competitive while being faster and less demanding on resources compared to other state-of-the-art methods, which makes it well suited for applications with constraints on compute resources, power consumption and runtime. Our method shows very high generalization capability to different data modalities and exhibits little overfit to geometry details of the training set. The strong performance for partial overlap, even for object classes not present in training, shows the merits of the cross-attention mechanism for feature augmentation. A further benefit of our method is its ability to provide an indication on the quality of predicted correspondences, thereby giving opportunity to tune between high registration accuracy and high recall as well as to reject failed or bad registrations without additional knowledge. In practice, failure cases can be remedied by either performing multiple registrations with different sub-sampling in parallel in a batched fashion, or by repeating the registration with a different sample in case of failure.

In the future, we plan to tackle the limitation to only small subsets of point clouds by applying the underlying architectural principles to the registration of large point sets directly, while still keeping with the paradigm of light-weight architecture and fast execution.

ACKNOWLEDGMENT

This work was supported by Land Steiermark within the research initiative “Digital Material Valley Styria”.

REFERENCES

[1] H. Li and R. Hartley, "The 3D-3D Registration Problem Revisited," in *Proc. IEEE Intl. Conf. on Computer Vision (ICCV2007)*, 2007, pp. 1–8.

[2] A. Hietanen, J. Latokartano, A. Foi, R. Pieters, V. Kyrki, M. Lanz, and J.-K. Kämäräinen, "Object Pose Estimation in Robotics Revisited," 2020.

[3] J. Kim, M. Muramatsu, Y. Murata, and Y. Suga, "Omnidirectional vision-based ego-pose estimation for an autonomous in-pipe mobile robot," *Advanced Robotics*, vol. 21, no. 3-4, pp. 441–460, 2007.

[4] S. Thrun and J. J. Leonard, "Simultaneous Localization and Mapping," *Springer Handbook of Robotics*, pp. 871–889, 2008.

[5] J.-E. Deschaud, "IMLS-SLAM: Scan-to-Model Matching Based on 3D Data," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA2018)*, 2018, pp. 2480–2485.

[6] K. Fischer, M. Simon, S. Milz, and P. Mäder, "StickyLocalization: Robust End-To-End Relocalization on Point Clouds using Graph Neural Networks," in *IEEE/CVF Winter Conf. on Applications of Computer Vision (WACV2022)*, 2022, pp. 307–316.

[7] F. Pomerleau, F. Colas, and R. Siegwart, "A Review of Point Cloud Registration Algorithms for Mobile Robotics," *Foundations and Trends® in Robotics*, vol. 4, pp. 1–104, 05 2015.

[8] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP Variants on Real-World Data Sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, apr 2013.

[9] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI2016)*, 2016.

[10] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, p. 381–395, 1981.

[11] I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk, "Iteratively reweighted least squares minimization for sparse recovery," *Communications on Pure and Applied Mathematics*, vol. 63, no. 1, pp. 1–38, 2010.

[12] C. Choy, J. Park, and V. Koltun, "Fully Convolutional Geometric Features," in *Proc. IEEE/CVF Intl. Conf. on Computer Vision (ICCV2019)*, 2019, pp. 8957–8965.

[13] H. Deng, T. Birdal, and S. Ilic, "PPFNet: Global Context Aware Local Features for Robust 3D Point Matching," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition(CVPR2018)*, 2018, pp. 195–205.

[14] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR2019)*, 2019, pp. 7156–7165.

[15] Y. Wang and J. Solomon, "Deep Closest Point: Learning Representations for Point Cloud Registration," in *Proc. IEEE/CVF Intl. Conf. on Computer Vision (ICCV2019)*, 2019, pp. 3522–3531.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All You Need," in *Proc. Intl. Conf. on Neural Information Processing Systems, (NIPS2017)*, ser. NIPS'17, 2017, p. 6000–6010.

[17] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning Feature Matching with Graph Neural Networks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2020)*, 2020.

[18] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2012)*, 2012, pp. 3354–3361.

[19] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proc. Intl. Conf. on 3-D Digital Imaging and Modeling*, 2001, pp. 145–152.

[20] Q.-Y. Zhou, , J. Park, and V. Koltun, "Fast Global Registration," in *Proc. European Conf. on Computer Vision (ECCV2016)*, 2016, pp. 766–782.

[21] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige, "Going Further with Point Pair Features," in *Proc. European Conf. on Computer Vision (ECCV2016*, 2016, pp. 834–848.

[22] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA2009)*, 2009, pp. 3212–3217.

[23] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A Comprehensive Performance Evaluation of 3D Local Feature Descriptors," *Intl. Journal of Computer Vision*, vol. 116, no. 1, pp. 66–89, 2016.

[24] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2017)*, 2017, pp. 77–85.

[25] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems (NIPS2017)*, vol. 30, 2017.

[26] M. Saleh, S. Dehghani, B. Busam, N. Navab, and F. Tombari, "Graphite: Graph-Induced Feature Extraction for Point Cloud Registration," in *Proc. Intl. Conf. on 3D Vision (3DV2020)*, 2020, pp. 241–251.

[27] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2019)*, 2019.

[28] C. Choy, W. Dong, and V. Koltun, "Deep Global Registration," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR2020)*, 2020, pp. 2511–2520.

[29] Z. Gojcic, C. Zhou, J. D. Wegner, and W. Andreas, "The Perfect Match: 3D Point Cloud Matching with Smoothed Densities," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR2019)*, 2019.

[30] K. Fischer, M. Simon, F. Olsner, S. Milz, H.-M. Gross, and P. Mader, "StickyPillars: Robust and Efficient Feature Matching on Point Clouds Using Graph Neural Networks," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR2021)*, June 2021, pp. 313–323.

[31] S. Huang, Z. Gojcic, M. Usvyatsov, A. Wieser, and K. Schindler, "Predator: Registration of 3D Point Clouds With Low Overlap," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR2021)*, June 2021, pp. 4267–4276.

[32] Z. Qin, H. Yu, C. Wang, Y. Guo, Y. Peng, and K. Xu, "Geometric Transformer for Fast and Robust Point Cloud Registration," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR2022)*, June 2022, pp. 11 143–11 152.

[33] J. Li, C. Zhang, Z. Xu, H. Zhou, and C. Zhang, "Iterative Distance-Aware Similarity Matrix Convolution with Mutual-Supervised Point Elimination for Efficient Point Cloud Registration," in *Proc. European Conf. on Computer Vision (ECCV2020)*, 2020, pp. 378–394.

[34] W. Yuan, B. Eckart, K. Kim, V. Jampani, D. Fox, and J. Kautz, "DeepGMR: Learning Latent Gaussian Mixture Models for Registration," in *Proc. European Conf. on Computer Vision (ECCV2020)*, 2020, pp. 733–750.

[35] Z. J. Yew and G. H. Lee, "RPM-Net: Robust Point Matching using Learned Features," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2020)*, 2020.

[36] M. Cuturi, "Sinkhorn Distances: Lightspeed Computation of Optimal Transport," in *Proc. Intl. Conf. on Neural Information Processing Systems (NIPS2013)*. Red Hook, NY, USA: Curran Associates Inc., 2013, p. 2292–2300.

[37] K. Fu, J. Luo, X. Luo, S. Liu, C. Zhang, and M. Wang, "Robust Point Cloud Registration Framework Based on Deep Graph Matching," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2021)*, 2021.

[38] H. W. Kuhn, "The Hungarian Method for the assignment problem," *Naval Research Logistics Quarterly*, pp. 83–97, 1955.

[39] G. Peyré and M. Cuturi, "Computational Optimal Transport: With Applications to Data Science," *Found. Trends Mach. Learn.*, vol. 11, no. 5–6, pp. 355–607, 2019.

[40] D. Chicco, "Siamese Neural Networks: An Overview," *Artificial Neural Networks*, pp. 73–94, 2021.

[41] R. Sinkhorn and P. Knopp, "Concerning Nonnegative Matrices And Doubly Stochastic Matrices," *Pacific Journal of Mathematics*, vol. 21, no. 2, 1967.

[42] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A Deep Representation for Volumetric Shapes," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2015)*, 2015.

[43] (2023, april) ModelNet40 2048 pre-sampled. [Online]. Available: https://shapenet.cs.stanford.edu/media/modelnet40_ply_hdf5_2048.zip

[44] (2023, april) Artec Leo. [Online]. Available: https://www.artec3d.com/portable-3d-scanners/artec-leo

Revisiting Distribution-Based Registration Methods

Himanshu Gupta[1], Henrik Andreasson[1], Martin Magnusson[1], Simon Julier[2], and Achim J. Lilienthal[1,3]

***Abstract*— Normal Distribution Transformation (NDT) registration is a fast, learning-free point cloud registration algorithm that works well in diverse environments. It uses the compact NDT representation to represent point clouds or maps as a spatial probability function that models the occupancy likelihood in an environment. However, because of the grid discretization in NDT maps, the global minima of the registration cost function do not always correlate to ground truth, particularly for rotational alignment. In this study, we examined the NDT registration cost function in-depth. We evaluated three modifications (Student-t likelihood function, inflated covariance/heavily broadened likelihood curve, and overlapping grid cells) that aim to reduce the negative impact of discretization in classical NDT registration. The first NDT modification improves likelihood estimates for matching the distributions of small population sizes; the second modification reduces discretization artifacts by broadening the likelihood tails through covariance inflation; and the third modification achieves continuity by creating the NDT representations with overlapping grid cells (without increasing the total number of cells). We used the Pomerleau Dataset evaluation protocol for our experiments and found significant improvements compared to the classic NDT D2D registration approach (27.7% success rate) using the registration cost functions "heavily broadened likelihood NDT" (HBL-NDT) (34.7% success rate) and "overlapping grid cells NDT" (OGC-NDT) (33.5% success rate). However, we could not observe a consistent improvement using the Student-t likelihood-based registration cost function (22.2% success rate) over the NDT P2D registration cost function (23.7% success rate). A comparative analysis with other state-of-art registration algorithms is also presented in this work. We found that HBL-NDT worked best for easy initial pose difficulties scenarios making it suitable for consecutive point cloud registration in SLAM application.**

I. INTRODUCTION

Point cloud registration is used in various computer vision tasks like point cloud matching, 3D reconstruction, localization and mapping, and odometry estimation [1] [2]. In literature, several registration algorithms are available such as iterative closest point (ICP), which utilizes point [3] and point-normal [4] as features to find the correspondence between point clouds. Normal distribution transform (NDT) registration uses distribution transform maps [5]. Furthermore, coherent point drift (CPD) solves point cloud registration as a probability density estimation problem where the point cloud is assumed to be a Gaussian mixture model (GMM). Several other variants of these registration methods are available in the literature, and recently the focus is shifting to using deep learning for registration [2]. NDT registration is a fast, learning-free method that works well in diverse environments, which has been used in research and the industry for more than 15 years and is the main focus of this work.

NDT registration uses a discrete and compact representation of point cloud [6] called NDT maps, a collection of normal distributions (μ_i, Σ_i) of the points in fixed-size grid cells. There are two types of NDT registration, NDT point-to-distribution (NDT-P2D) registration finds the pose variation between a NDT map and a point cloud, and NDT distribution-to-distribution (NDT-D2D) registration matches two NDT maps. The grid discretization in NDT maps can be seen as the discretization of surface geometry estimation due to the point cloud's voxelization. Due to the discretization of NDT maps, NDT registration has an inherent problem with the global minima of registration cost function not always being at the ground truth pose variation between point clouds. In this work, we present and evaluate three modifications in the NDT registration cost function to reduce the effect of discretization of the NDT map, which results in the following contributions.

- The effect of the normalization term of Gaussian distribution on the registration cost function, which is considered a constant.
- Deriving and evaluating registration cost function based on Student-t likelihood function and NDT maps (Section III-A).
- Proposing and evaluating the modification in NDT registration based on cost function smoothing, HBL-NDT (Section III-B) and by creating a more continuous NDT map, OGC-NDT (Section III-C).
- Performance comparison of modified NDT registration with state-of-art registration methods using the Pomerleau dataset (Section IV-B).

II. RELATED WORK

A. NDT Registration

NDT registration finds the transformation between two point clouds using their NDT map representation. The NDT map is a collection of NDT cells created by sub-dividing the point cloud into fixed-size non-overlapping grid cells.

*This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 858101.

[1]Himanshu Gupta, Henrik Andreasson, and Martin Magnusson are with the Centre for Applied Autonomous Sensor Systems (AASS), Orebro University, Sweden. firstname.lastname@oru.se

[2]Simon Julier is with the Department of Computer Science, University College London, UK. s.julier@ucl.ac.uk

[3]Achim J. Lilienthal is with Perception for Intelligent Systems, Technical University of Munich, Germany and Centre for Applied Autonomous Sensor Systems (AASS), Orebro University, Sweden., achim.lilienthal@tum.de

979-8-3503-0704-7/23/$31.00

For each grid cell, the points ($p_i = (x_i, y_i, z_i)^T, i = 1...n_p$) distribution in the grid cells is estimated using the normal distribution ($\mathcal{N}(\mu, \Sigma)$). Our experiments used grid cells where $n_p \geq 5$ for registration cost calculation and grid cells with $n_p < 5$ were discarded.

$$\mu = \frac{1}{n}\sum_{i}^{n} p_i \tag{1}$$

$$\Sigma = \frac{1}{n-1}\sum_{i}^{n}(p_i - \mu)(p_i - \mu)^T \tag{2}$$

Broadly, NDT registration cost functions are of two types, point-to-distribution (P2D) and distribution-to-distribution (D2D), which maximize the total likelihood of points in the NDT map and the similarity between NDT maps, respectively, with respect to the rigid transformation matrix Θ. The NDT P2D registration cost function is the approximation of the negative log-likelihood of the points in the source point cloud ($\mathcal{X}$) belonging to the NDT cells of target NDT map ($\mathcal{M}$). The NDT P2D cost function is represented by (3) where k_1 and k_2 are regularization parameters described in [7].

$$f_{p2d} = \sum_{x}^{\mathcal{X}} \sum_{\mu,\Sigma}^{\mathcal{M}} -k_1 \exp\left(\tfrac{-k_2}{2}(T(x,\Theta) - \mu)^T \Sigma^{-1}(T(x,\Theta) - \mu)\right) \tag{3}$$

The NDT D2D registration cost function represents the similarity between the NDT representation of the source ($\mathcal{M}_S$) and target ($\mathcal{M}_{prv}$) point cloud as the negative summation of $L2$ distance between NDT cells.

$$f_{d2d} = \sum_{i=1}^{N_{\mathcal{M}_S}} \sum_{j=1}^{N_{\mathcal{M}_T}} -k_1 \exp\left(\frac{-k_2}{2} d_{ij}\right) \tag{4}$$

where,

$$\mu_{ij} = T(\mu_i, \Theta) - \mu_j, \quad \Sigma_{ij} = R^T \Sigma_i R + \Sigma_j$$

B. Background

One of the reasons for incorrectness in the NDT registration is the discretization of NDT maps which is a well know issue and has been addressed in previous works [6] [8] [9]. The two main approaches used in previous literature for tackling this issue are hierarchical registration and overlapped grid cells. In [8] [10], a hierarchical registration approach in which registration was done multiple times with NDT maps of different resolutions (coarse-to-fine cell size) was used. The result of coarse NDT registration becomes the initial guess for fine NDT registration for faster convergence. However, this approach takes longer than single-step registration as registration is done in multiple steps with NDT maps of different resolutions. The second approach to rectify the problem of discreteness used in [6] is to create overlapping grid cells. This approach results in continuous NDT representation with an increased number of NDT cells, which results in higher accuracy and significantly increases computation time.

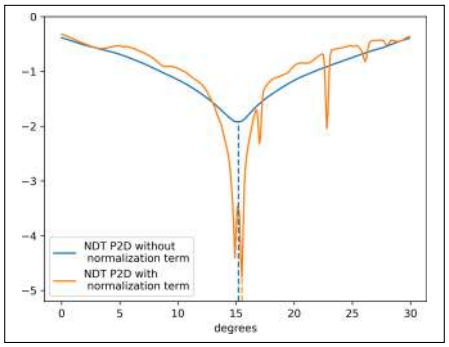


Fig. 1. Plot of NDT P2D cost function with (Orange) and without (Blue) the normalization term of the normal likelihood function. The cost function minimas are not at the ground truth pose for both conditions.

In [11], a new concept of dynamic scaling factors was introduced that scales the covariance of the NDT representation dynamically in each iteration to rectify the issue of NDT map discreteness and negative correlation of normal likelihood with rotation alignment. The method was evaluated on consecutive scan registration, and the effect of initial pose difficulty or point cloud overlap was not investigated.

Recently, several GMM-based registration approaches [12] [13] have proposed using the Student-t distribution instead of the Normal distribution in the cost function. The GMM-based registration method with Student-t distribution methods results in better convergence than GMM-based registration with Gaussian distribution due to the robustness against outliers and noise and better distribution estimation for small population sizes. However, no work has yet analyzed Student-t likelihood as a registration cost function for NDT registration.

In this work, we empirically studied the NDT registration cost function in detail and introduced three different modifications to improve the pairwise registration results by either using a better estimate of point distribution or by reducing the effect of discreteness. The first modification uses Student-t likelihood as a registration cost function instead of Gaussian likelihood registration to better estimate the likelihood of matching the distributions of small population sizes. The second modification, a heavily broadened likelihood NDT (HBL-NDT) registration cost function, was inspired by the broader-tailed Student-t likelihood function and has shown improvement in scan registration. In the third approach, an NDT map with overlapping grid cells (OGC-NDT) is used without increasing the number of cells; hence, the computation time of registration does not increase while registration results improve.

III. Proposed Modifications

The likelihood of point (x) measured in multivariate normal distribution ($\mathcal{N}(\mu, \Sigma)$) with dimension d is calculated using (5).

$$\mathcal{L}(x) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \tag{5}$$

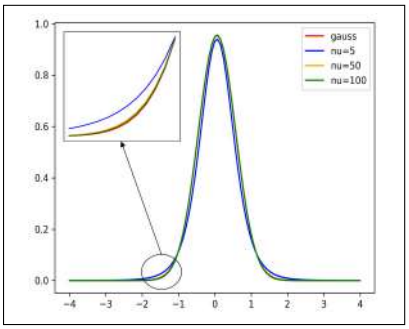


Fig. 2. Likelihood plots for 1D Gaussian, and Student-t distribution that shows broader tail for Student-t distribution.

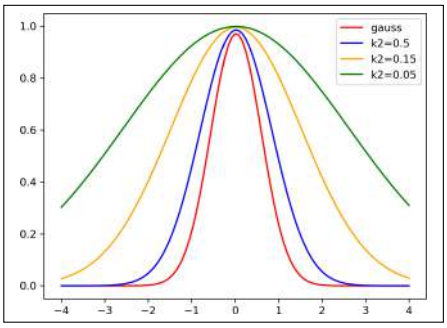


Fig. 3. Likelihood plot for 1D point distribution with heavily broadened likelihood.

Both P2D and D2D NDT registration approximate the negative log-likelihood function with a scaling factor k_2, which can be summarized as the negative summation of the exponent of the square of Mahalanobis distance (d_{ij}) as given in (3) and (4) without the normalization term. To test the normalization term's effect on registration, we plotted the NDT P2D registration cost function with and without the normalization term as shown in Fig. 1. To plot the figure, we rotated the Stanford dataset's Dragon point cloud by $-15°$, and calculated registration cost by rotating the transformed point cloud from $0°$ to $30°$ at an interval of $0.1°$. From Fig.1, we see that the normalization term of likelihood negatively impacts the registration cost function with more local minima in the cost function compared to the cost function without the regularization term. Also, the ground truth ($15°$) is not at the global minimum of the cost function plot for normalized and unnormalized costs in this case.

A. Student-t (StDT) registration cost functions

The Student-t likelihood of point x being in the distribution ($\mathcal{N}(\mu, \Sigma)$) is calculated using (6). And (7) is the likelihood of point x being part of the NDT map $\mathcal{M}$ which is expressed as the summation of (6), and the likelihood of the point cloud $\mathcal{X}$ at certain pose Θ being a part of an NDT map $\mathcal{M}$ can be given as the product of (7) and expressed as (8).

$$\mathcal{L}(x) = \frac{k}{|\Sigma|^{1/2}} \left[1 + \frac{1}{\nu}(x-\mu)^T \Sigma^{-1}(x-\mu)\right]^{-(\nu+p)/2} \tag{6}$$

$$\mathcal{L}(x|\mathcal{M}) = \sum_{i=1}^{n_M} \frac{k}{|\Sigma_i|^{1/2}} \left[1 + \frac{1}{\nu_i}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right]^{-(\nu_i+p)/2} \tag{7}$$

$$\mathcal{L}(X, \Theta|\mathcal{M}) = \prod_{j=1}^{n_X} \mathcal{L}(T(x_j, \Theta)|\mathcal{M}) \tag{8}$$

The best pose Θ that fits the point cloud $\mathcal{X}$ to the NDT map $\mathcal{M}$ should maximize the likelihood function (8) or, equivalently, minimize the negative log-likelihood (9).

$$-\log(\mathcal{L}(\mathcal{X}, \Theta|\mathcal{M})) = -\sum_{j=1}^{n_X} \log(\mathcal{L}(T(x_j, \Theta)|\mathcal{M})) \tag{9}$$

Similar to the NDT P2D registration cost function and the conclusions from the effect of normalization term on the registration cost function (Fig. 1), the StDT P2D registration algorithm will minimize the approximation of the negative log-likelihood as given in (10), over the space of transformation parameters Θ

$$f_{StDT} = -\sum \left[1 + \frac{k_2}{\nu}(T(x,\Theta)-\mu)^T \Sigma^{-1}(T(x,\Theta)-\mu)\right]^{\frac{\nu+d}{2}} \tag{10}$$

where ν is the degree of freedom and d is number of spatial dimension. By increasing the value of ν, the Student-t distribution approximates the Normal distribution. By increasing the value of ν, the Student-t distribution approximates the Normal distribution as shown in Fig. 2. Fig. 2 shows the likelihood plot of Gaussian and Student-t distribution for 2d point distribution with $\nu = \{5, 50, 100\}$ and the number of points is 30. Given that the Student-t likelihood has broader tails for $\nu = 5$ and the NDT cell with $n_p \geq 5$ was used during experiments, we report the StDT registration results with $\nu = 5$. Higher values of ν were also used for experiments, but not much difference was observed.

B. Heavily broadened likelihood NDT (HBL-NDT) registration cost function

The aim of using the Student-t likelihood function was its robustness and capability of better likelihood estimation for a small sample size because of broader tails compared to normal distribution. We have proposed a way to artificially broaden the likelihood curve of the normal distribution to smoothen the cost function, reducing discreteness. For a normal distribution, lowering the value of k_2 in Equation (3) and (4) makes the likelihood tail broad. Fig. 3 shows the effect of k_2 on tail broadness for normal distribution in the case of 2D point distribution. This modification can be interpreted as inflating the covariance matrix with a factor of $s = 1/k_2$. We experimented with different values of $k_2 = \{0.05, 0.15, 0.5\}$ and reported the best result obtained using $k_2 = 0.15$.

C. Overlapping grid cells NDT (OGC-NDT) registration cost function

The NDT map's discreteness due to grid cells can be viewed as discreteness in estimating the surface geometry

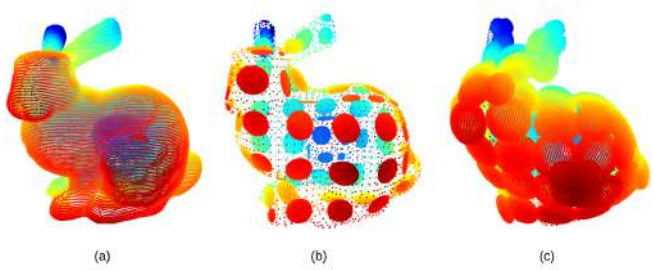


Fig. 4. NDT map representation for (b) non-overlapping grid cells and (c) overlapping grid cells for (a) Bunny point cloud of Stanford Dataset.

using the NDT cell's covariance matrix. Fig. 4 displays plots of covariance matrices as ellipsoids representing the approximate surface geometry for NDT map representations. In previous work, overlapping grid cells were created by adding an NDT cell at the boundary of two regular NDT cells, which reduces the discreteness but increases the computation time dramatically due to an increase in the number of NDT cells. In our approach, the computation of the mean and covariance for NDT cells do not change the number of cells; hence the computation time does not change much. Equations (11) and (12) show the calculation for distribution's mean μ' using the points inside the grid cell and calculation of distribution's covariance Σ' using points in a box bigger than the cell size (a) positioned at the center (c) of the cell. In Eq.12, k is a factor to increase the point search radius parameter in proportion to cell size for covariance calculation. During experiments, we used $k = 1.2$.

$$\mu' = \frac{1}{n}\sum_{i}^{n} p_i, \forall p_i : |p_i - c| < a/2 \tag{11}$$

$$\Sigma' = \frac{1}{n'-1}\sum_{i}^{n'} (p_i - \mu')(p_i - \mu')^T, \forall p_i : |p_i - c| < k \times a \tag{12}$$

IV. Experiments and Results

A. Datasets and Evaluation

We evaluated the modified NDT registration methods using the Pomerleau dataset [14] following the protocol described in [15]. The Pomerleau dataset includes a protocol file and validation file for each scenario. The protocol file specifies the scan pair and initial pose to be used, and the validation file includes information on the initial pose difficulty, overlap ratio of the scan pair, and ground truth pose. We uniformly sub-sampled 10% of the scan pairs from the protocol file for each scenario. Additionally, we compared our modified NDT methods with several state-of-the-art registration algorithms, including ICP, NDT, CPD, TEASER++ [16], and FuzzyPSR [17] registration to provide a comprehensive analysis of the registration methods.

The success rate of scan registration for rotation and translation is reported separately and compared for different registration algorithms. The registration was considered successful if the translation error (e_t) was less than 10cm and the rotation error (e_r) was less than 2.5°. The translation and rotation error is calculated using (13) and (14) respectively, given the ground truth pose variation (ΔT) between two scans and estimated pose variation ($\Delta \hat{T}$) from registration.

$$e_t = ||\delta_t|| \tag{13}$$

$$e_r = \cos^{-1}\left(\frac{1}{2}(trace(\delta_R) - 1)\right) \tag{14}$$

where,

$$\delta = \Delta\hat{T}^{-1}\Delta T = \begin{bmatrix} \delta_R & \delta_t \\ 0 & 1 \end{bmatrix}$$

For optimization of the registration cost function, we used the Ceres Solver [18], which uses auto-differentiation; hence, the derivative of cost functions was not manually derived. In all cases, the initial guess of the pose (Θ) for optimization was the identity matrix. All-to-all correspondence was used for all NDT and StDT registrations to get the best registration result. For distribution-based transform registration, the grid cell size was 0.5m in all experiments.

The evaluation results for different registration algorithms for difficulty in pose and point cloud overlap are shown in Fig. 5 and Fig. 6, respectively. The plots show the successful registration rate (%) vs. different scenarios in the Pomerleau dataset for various registration algorithms.

B. Results

1) Comparison of NDT P2D and StDT P2D registration cost function: From Fig. 5 and Fig. 6, we observe that the overall successful translation registration rate for StDT P2D registration (22.25%) is close to the successful registration rate for NDT P2D registration (23.69%) for different pose difficulty or point cloud overlap. The reason might be the similarity in the likelihood plots for both Gaussian and Student-t distributions, as shown in Fig. 2. The StDT likelihood has a broader likelihood tail, but the broadness is not enough to curb the discreteness in NDT maps; thus, no improvement in registration results can be seen.

2) Effect of heavily broaden likelihood: This modification involves reducing the value of k_2 in equations 3 and 4, resulting in a heavily broadened likelihood, which in turn leads to a smoothened registration cost function. The smoothening effect due to the modification is similar to inflating the covariance matrices resulting in increased continuity in the NDT maps. This continuity can result in improved registration results, as evidenced by the results in Fig. 5 and Fig. 6. The successful translation registration increased from 27.7% to 34.7% by this simple modification. It is worth noting that the improvement obtained with this modification was higher in translation alignment compared to rotation alignment for different pose difficulties and point cloud overlap. From the results, it can be concluded that the modification of reducing the value of k_2 in equations 3 and 4 can lead to improved registration results in the context of NDT point cloud registration.

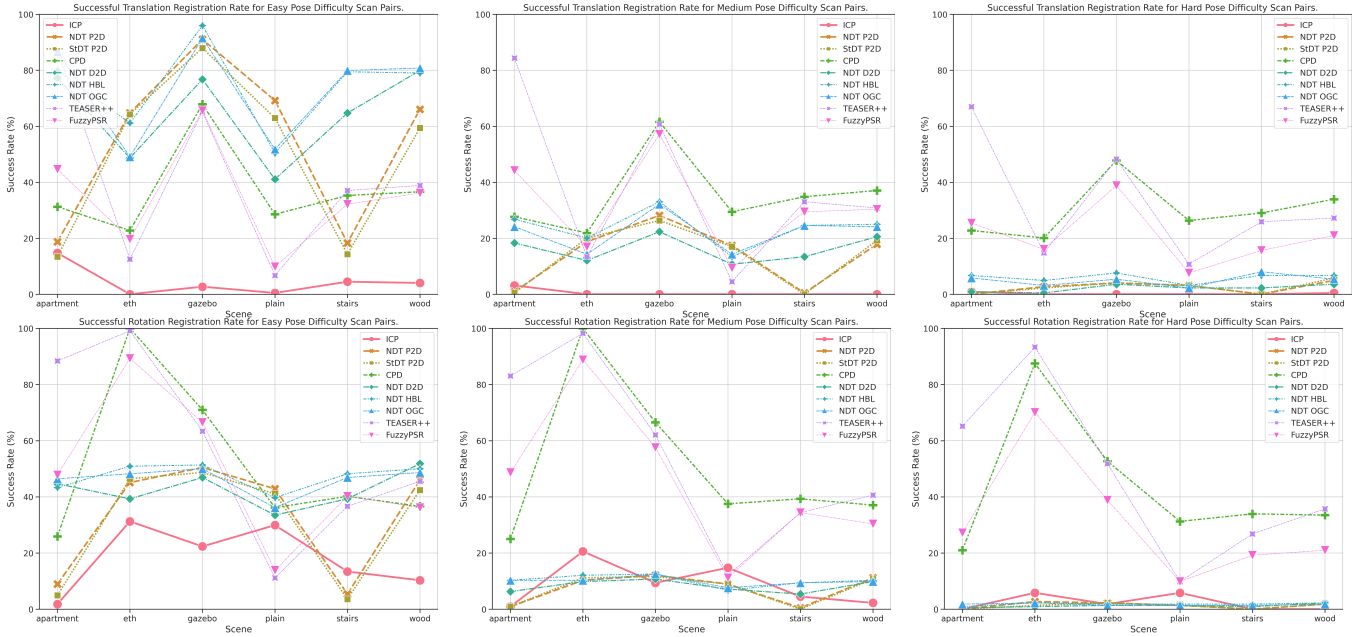


Fig. 5. Line graph showing the successful translation (Top Row) and rotation (Bottom Row) registration (%) vs. registration algorithms for easy (Left), medium (Middle), and hard (Right) pose difficulty. The line graph is used to enhance visibility.

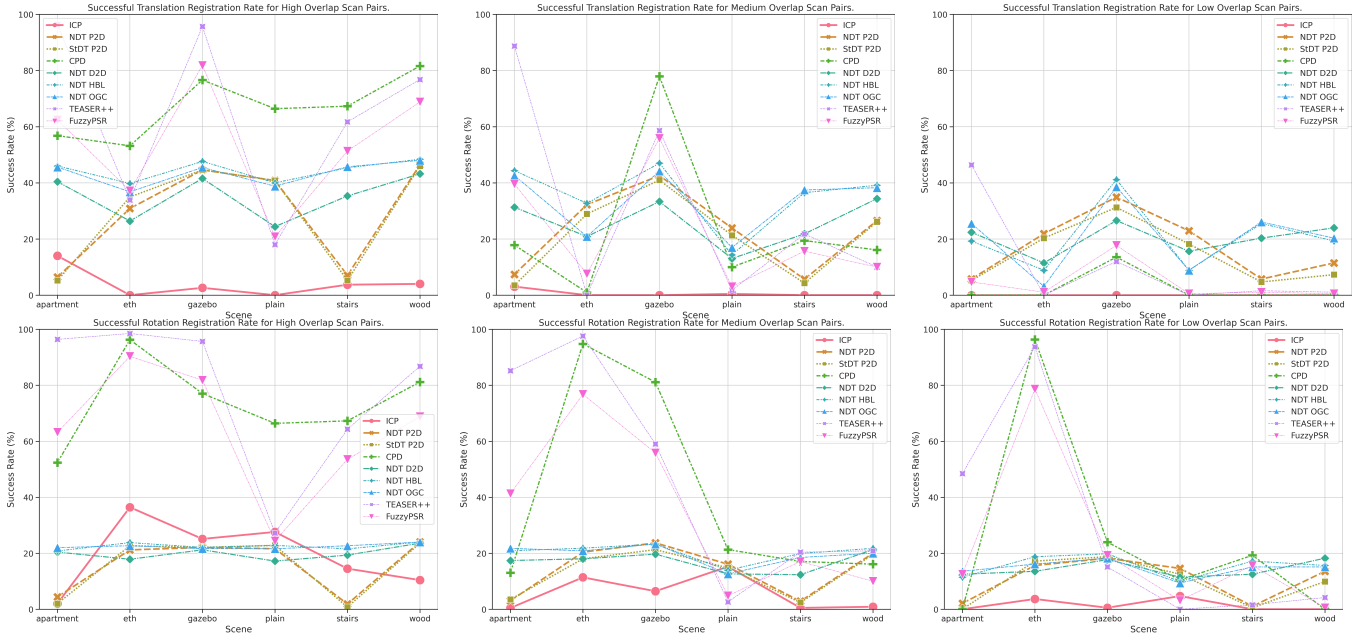


Fig. 6. Line graph showing the successful translation (Top Row) and rotation (Bottom Row) registration (%) vs. registration algorithms for high (Left), medium (Middle), and low (Right) point cloud overlap. The line graph is used to enhance visibility.

TABLE I

OVERALL SUCCESS RATE (%) FOR VARIOUS REGISTRATION METHODS ON THE POMERLEAU DATASET.

	apartment		eth		gazebo		plain		stairs		wood		Total	
	T	R	T	R	T	R	T	R	T	R	T	R	T	R
ICP	6.25	0.89	0.00	19.20	0.89	11.16	0.15	16.82	1.49	5.95	1.49	4.17	1.71	9.70
NDT P2D	6.55	3.27	28.72	19.35	41.07	21.58	**29.91**	17.71	6.25	1.93	29.61	19.64	23.69	13.91
StDT P2D	4.61	1.93	**28.87**	19.64	39.43	20.83	27.68	17.11	4.76	1.19	28.12	18.30	22.25	13.17
CPD	27.23	23.96	21.58	95.83	**59.08**	**63.39**	28.12	**34.97**	33.04	**37.80**	35.86	35.71	34.15	48.61
NDT D2D	32.14	17.11	20.39	16.67	34.23	19.64	18.01	13.99	26.79	15.18	34.67	21.28	27.70	17.31
NDT HBL	37.80	18.15	28.72	21.73	45.54	21.88	22.32	16.07	36.90	19.79	**36.90**	20.83	34.70	19.74
NDT OGC	38.84	19.49	22.17	20.24	43.01	21.28	22.77	15.03	**37.50**	19.20	36.76	20.09	33.51	19.22
TEASER++	**79.46**	**78.87**	13.54	**96.88**	58.04	59.08	7.29	11.01	31.99	32.59	32.29	**40.62**	**37.10**	**53.17**
FuzzyPSR	38.10	41.22	17.56	82.74	53.87	54.32	8.93	11.61	25.74	31.25	29.17	29.17	28.89	41.72

3) Effect of overlapping grid cells: In this modification, we create overlapping grid cells using the criterion given in Eq.12, which slightly increases the number of NDT cells ($\sim$1–2%) compared to the classic NDT map. This slight increase in the number of cells has only a minor effect on computation time while majorly improving the registration results compared to NDT D2D registration, as evident from Fig. 5 and Fig. 6. Interestingly, the OGC-NDT registration method shows less performance improvement than the HBL-NDT registration method, OGC-NDT: 33.5% and HBL-NDT:34.7%. However, the successful rotation registration rate was slightly higher for OGC-NDT (19.2%) than for HBL-NDT (17.7%). Overall, these results demonstrate the effectiveness of the proposed modification in improving the registration accuracy of NDT point clouds with only a small increase in NDT cells.

V. Discussion and Conclusion

This study investigated the distribution-based registration approach to improve the registration results by comparing several novel variants of NDT-based registration. The first part of our investigation involved an analysis of the effect of the normalization term of the Gaussian distribution on the registration cost function. Adding this term increased the local minima in the cost function, making optimization more difficult. We then examined the use of the Student-t likelihood as an NDT-based registration cost function and found that better likelihood estimation using Student-t does not consistently improve the registration results.

Based on the broader likelihood tail concept, we introduced and evaluated the HBL-NDT cost function, which smoothens the cost function and results in better registration. We also evaluated the OGC-NDT, which reduces the discreteness in the NDT map, resulting in successful registration rates. However, none of the proposed modifications in NDT showed significant improvements in successful rotation alignment compared to successful translation alignment, with cost function smoothening (HBL-NDT) having better results than overlapping grid cell NDT (OGC-NDT) overall. On the individual scan pair level, there were a few cases where one modification in NDT worked better while others did not. However, it is hard to pinpoint the reason for this.

We also compared various registration algorithms and found that the feature-based registration method, TEASER++ had the best performance overall regarding the percentage of successful registration, with consistent results for the initial pose difficulty and point cloud overlap. The best-performing registration algorithm for easy and hard pose difficulties was HBL-NDT and CPD, respectively. For high and low point cloud overlap, CPD and TEASER++ had the best performance, respectively. We did not compare the algorithms based on computation time, but the CPD algorithm was the most time-consuming as it computed all-to-all correspondence between points.

The performance of NDT registration was consistent with the scene complexity and is heavily impacted by the initial pose difficulty. HBL-NDT has the best performance for easy initial pose difficulty cases making it suitable for SLAM tasks. However, further improvements in data association techniques can help improve its performance for rotation alignment.

In conclusion, our study provides insights into the distribution-based approaches and proposes modifications to improve registration results. Our findings also provide a comparative analysis of various registration algorithms, which can guide researchers in selecting the best algorithm based on the specific requirements of their application.

References

[1] F. Pomerleau, F. Colas, R. Siegwart *et al.*, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends® in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[2] Z. Zhang, Y. Dai, and J. Sun, "Deep learning based point cloud registration: an overview," *Virtual Reality & Intelligent Hardware*, vol. 2, no. 3, pp. 222–246, 2020.

[3] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.

[4] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," *Chapel Hill, University of North Carolina*, vol. 4, no. 10, pp. 1–3, 2004.

[5] M. Magnusson, A. Nuchter, C. Lorken, A. J. Lilienthal, and J. Hertzberg, "Evaluation of 3d registration reliability and speed-a comparison of icp and ndt," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3907–3912.

[6] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3. IEEE, 2003, pp. 2743–2748.

[7] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3d-ndt," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007.

[8] E. Takeuchi and T. Tsubouchi, "A 3-d scan matching using improved 3-d normal distributions transform for mobile robotic mapping," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 3068–3073.

[9] M. Magnusson, "The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection," Ph.D. dissertation, Örebro universitet, 2009.

[10] C. Ulaş and H. Temeltaş, "3d multi-layered normal distribution transform for fast and long range scan matching," *Journal of Intelligent & Robotic Systems*, vol. 71, pp. 85–108, 2013.

[11] H. Hong and B. Lee, "Dynamic scaling factors of covariances for accurate 3d normal distributions transform registration," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1190–1196.

[12] Z. Zhou, J. Zheng, Y. Dai, Z. Zhou, and S. Chen, "Robust non-rigid point set registration using student's-t mixture model," *PloS one*, vol. 9, no. 3, p. e91381, 2014.

[13] Z. Tang, M. Liu, F. Zhao, S. Li, and M. Zong, "Toward a robust and fast real-time point cloud registration with factor analysis and student's-t mixture model," *Journal of Real-Time Image Processing*, vol. 17, pp. 2005–2014, 2020.

[14] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms," *The International Journal of Robotics Research*, vol. 31, no. 14, pp. 1705–1711, 2012.

[15] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets: Open-source library and experimental protocol," *Autonomous Robots*, vol. 34, pp. 133–148, 2013.

[16] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and Certifiable Point Cloud Registration," *IEEE Trans. Robotics*, 2020.

[17] Q. Liao, D. Sun, and H. Andreasson, "Fuzzypsreg: Strategies of fuzzy cluster-based point set registration," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2632–2651, 2021.

[18] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 3 2022. [Online]. Available: https://github.com/ceres-solver/ceres-solver

Enhancing Door–Status Detection for Autonomous Mobile Robots during Environment–Specific Operational Use

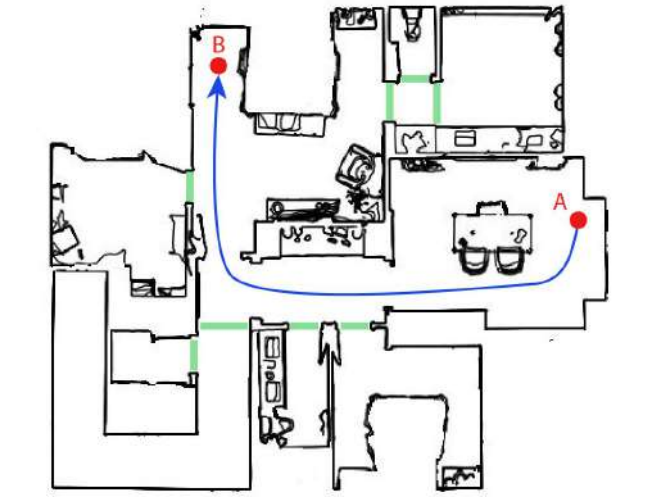
Michele Antonazzi, Matteo Luperto, Nicola Basilico, N. Alberto Borghese

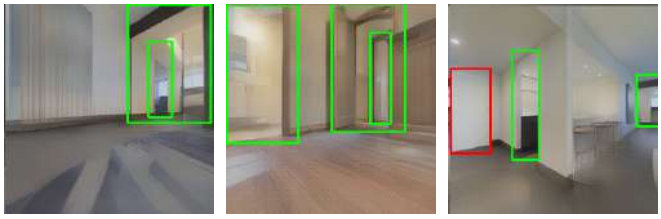
***Abstract*— Door–status detection, namely recognising the presence of a door and its status (open or closed), can induce a remarkable impact on a mobile robot's navigation performance, especially for dynamic settings where doors can enable or disable passages, changing the topology of the map. In this work, we address the problem of building a door–status detector module for a mobile robot operating in the same environment for a long time, thus observing the same set of doors from different points of view. First, we show how to improve the mainstream approach based on object detection by considering the constrained perception setup typical of a mobile robot. Hence, we devise a method to build a dataset of images taken from a robot's perspective and we exploit it to obtain a door–status detector based on deep learning. We then leverage the typical working conditions of a robot to *qualify* the model for boosting its performance in the working environment via fine–tuning with additional data. Our experimental analysis shows the effectiveness of this method with results obtained both in simulation and in the real–world, that also highlights a trade–off between the costs and benefits of the fine–tuning approach.**

I. Introduction

Autonomous mobile robots are nowadays increasingly employed for cooperating with humans in a variety of tasks settled in indoor public, private, and industrial workplaces. A challenge posed to these *service robots* is coping with highly dynamic environments characterised by features that can rapidly and frequently change, very often due to the presence of human beings [1]. Consider, as examples, a domestic setup in an apartment or a workspace with several offices. In a time span of hours or days, the topology itself of these environments might frequently change its connectivity, since doors may be left open or closed, hence modifying in time the reachability of free spaces. This phenomenon strongly impacts the capability of robots to efficiently navigate and perform their tasks. At the same time, during their operational time, robots are often exposed to large amounts of data about their surroundings that offer an opportunity to track, model, and predict doors' statuses (and topology variations). The relevance of this problem is well–established in the literature. Different works, such as [2], [3], show how modelling the status of doors across a long time span and predicting the changes in the environment topology improves a robot's task performance. Intuitively, better paths can be planned by taking into account whether a room will be reachable or not upon arriving there.

Central to unlocking such enhanced indoor navigation behaviours is what we call in this work *door–status detection*: the robot's capability to extract, from visual perceptions, the presence and location of a door and, at the same time, to recognise its traversability (open or closed status).





(a) (b) (c)

Fig. 1: A robot, navigating from A to B, can observe the status (open or closed) of different doors (highlighted in the top image). In this condition, door–status detection can be a difficult task as, from the robot's point of view, doors can be nested (a–b), doors can be hidden in the wall (c), or instead of a door sometimes there are just passages (a–c). The bounding boxes of open (closed) doors, as identified by our method, are shown in green (red). For the remainder of this work, we follow the same colour schema.

In this work, we propose a method to endow a robot with door–status detection capabilities that can be run during task–related autonomous navigation.

Door–status detection is particularly challenging for mobile robots operating indoor since clear and well–framed views of a door are seldom encountered during navigation. Fig. 1 depicts some typical instances of these challenges. While navigating, the robot can view nested doors (Fig. 1a, 1b), doors that are partially occluded (Fig. 1b, 1c), or closed

All authors are with the Department of Computer Science, University of Milan, Milano, Italy name.surname@unimi.it
979-8-3503-0704-7/23/$31.00 ©2023 IEEE

doors difficult to distinguish from their background (Fig. 1c).

To tackle the above problem, our approach starts with the choice of modelling door–status detection as a variant of object detection (OD) performed with deep neural networks. However, we found that OD deep learning methods, despite their great capabilities, exhibit important shortfalls when cast into the indoor robot navigation setting. Hence, our approach proposes a deployment methodology specific for mobile robots that allows harnessing the potentials of OD based on deep learning while solving what we recognised as the two most important limitations of such techniques in this domain.

First, OD methods are usually trained on large–scale datasets whose images are acquired from a human point of view. As a result, training examples follow a distribution that could be significantly different from the one generating the data perceived by a mobile robot. We show how popular datasets employed to train state–of–the–art deep learning detectors [4], [5], do not properly represent the embodied perception constraints and uncertainty typically characterising a mobile robot [6], thus causing generalisation issues.

Second, deep–learning OD modules are commonly trained with the main objective of obtaining a *general detector*. This model is trained once, stored, and is meant to work in previously unseen environments. These practices are not optimal when considering the typical working conditions of an indoor service robot. After an initial deployment phase, the robot is commonly used in the same environment for a long time, sometimes even for its entire life cycle. In such persistent conditions, the robot eventually observes the same doors multiple times, from different points of view, and under various environmental conditions. Also, different doors may present similar visual features (e.g., multiple doors of the same model). Against this operative background, and from a practical point of view, the ability to generalise in new environments becomes less important, while correctly performing door–status detection in challenging images from the deployment environment becomes paramount.

To address the first limitation, we devise a method for acquiring a large visual dataset from multiple photorealistic simulations taking into account the robot's perception model along realistic navigation paths. This allows us to train a deep *general* door–status detector with examples following a distribution compliant with the robot's perception capabilities. To deal with the second limitation, we exploit the robot's operational conditions to tailor our general detector for a given target environment. We obtain what we call a *qualified detector*, whose performance can substantially improve from the robot's experience enabling door–status detection in challenging instances (see the examples of Fig. 1). Our solution relies on fine–tuning sessions [7]–[9] of the general detector (which shall be considered as a baseline) with new examples from the target environment. These data can be collected and labelled, for example, during the robot installation phases or while the robot carries out its duties. (A setting motivated also by our on–the–field experience with assistive robots [10].)

We evaluate our approach by assessing its performance, also in the challenging cases exemplified in Fig. 1, with an extensive experimental campaign conducted in simulated settings and in different real–world environments and conditions, as perceived by a mobile robot during its deployment.

II. Related Works

Detecting a door's location can be useful for several tasks, as *room segmentation* [11], i.e., to divide the map of the environment into semantically meaningful regions (rooms), to predict the shape of unobserved rooms [12], or to do *place categorisation* [13], [14], which assigns to the rooms identified within the occupancy map a semantic label (e.g., *corridor* or *office*) according to their aspect.

Recent studies [2], [3] show how recognising door statuses can improve the navigation performance of robots in long–term scenarios. The work of [3] models the periodic environmental changes of a dynamic environment in a long–term run, while [2] proposes a navigation system for robots that operate for a long time in indoor environments with traversability changes.

Detecting doors in RGB images has been addressed as an OD task. Classical methods are based on the extraction of handcrafted features [15]–[17]. Deep learning end–to–end methods [18] provide significant improvements thanks to their capability of automatically learning how to characterise an object class, robustly to scale, shift, rotation, and exposure changes. As a significant example, the work of [19] describes a method for door detection with the goal of supporting and improving the autonomous navigation task performed by a mobile robot. A convolutional neural network is trained to detect doors in an indoor environment and its usage is shown to help a mobile robot to traverse passages in a more efficient way. Another approach, proposed in [20], focuses on robustly identifying doors, cabinets, and their respective handles in order to allow grasping by a robot. The authors use a deep architecture based on YOLO [9] to detect the Region Of Interest (ROI) of doors. This allows to obtain the handle's location by focusing only on the area inside the door ROI.

These works are representative examples of methods partially addressing the door–status detection problem in the mobile robotics domain. Indeed they do not explicitly consider the point of view of a mobile robot or do not take advantage of the robot's typical operational conditions. In this work, we devise an approach to overcome such limits.

III. Building a Doors Dataset for Mobile Robots

One of the key prerequisites to exploit deep learning to synthesise an effective door–status detector for a mobile robot is the availability of a dataset consistent with its challenging perception model (see Fig. 1). The examples contained in the dataset should follow three main desiderata. Images (i) should represent different environments with different features, thus allowing the model to learn how to generalise; (ii) should contain doors as observed from a point of view similar to the one of a robot navigating in an indoor

environment; (iii) should be taken from real environments or with an adequate level of photorealism.

An effective but impractical and time–consuming way to comply with the above requirements would be to deploy a robot on the field and having it exploring different environments while acquiring image samples of doors. The large overheads of such a procedure are well–known and a popular alternative is to rely on simulations [21] or publicly available datasets [4], [22], [23].

Meeting the desiderata (i)-(iii) in simulation is not straightforward since these are seldom guaranteed by available frameworks. For example, simulation tools popular in robotics such as Gazebo [24] or Unreal [25], while providing accurate physics modelling, fail to represent the realism and complexity of the perceptions in the real world. At the same time, public datasets as [4], [22], [23], do not well represent the point of view of a robot in its working conditions [6]. To address these issues, we resorted to Gibson [26], a simulator for embodied agents that focuses on realistic visual perceptions, and to the environments from Matterport3D [27], an RGB–D dataset of 90 real–world scans.

Given a simulated environment, we extract a set of poses that could describe views compatible with a mobile robot by applying a set of principles; the key ones include lying in the reachable free space (feasibility), ensuring a minimum clearance from obstacles, and being along the shortest paths between key connecting locations in the environment's topology. We achieve them with an extraction algorithm working in three phases: grid extraction, navigation graph extraction, and pose sampling.

The grid extraction phase aims at obtaining a 2D occupancy grid map, similar to those commonly used by mobile robots for navigation. We start from the environment's 3D mesh, and we aggregate obstacles from multiple cross–sections of the 3D mesh performed with parallel planes. The result is then manually checked for inaccuracies and artefacts produced during the procedure.

The *navigation graph* (shown in Fig. 2a) is a data structure that we use to represent the topology of the locations on the grid map that correspond to typical waypoints a robot occupies while navigating in the environment. We compute it from a Voronoi tessellation of the grid map by using obstacle cells as basis points [28], extracting graph edges from those locations that maintain maximum clearance from obstacles.

We then perform pose sampling on the navigation graph. The algorithm extracts from the graph a list of positions keeping a minimum distance D between them (this parameter controls the number and the granularity of the samples). A visual example of the algorithm's results is shown in Fig. 2b.

To build the dataset we acquire an image from the points of view of a robot's front–facing camera simulating its perceptions in the virtual environment from the sampled poses. Specifically, in each pose on the grid map, we acquire perceptions at two different height values ($0.1\,\mathrm{m}$ and $0.7\,\mathrm{m}$ – to simulate different embodiments of the robots) and at 8 different orientations (from $0°$ to $315°$ with a step of $45°$). Each acquisition includes the RGB image, the depth information, and the semantic data from Matterport3D. Since the semantic annotation of Matterport3D presents some inaccuracies, data labelling is manually performed by a human operator who specifies the door bounding boxes and the door status as open or closed. We considered 10 different Matterport3D environments (small apartments or large villas with multiple floors and a heterogeneous furniture style) by setting $D = 1\mathrm{m}$. The final dataset we obtained is composed of approx. 5500 examples.

(a) Navigation graph (b) Sampled poses

Fig. 2: Different phases of the pose extraction algorithm.

IV. Door–Status Detection for Mobile Robots

In this section, we first detail how we synthesised a *General Detector* (GD, Section IV-A) using a dataset generated with the approach of Section III. Subsequently, leveraging the assumption that the environment e will not change in its core features (location and visual aspect of doors) during the robot's long–term deployment, we introduce our *Qualified Detector* for e (QD_e) by applying a procedure based on fine–tuning [7]–[9], [29] of the GD on additional data that, in our envisioned scenario, can be acquired and labelled during the first setup of the robot in e.

A. General Door–Status Detector

As previously introduced, we aim at building and deploying door–status detectors for mobile robots leveraging deep–learning for object detection. In a preliminary experimental phase, we evaluated and compared three popular models suitable for such a task: DETR [29], Faster–RCNN [8], and a YOLO architecture [30]. We decided to adopt DETR since, with respect to the other two methods, it turned out to be easier to deploy in our robotic setting primarily due to two key features. First, DETR does not require setting in advance the number and dimension of anchors (i.e., sets of predefined bounding boxes used to make detections) according to the image resolution and the objects' shape, a task that instead the YOLO architecture requires. Second, both competitors require a final non–maximum suppression step to discard multiple detections of the same object. DETR, instead, matches each bounding box to a different object by construction. Hence, the methodology we describe in this paper and the empirical results evaluating it shall develop around DETR–based detectors. However, we stress the fact that our methods can be applied to any architecture, including those mentioned above, and, eventually, to their improvements.

DETR combines a CNN backbone based on ResNet [7] to produce a compact representation of an image and a transformer [31]. We used the pre–trained version of DETR on the COCO 2017 [4] dataset and, to adjust for door–status detection, we chose the smallest configuration provided by the authors.

The model requires setting one hyper–parameter, N, which determines the fixed number of bounding boxes predicted for each image. As a consequence, to filter out the detected doors, we select the $n \leq N$ bounding boxes whose confidence is not below a threshold ρ_c. We tuned N to be higher (but close) to the maximum number of doors in any single image of our dataset.

To train the general detector, we fixed the first two layers of the CNN backbone (as in [29]) with the weights of the pre–trained model. We then re–trained the remaining layers with images from the dataset of Section III. To achieve data augmentation, we generated additional samples by applying a random horizontal flip and resize transformation to a subset of the images (each training sample is selected for this procedure with a probability of 0.5).

B. Qualification on a Target Environment

Given a new environment e we use a randomly sampled subset of the images collected in it to fine–tune the GD, obtaining the qualified detector QD_e. To be used in the fine–tuning procedure, these images need to be labelled specifying the bounding boxes and the status for each visible door.

In our envisioned scenario, this data acquisition and labelling tasks can be carried out by a technician during the robot's first installation in e or in a second phase by uploading the data to a remote server. Such a setup phase requires to build the map of the environment (either autonomously or with teleoperation) by observing the entirety of the working environment and is very relevant to many real–world installations of collaborative robots, as we recently experienced with extensive on–the–field testing in the use case of assistive robotics [10]. This manual labelling task is quite time–expensive; yet it is required. In principle, we could use *pseudo–labels* automatically obtained by running the GD over the additional samples for incremental learning. Despite intriguing, we empirically observed that this is particularly challenging due to the fact that pseudo–labels are not enough accurate for this process. Recently, the work of [32] showed how pseudo–labels are particularly noisy and inaccurate: while they can be used to improve performance in tasks where precise labels are less important (like semantic segmentation), they are still too inaccurate to be used in object detection tasks, like the one investigated in this work. We observed how fine–tuning a general detector using pseudo–labels results in a performance degradation of about 20% when compared with the GD. These challenges are well-known and the approach we follow in this work is customary. See, for example, the work of [33], where manual annotations have been used to label 3D objects to fine–tune a model employed in long–term localisation. The study of methods to ease the burden of this task will be addressed as a part of our future work.

Finally, note that, while we focus here on a specific GD based on DETR [29], this method to obtain a qualified detector QD is general and can be applied to other deep learning–based models, such as YOLO [30] or Faster–RCNN [8].

V. Evaluation in Simulation

A. Experimental Setting

We evaluate our method using simulated data $\mathcal{D}$ obtained, as described in Section III, from 10 different Matterport3D [27] environments. We test the performance of our detectors on each environment e independently. First, we train the general detector GD_{-e} using the dataset $\mathcal{D}_{-e}$, where $\mathcal{D} = \{\mathcal{D}_{-e}, \mathcal{D}_e\}$, $\mathcal{D}_e$ contains all the instances acquired from poses sampled in environment e, and $\mathcal{D}_{-e} = \mathcal{D} \setminus \mathcal{D}_e$. This general detector will be used as a baseline in most of the evaluations we present. Then, we randomly partition the first subset as $\mathcal{D}_e = \{\mathcal{D}_{e,1}, \mathcal{D}_{e,2}, \mathcal{D}_{e,3}, \mathcal{D}_{e,4}\}$, where each $\mathcal{D}_{e,i}$ contains the 25% of the examples from e, randomly selected.

While $\mathcal{D}_{e,4}$ is reserved for testing, the remaining subsets are used to perform a series of fine–tuning rounds to obtain the corresponding qualified door–status detectors. Specifically, we fine–tune GD_{-e} using these three additional data subsets: $\{\mathcal{D}_{e,1}\}$, $\{\mathcal{D}_{e,1}, \mathcal{D}_{e,2}\}$, and $\{\mathcal{D}_{e,1}, \mathcal{D}_{e,2}, \mathcal{D}_{e,3}\}$. We denote the obtained qualified detectors as QD_e^{25}, QD_e^{50}, and QD_e^{75}, respectively. The superscript denotes the percentage of data instances from e that are required to fine–tune the general door–status detector. Such a percentage can be interpreted as an indicator of the cost to acquire and label the examples. To give a rough idea, labelling the 25% of the dataset (approximately 150 images) took a single human operator an effort of about 1 hour.

We empirically set the parameters of the door–status detector as $N = 10$ (number of bounding boxes) and $\rho_c = 0.75$ (confidence threshold). We conducted an extensive preliminary experimental campaign spanning different batch sizes ($\{1, 2, 4, 16, 32\}$) and the number of epochs ($\{20, 40, 60\}$) selecting 1 and 60 for the general detector and 1 and 40 for the qualified ones, respectively.

We measure performance with the average precision score (AP) used in the Pascal VOC challenge [5] by adjusting for a finer interpolation of the precision/recall curve to get a more conservative (in the pessimistic sense) evaluation. The AP is a popular evaluation metric widely adopted for object detection tasks, it represents the shape of the precision/recall curve as the mean precision over evenly distributed levels of recall. To accept a true positive, the bounding box computed by the network must exhibit an Intersection Over Union area (IOU) with one true bounding box above a threshold ρ_a, that we empirically set to 50%.

The source code of our simulation framework (Section III), the door–status detectors (Section IV), and the collected datasets are maintained in a freely accessible repository[1].

[1] https://github.com/aislabunimi/door-detection-long-term

B. Results

Table I reports the mean AP scores (averaged over the 10 environments) reached by the 4 detectors divided by label (closed door, and open door), the average increments (with respect to the detector immediately above in table) obtained with fine–tuning, and the standard deviation (σ). We also report the AP scores for every environment in Fig. 3. These results show the trade-off between performance increase (via fine–tuning) and costs due to data collection and labelling.

Exp.	Label	AP	σ	Increment	σ
GD_{-e}	Closed	34	12	–	–
	Open	48	12	–	–
QD_e^{25}	Closed	55	15	70%	58
	Open	60	10	30%	34
QD_e^{50}	Closed	64	12	21%	21
	Open	68	10	14%	11
QD_e^{75}	Closed	72	10	14%	9
	Open	72	9	7%	5

TABLE I: Average AP in Matterport3D environments.

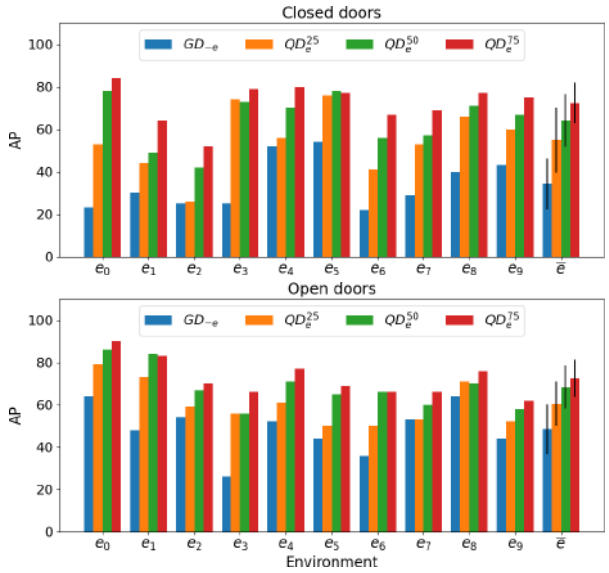


Fig. 3: AP scores in Matterport3D environments.

Results from Table I and Fig. 3 show that the general detector GD_{-e}, thanks to our dataset's consistency with the robot's perception model (see Section III), is able to correctly detect doors statuses in those cases where they are clearly visible, as shown in Fig. 4. However, while such a performance allows its use on a robot, there is significant room for improvement in detecting more challenging examples.

More interestingly, qualified detectors achieve a steep increase in performance. Unsurprisingly, the performance improves with more data (and data preparation costs) from QD_e^{25} to QD_e^{75}. However, it can be seen how QD_e^{25}, despite requiring a relatively affordable effort in manual labelling, obtains the highest performance increase. From a practical perspective, this shows how the availability of *a few* labelled examples from the robot target environment could be a good compromise between performance and costs to develop an

Fig. 4: Door–statuses found by the general detector GD_{-e}.

environment–specific door–status detector. This suggests that the number of examples that have to be collected and labelled on the field can be limited, thus promoting the applicability of our proposed framework. An example of this is shown in Fig. 5, where it can be seen how a QD_e^{25} (bottom row) fixes the mistakes of its corresponding general detector GD_{-e} (top row) in challenging images with nested or partially observed doors.

Fig. 5: Door–statuses as identified by GD_{-e} (top row) compared to QD_e^{25} (bottom row) in Matterport3D environments.

VI. Evaluation in the Real World

A. Experimental Settings

In this section, we evaluate the performance of our method with a real robot by using images collected by a Giraff–X platform [10] (Fig. 6a) during a teleoperated exploration of 3 single–floor indoor environments with multiple rooms from two buildings in our campus. Images were extracted from the robot's perceptions during navigation at 1 fps. As it commonly happens with real–world robot data, images are acquired in noisy environmental conditions with low–quality cameras, thus making the detection task even more difficult. In our setting, we used 320x240 RGB images acquired with an Orbbec Astra RGB–D camera.

First, we consider two general detectors. One is trained with simulated data $\mathcal{D}$, as in the previous section but with all the 10 environments. Another one is trained with real–world images from the publicly available *DeepDoors2* dataset (DD2) [23], which features 3000 images of doors that we re-labelled to include the ground truth for challenging examples

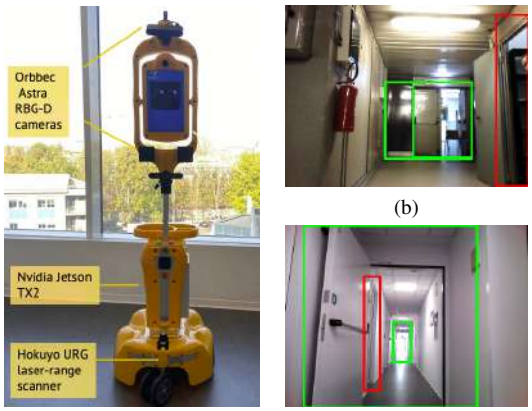
not originally provided. Comparing these two general detectors, we aim at assessing the advantages of training with a dataset following the principles we proposed in Section III instead of relying on mainstream datasets for classical object detection.

Subsequently, following the same steps of Section V-A, we qualify both GDs by using the 25, 50, and 75% of data collected in the three real environments.

B. Evaluation metrics

Analogously to what is done in simulation, we report the AP scores, but we argue that the real–world evaluation of our method can be conducted also with additional metrics, which are more representative of the actual application domain where door–status detection is meant to be cast.

The AP (as well as other metrics used in computer vision) presents some limitations when used in our context. Such a metric considers as false positives multiple bounding boxes assigned to the same door, as in Fig. 6b. However, a robot can easily disambiguate this by leveraging additional data such as its estimated pose and the map of the environment. On one side, although an erroneous localisation of the bounding box of a door (Fig. 6c) penalises the AP, it might have little effect in practice. On the other side, the AP is very marginally affected by a wrong detection of the door status if the bounding box is sufficiently accurate due to the fact that different labels are treated as two independent object classes, as the case in Fig. 6b. Conversely, the error of misleading a closed passage for an open one (and *vice versa*) can significantly impact the robot's performance when the robot translates such information into actions.



(a) (b) (c)

Fig. 6: Examples (b–c) of different types of errors made by a door–status detector mounted on our Giraff–X robot (a). While the AP considers all these errors in a similar way, our proposed metric considers them differently, according to their potential impact on robot performance.

To address this shortcoming and better capture performance in our robotic setting, we introduce three additional metrics. Consider a door i in a given image. If multiple bounding boxes are matched to i, where matching means $IOU \geq \rho_a$, the one with the maximum above–threshold (ρ_c) confidence is selected. If the status of the door is correctly identified, we consider it as a true positive (TP). Otherwise, we classify it as a False Positive (FP). All the remaining bounding boxes matched to i are, as per our previous considerations, ruled out from the evaluation. Finally, when a bounding box does not meet the IOU condition with any door in the image, we count it as a Background False Detection (BFD). A False Positive and a Background False Detection are errors that can play very different roles inside a robotic use case. While the first is likely to affect the robot's decisions, the second one might increase the uncertainty in the robot world–model. We scale the above metrics using the true number of doors in the testing set, denoted as GT, thus obtaining $TP_{\%} = TP/GT$, $FP_{\%} = FP/GT$, and $BFD_{\%} = BFD/GT$.

C. Results

Table II compares the average AP of the GD trained with DD2 [23] with that trained with our dataset $\mathcal{D}$. Intuitively, a model trained with real–world data (such as those featured in DD2) should have higher performance when used with real–world images, if compared with a model trained with simulated data (as $\mathcal{D}$). However, Table II shows how the GD and QDs trained with $\mathcal{D}$ have higher performance than those trained with DD2. This is because training images of $\mathcal{D}$, collected from the simulated point of view of a robot, better represent the actual distribution of robot perceptions, allowing us to fill, to some extent, the sim–to–real gap. Moreover, Table II shows that the fine–tuning operation to qualify general detectors to the target environment works remarkably well also when used in real–world conditions. For these reasons, from now on, we present results referring to the general detector trained with $\mathcal{D}$.

Exp.	Label	DeepDoors2 (DD2) AP	σ	Inc.	σ	Simulation dataset ($\mathcal{D}$) AP	σ	Inc.	σ
GD	Closed	5	3	–	–	13	10	–	–
	Open	18	5	–	–	31	11	–	–
QD_e^{25}	Closed	33	9	631%	240	53	9	508%	424
	Open	43	14	134%	20	55	14	83%	19
QD_e^{50}	Closed	52	7	66%	51	65	8	24%	15
	Open	51	14	18%	7	70	7	29%	22
QD_e^{75}	Closed	55	8	5%	6	72	8	10%	5
	Open	65	8	32%	18	78	8	13%	1

TABLE II: Average AP in real–world environments when DD2 dataset and our one ($\mathcal{D}$) are used to train the GD.

The performance of QDs is similar to that obtained in the far less challenging dataset of Table I. Most importantly, the performances of QD_e^{25} corroborate our findings from Section V-A, confirming how few additional examples can induce a significant increase in performance. Beyond the improvements observed in the average scores, QD_e^{25} managed to provide correct door–status detection in very challenging cases. We report in Fig. 7 some representative examples. As it can be seen, our detectors correctly recognised cases with nested doors, partially visible frames, and narrow side views. Another relevant example is in the second image (top row) of Fig. 7, where the qualified model succeeds in detecting a

white closed door in the background while, at the same time, not making a false detection of the white wardrobe doors on the right.

Fig. 7: Challenging door–statuses detected by QD_e^{25} in the real environments e_1, e_2, and e_3 (ordered by columns).

Env.	Exp.	GT	TP ($TP_{\%}$)	FP ($FP_{\%}$)	BFD ($BFD_{\%}$)
e_1	GD	235	71 (30%)	18 (7%)	51 (21%)
	QD_e^{25}	235	145 (61%)	10 (4%)	64 (27%)
	QD_e^{50}	235	179 (76%)	4 (1%)	44 (18%)
	QD_e^{75}	235	190 (80%)	4 (1%)	36 (15%)
e_2	GD	269	96 (35%)	17 (6%)	56 (20%)
	QD_e^{25}	269	192 (71%)	11 (4%)	87 (32%)
	QD_e^{50}	269	206 (76%)	6 (2%)	66 (24%)
	QD_e^{75}	269	228 (84%)	7 (2%)	60 (22%)
e_3	GD	327	62 (18%)	19 (5%)	108 (33%)
	QD_e^{25}	327	183 (55%)	22 (6%)	190 (58%)
	QD_e^{50}	327	230 (70%)	13 (3%)	103 (31%)
	QD_e^{75}	327	248 (75%)	8 (2%)	75 (22%)

TABLE III: Extended results in the real–world environments.

Table III reports the detailed results, for all three environments, of the metrics we defined in Section VI-B. The results show that GD, although it has a low number of wrong predictions (FP and BFD), is capable of detecting only a few of the GT doors in the images (TP). On the contrary, QDs dramatically improve performance, with QD_e^{25} showing a $TP_{\%}$ of 62% on average.

Among the three environments considered, we argue that e_3 is the more challenging, as it can be seen by the higher number of BFD. To cope with this, there are two possible directions. First, increasing the number of manually labelled examples reduces BFD (as can be seen already with QD_e^{50}). Alternatively, adopting a more conservative selection rule by increasing the confidence threshold ρ_c, at the cost of slightly reducing the number of TP. In Fig. 8, we show how $TP_{\%}$, $FP_{\%}$, and $BFD_{\%}$ for $QD_{e_3}^{25}$ change when varying ρ_c in e_3. Such an instance confirms how $\rho_c = 0.75$ is an acceptable trade–off among $TP_{\%}$ (high) and $BFD_{\%}$ (low) for such a detector.

In long–term runs, the illumination conditions of an environment might change from those of the initial setup, and this may affect the performance of the door–status detector. To test the robustness of our approach to this event, we acquire (following the same procedure of Section VI-A) data from environments e_1 and e_2 during nighttime, when only artificial

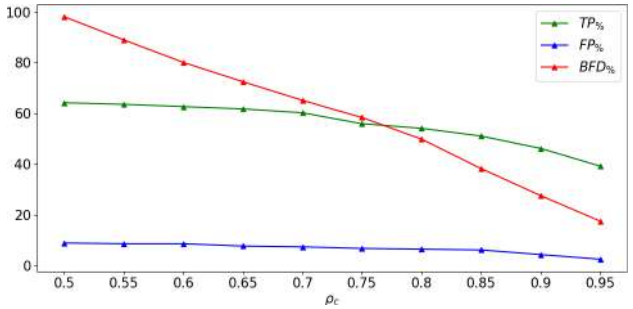


Fig. 8: $TP_{\%}$, $FP_{\%}$, and $BFD_{\%}$ obtained by $QD_{e_3}^{25}$ for increasing confidence thresholds.

light is present and some rooms are dark. Then, we use these images to test the GD and the QDs fine-tuned with data acquired during the initial setup time, with daylight.

Exp.	Label	AP	σ	Increment	σ
GD	Closed	14	18	–	–
	Open	31	8	–	–
QD_e^{25}	Closed	38	8	781%	1016
	Open	45	11	46%	3
QD_e^{50}	Closed	48	8	26%	8
	Open	53	17	17%	8
QD_e^{75}	Closed	54	8	14%	1
	Open	56	16	6%	5

TABLE IV: Average AP results in e_1 and e_2 tested in different light conditions with respect to those used to qualify the detector (day/night time).

Env.	Exp.	GT	TP ($TP_{\%}$)	FP ($FP_{\%}$)	BFD ($BFD_{\%}$)
e_1	GD	1079	334 (30%)	56 (5%)	150 (13%)
	QD_e^{25}	1079	532 (49%)	62 (5%)	306 (28%)
	QD_e^{50}	1079	572 (53%)	65 (6%)	299 (27%)
	QD_e^{75}	1079	634 (58%)	56 (5%)	248 (22%)
e_2	GD	1051	335 (31%)	68 (6%)	276 (26%)
	QD_e^{25}	1051	584 (55%)	55 (5%)	357 (33%)
	QD_e^{50}	1051	690 (65%)	40 (3%)	217 (20%)
	QD_e^{75}	1051	700 (66%)	48 (4%)	236 (22%)

TABLE V: Extended results in e_1 and e_2 tested in different light conditions with respect to those used to qualify the detector (day/night time).

The average AP obtained with different lighting conditions is reported in Table IV while the results of our extended metric (presented in Section VI-B) are shown in Table V. Comparing them with Tables II and III respectively, we can see how the performances of the GD are robust to illumination changes, as they are similar to those obtained during daytime. More interestingly, it can be seen how the improvement of QDs from the fine–tune is maintained also with different light conditions, with a slight performance decrease if compared to the results of Tables II and III. This is a direct consequence of the fine–tune, which produces QDs that slightly overfit the illumination conditions seen during training. Despite this, our method ensures a performance improvement to the GD when used in long–term scenarios with illumination changes, enabling the QDs to still solve

challenging examples, as shown in Fig. 9. Once again, QD_e^{25}, albeit using a few examples for fine–tuning, ensures the best performance improvement also under variable light conditions. See the video attachment, also linked in the repository, for additional examples of our method.

Fig. 9: Challenging nighttime examples classified by GD (top row) and QD_e^{25} (bottom row). QD_e^{25} is fine–tuned with examples obtained with daylight.

VII. Conclusions

In this work, we presented a door–status detection method for mobile robots. Our method, based on a deep learning architecture, allows robots to recognise open or closed doors in challenging situations. To train our model, we built a dataset of labelled images from photorealistic simulations taking into account the point of view of a mobile robot. We then fine–tuned a general model into a qualified one to increase performance in the robot's working environment.

Future work will investigate how to quantify and reduce the effort needed for labelling examples to qualify a general detector. Furthermore, we will investigate online fine–tuning methods towards the goal to have a robot that can learn with experience to better distinguish features in its environment.

References

[1] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajnik, "Artificial Intelligence for Long-Term Robot Autonomy: A Survey," *IEEE RA-L*, vol. 3, no. 4, pp. 4023–4030, 2018.

[2] L. Nardi and C. Stachniss, "Long-term robot navigation in indoor environments estimating patterns in traversability changes," in *Proc. ICRA*, 2020, pp. 300–306.

[3] T. Krajník, J. P. Fentanes, J. M. Santos, and T. Duckett, "Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments," *IEEE Trans. Rob.*, vol. 33, no. 4, pp. 964–977, 2017.

[4] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. ECCV*, 2014, pp. 740–755.

[5] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *Int. J. Comput. Vision*, vol. 88, pp. 303–338, 2009.

[6] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and P. Corke, "The limits and potentials of deep learning for robotics," *Int. J. Rob. Res.*, vol. 37, no. 4-5, pp. 405–420, 2018.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. CVPR*, 2016.

[10] M. Luperto, M. Romeo, J. Monroy, J. Renoux, A. Vuono, F.-A. Moreno, J. Gonzalez-Jimenez, N. Basilico, and N. A. Borghese, "User feedback and remote supervision for assisted living with mobile robots: A field study in long-term autonomy," *Robot Auton Syst*, vol. 155, p. 104170, 2022.

[11] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, "Room segmentation: Survey, implementation, and analysis," in *Proc. ICRA*, 2016, pp. 1019–1026.

[12] M. Luperto, F. Amadelli, M. Di Berardino, and F. Amigoni, "Mapping beyond what you can see: Predicting the layout of rooms behind closed doors," *Robot Auton Syst*, vol. 159, p. 104282, 2023.

[13] P. Espinace, T. Kollar, A. Soto, and N. Roy, "Indoor scene recognition through object detection," in *Proc. ICRA*, 2010, pp. 1406–1413.

[14] N. Sünderhauf, F. Dayoub, S. McMahon, B. Talbot, R. Schulz, P. Corke, G. Wyeth, B. Upcroft, and M. Milford, "Place categorization and semantic mapping on a mobile robot," in *Proc. ICRA*, 2016, pp. 5729–5736.

[15] N. Kwak, H. Arisumi, and K. Yokoi, "Visual recognition of a door and its knob for a humanoid robot," in *Proc. ICRA*, 2011, pp. 2079–2084.

[16] I. Monasterio, E. Lazkano, I. Rañó, and B. Sierra, "Learning to traverse doors using visual information," *Math. Comput. Simul.*, vol. 60, no. 3, pp. 347–356, 2002.

[17] X. Yang and Y. Tian, "Robust door detection in unfamiliar environments by combining edge and corner features," in *Proc. CVPR*, 2010, pp. 57–64.

[18] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, 2018.

[19] W. Chen, T. Qu, Y. Zhou, K. Weng, G. Wang, and G. Fu, "Door recognition and deep learning algorithm for visual based robot navigation," in *Proc. ROBIO*, 2014, pp. 1793–1798.

[20] A. Llopart, O. Ravn, and N. A. Andersen, "Door and cabinet recognition using convolutional neural nets and real-time method fusion for handle detection and grasping," in *Proc. ICCAR*, 2017, pp. 144–149.

[21] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, vol. 9, pp. 51 416–51 431, 2021.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. CVPR*, 2009, pp. 248–255.

[23] J. Ramôa, V. Lopes, L. Alexandre, and S. Mogo, "Real-time 2d–3d door detection and state classification on a low-power device," *SN Appl. Sci.*, vol. 3, 2021.

[24] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IROS*, 2004, pp. 2149–2154.

[25] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "Usarsim: a robot simulator for research and education," in *Proc. ICRA*, 2007, pp. 1400–1405.

[26] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *Proc. CVPR*, 2018, pp. 9068–9079.

[27] A. X. Chang, A. Dai, T. A. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from RGB-D data in indoor environments," *arXiv:1709.06158*, 2017.

[28] S. Thrun and A. Bücken, "Integrating grid-based and topological maps for mobile robot navigation," in *Proc. AAAI*, 1996, pp. 944–951.

[29] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. ECCV*, 2020.

[30] A. Farhadi and J. Redmon, "YoloV3: An incremental improvement," in *Proc. CVPR*, 2018, pp. 1804–2767.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, 2017, pp. 5998–6008.

[32] J. Frey, H. Blum, F. Milano, R. Siegwart, and C. Cadena, "Continual adaptation of semantic segmentation using complementary 2d-3d data representations," *IEEE-RAL*, vol. 7, no. 4, pp. 11 665–11 672, 2022.

[33] N. Zimmerman, T. Guadagnino, X. Chen, J. Behley, and C. Stachniss, "Long-term localization using semantic cues in floor plan maps," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 176–183, 2023.

Self-supervised Learning for Fusion of IR and RGB Images in Visual Teach and Repeat Navigation

Xinyu Liu*[1], Zdeněk Rozsypálek*[2] and Tomáš Krajník[2]

***Abstract*— With increasing computation power, longer battery life and lower prices, mobile robots are becoming a viable option for many applications. When the application requires long-term autonomy in an uncontrolled environment, it is necessary to equip the robot with a navigation system robust to environmental changes. Visual Teach and Repeat (VT&R) is one such navigation system that is lightweight and easy to use. Similarly, as other methods rely on camera input, the performance of VT&R can be highly influenced by changes in the scene's appearance. One way to address this problem is to use machine learning or/and add redundancy to the sensory input. However, it is usually complicated to collect long-term datasets for given sensory input, which can be exploited by machine learning methods to extract knowledge about possible changes in the environment from the data. In this paper, we show that we can use a dataset not containing the environmental changes to train a model processing infrared images and improve the robustness of the VT&R framework by fusion with the classic method based on RGB images. In particular, our experiments show that the proposed training scheme and fusion method can alleviate the problems arising from adverse illumination changes. Our approach can broaden the scope of possible VT&R applications that require deployment in environments with significant illumination changes.**

I. Introduction

Navigation is a crucial ability for robots to find a safe and suitable path from the starting point to the goal point [1]. Various sensors, including LiDAR and cameras, have been used for this purpose, leading to different solutions. Vision has emerged as a popular sensory modality, and visual navigation for mobile robots has become a widely studied research area [2]. Visual teach & repeat navigation (VT&R) is a framework that uses a camera to guide the robot along a learned trajectory. In the paper, we focus on a particular type of VT&R, which does not require a precise metric map and relies on a convergence theorem to repeat the path [3], [4], [5]. It is simple and reliable, making it suitable for warehouse logistics and inspection applications. However, many visual systems tend to perform poorly in uncontrolled outdoor environments during a long-term deployment [6], [7]. The degraded performance is usually caused by significant appearance variations of the scene due to day&night changes, sun glare or seasonal changes.

Machine learning methods have proven to be effective in addressing the challenges posed by environmental changes [8], [9]. By using pairs of images containing the environment changes to train neural networks, it is possible to extract features (or dense representations of the image) invariant to these changes, which makes them far more effective than hand-crafted features [10], [11]. As a result, these trained feature extractors can be deployed for navigation tasks in uncontrolled outdoor environments with day-to-night and seasonal changes [12].

The usage of RGB images with the learned features for the VT&R system can significantly improve the performance of navigation during long-term deployment [13], but RGB images are shown to be susceptible to adverse illumination conditions, such as sun glare and dark nights. In contrast, infrared images are typically more robust to these disturbances and can perform well in many different illumination conditions [14]. Hence, the incorporation of infrared images into visual stack can possibly help alleviate the problems arising from various illumination changes.

In this paper, we propose a method for training a model to produce a lighting-invariant representation of the image in the IR domain without having a dataset containing these variations. Obtaining image pairs containing environmental changes can be challenging because it usually requires prolonged data collection. In addition, different IR cameras can capture different light spectra, which could make the model bound to usage on a particular device. We show that using only a limited number of RGB-IR image pairs to train a model which can produce representations invariant to environmental changes is possible. This is achieved by inserting an RGB model robust to appearance variations into the contrastive learning pipeline to train the IR model. In addition, we benchmark multiple decision-level fusion methods, which exploit RGB and IR models.

The paper is structured as follows: Firstly, we review related work on infrared and RGB image fusion and contrastive learning. Then, we describe the image-matching pipeline and fusion methods. Further, we present the experimental methodology and results. Finally, we discuss the results and their implications.

II. Related Work

The problems arising from the appearance change of the environment are widely studied in the robot vision field. In this section, we provide a brief review of methods used to address the environmental changes for the navigation of mobile robots. We also shortly discuss appearance-based

* Authors with equal contribution

[1] Department of Electrical and Photonics Engineering, Technical University of Denmark, Anker Engelunds Vej 101, Kongens Lyngby, Denmark s212632@student.dtu.dk

[2] Artificial Intelligence Center, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo náměstí 13, Prague 2, Czechia

Krajnik was funded by the CSF project 20-27034J, Liu by OP VVV MEYS RCI project CZ.02.1.01/0.0/0.0/16_019/0000765 and Rozsypalek by EU H2020 project RoboRoyale no. 964492.

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

VT&R navigation systems and fusion methods in the field of IR and RGB image fusion.

Visual Teach and Repeat navigation (VT&R) is a popular navigation framework that has been deployed on both autonomous ground vehicles and unmanned aerial vehicles (UAV) [15]. In the teaching phase, the system typically stores a map consisting of the action commands and images captured by the robot's front camera during the trajectory traversal. In the repeat phase, the robot replays the velocity commands and uses the map images to determine its position and adjust the heading. The corrections in the heading ensure the robot converges to the original trajectory. The convergence of this framework was mathematically proven by [5]. However, the proof assumes that we have an estimator which can output the correct value of horizontal pixel displacement between the map and live images. The horizontal displacement between images is easier to estimate compared to a 6DoF transformation for localization in metric maps. However, environmental variations can still pose a challenge even for this simplified task.

Many approaches have been applied to address the changing environment. One of the ways is that the robot gathers the experiences during every traversal of a certain path. These experiences can be exploited to adjust the robot's behaviour based on the current state of the environment [16], [17], [18]. The obvious shortcoming is that the robot first needs to observe the current state of the environment, and only after that can it choose the most suitable version of the map to navigate itself. This can be addressed by using the gathered experiences to create a model of the environment and predict its state in given time or conditions [7], [19]. However, all of these methods require gathering the experiences by traversing the path multiple times to ensure robust behaviour. In addition, these experiences are not general and are tied to specific trajectories.

A different approach is having some prior knowledge of the world, usually obtained from large-scale datasets. In general, this knowledge can be used in multiple ways to improve the robustness of navigation. One of the approaches is to traverse the path once and try to generate maps of the same path under different conditions [20]. This reduces the importance of gathering a large number of experiences but choosing the most suitable map to navigate the robot is still necessary. A more straightforward way to exploit the prior knowledge is to train a model that can be used to create a representation of the map, invariant to common changes present in the real world [21].

It has been shown that contrastive learning is suitable for training a model that can capture the structure of provided dataset [22]. The learned representations have better generalization ability and provide more information than features [23]. A Siamese network, which takes two images containing the appearance change as input, can capture similarities and variations in various scenes. The Fully-convolutional Siamese network has been successfully applied in the VT&R [12]. Other research also suggests that the pipelines with dual backbones can be used to transfer some quality of one model to another in a self-supervised manner [24].

It has been shown that infrared images can be more robust to ambient illumination and work well in different lighting conditions [25]. On the other hand, RGB images are high-resolution and provide a considerable amount of detail. Fusing these two types of images can provide significant benefits [14], [26]. Besides, both types of images are easily acquired, and many cameras used in mobile robotics come with both types of lenses. Therefore, IR and RGB image fusion can improve the performance of visual systems for many applications. Several fusion methods operate at the pixel or feature level. Most of them use multi-scale transform [27], [28], sparse representation [29], and neural network [30], [31] strategies to generate high-quality fused images. Decision-level fusion is a straightforward strategy that combines decisions or results made by infrared and RGB images to produce a final output [32]. It has been used for face recognition in different illumination conditions, which shows potential for recognizing places in various illumination conditions [33].

We propose decision-level fusion methods for infrared and RGB images using a backbone neural network trained via self-supervised learning that can improve the performance of the pixel displacement estimation. Specifically, we obtain a dense representation of IR and RGB images, which are then cross-correlated with each other to produce likelihood histograms. These histograms are then combined in different

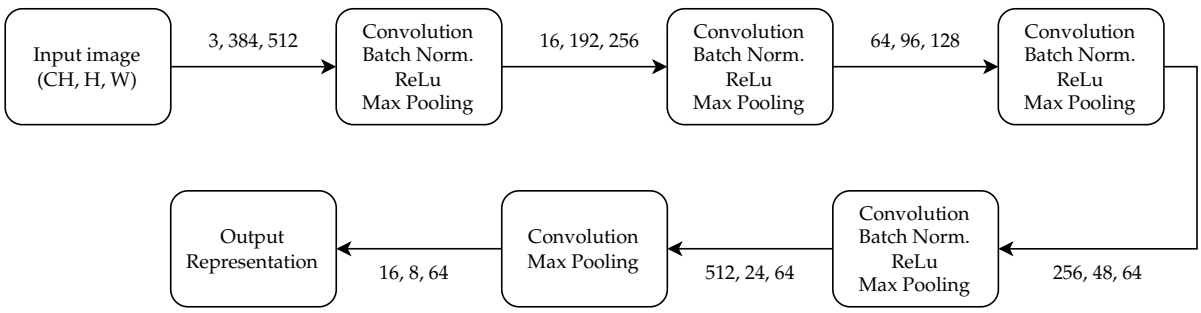


Fig. 1: The architecture of CNNs used in this paper. This particular version is for RGB image since it has 3 input channels. For the IR images is the architecture similar except the input having only one channel. The architecture tailored for mobile robots, thus it is simple and lightweight with only few convolutional layers.

ways to produce the final histogram indicating horizontal displacement. In our experiments, we used a neural network presented in [12]. We do not describe in detail how the CNN_{RGB} was trained, and we treat it as a pretrained model, which can be further exploited for training models with different input types without needing to collect long-term datasets. The architecture of all the CNNs is shown in Figure 1.

III. Method Description

This section explores the various fusion methods for correlating different types of representations and fusing their corresponding likelihood histograms. Specifically, we will delve into four methods: a baseline method without fusion utilizing only RGB images and three fusion methods using both RGB and IR images.

For our evaluation purposes, we collected pairs of IR and RGB images which serve as our source images, denoted as I_s^{IR} and I_s^{RGB}, respectively, and which comprise our map. Additionally, we collected other pairs of images taken at the exact locations but under different lighting conditions, referred to as target images, denoted as I_t^{IR} and I_t^{RGB}. We aim to use a CNN to produce image representations and calculate their horizontal displacements Δp. We benchmark the performance of the methods on collected datasets similarly as in [21], [12]. It has been shown that this methodology is suitable for evaluating the performance of the displacement estimator for the VT&R navigation.

A. Non-fusion method with single RGB images

In this method, one public network [12], trained on RGB images, is used to process two images input. As shown in Fig. 2, the Siameses-RGB pipeline processes the input images, I_s^{RGB} and I_t^{RGB}, and outputs corresponding representations. The target representation is shifted along the source representation and calculates the similarity, i.e., likelihood over all possible displacements using cross-correlation. One histogram, likelihood versus displacement, is generated based on these data and indicates the most likely displacement. The process can be written as:

$$\mathcal{L}(\Delta p|I_s^{RGB}, I_t^{RGB}) = C(f_{RGB}(I_s^{RGB})) \star f_{RGB}(I_t^{RGB}), \tag{1}$$

where I_s^{RGB} is the input source RGB image. f_{RGB} is the function of CNN in Siamese-RGB, which outputs the neural representation of one RGB input. Then, the source representation is circular pad C and cross-correlated with target representation to compute the likelihood histogram $\mathcal{L}$.

B. Neural network training

The f_{RGB} network, as described in [12], is designed to generate representations invariant to lighting and seasonal environmental changes. In this paper, we aim to train a second network that takes a different input of a similar modality (in our case, an IR image) and produces representations suitable for decision-level fusion. To achieve this, we utilize a contrastive learning pipeline similar to the original method but with a few modifications.

Firstly, we use two networks in our pipeline - the pretrained f_{RGB} network with fixed weights and a randomly initialized f_{IR} network that we train in the process. Secondly, we only use pairs of IR and RGB images obtained simultaneously during data gathering in our training set. The training pipeline is depicted in Fig. 3. The loss function used for training is binary cross-entropy, and the target is constructed on the fly based on the position of the random crop of the IR image.

The main idea behind this approach is that the lighting invariant representations learned by the f_{RGB} network can be utilized by the second network, f_{IR}, for fusion. The advantage of this method is that it eliminates the need to collect IR images with different lighting conditions. Instead, we only require a dataset with corresponding RGB and IR images. The f_{IR} network can extract the invariant properties of the representations from the f_{RGB} network.

C. Fusion methods

We have two neural networks outputting lighting and seasonal invariant representations R of RGB and IR images. The network f_{RGB} is taken from [12], and the f_{IR} is trained via the pipeline presented in Figure 3.

$$R^{RGB} = f_{RGB}(I^{RGB}) \tag{2}$$

$$R^{IR} = f_{IR}(I^{IR}). \tag{3}$$

In equation 1, only one type of input for map and the live image was used, but now we can similarly process the IR image and use it to improve the results. The following subsections present multiple ways to exploit IR representations.

1) IR-IR, RGB-RGB multiplication: First, the representations of the same input type are cross-correlated:

$$\mathcal{L}_{IR} = \mathcal{L}(\Delta p|I_s^{IR}, I_t^{IR}) \tag{4}$$

$$\mathcal{L}_{RGB} = \mathcal{L}(\Delta p|I_s^{RGB}, I_t^{RGB}). \tag{5}$$

The likelihood of all displacements is then calculated as the element-wise product of the individual histograms:

$$\mathcal{L}_{fusion1} = \mathcal{L}_{IR} \cdot \mathcal{L}_{RGB}. \tag{6}$$

2) IR-RGB concatenation: In this method, we concatenate the representations of RGB and IR images along the channel dimension and perform cross-correlation using this concatenated representation. The method can be written down as follows:

$$\mathcal{L}_{fusion2} = C([R_s^{RGB}, R_s^{IR}]) \star [R_t^{RGB}, R_t^{IR}]. \tag{7}$$

3) IR-RGB, IR-IR, RGB-RGB cross multiplication: The last fusion method is similar to the first one but exploits the similarity between IR and RGB representations. We first define this cross-domain likelihood:

$$\mathcal{L}_{cross} = \mathcal{L}(\Delta p|I_s^{IR}, I_t^{RGB}) \cdot \mathcal{L}(\Delta p|I_s^{RGB}, I_t^{IR}). \tag{8}$$

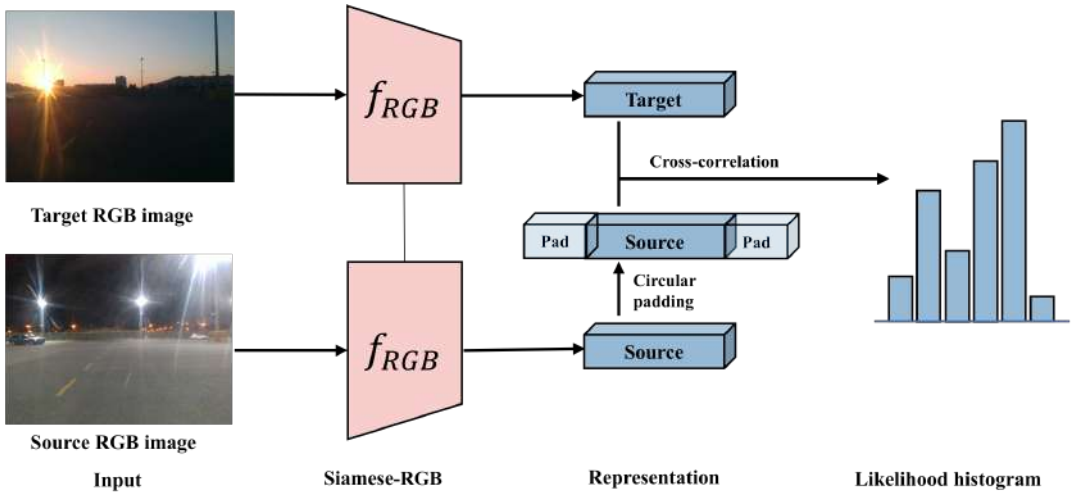


Fig. 2: Diagram of the architecture used for displacement estimation. The estimator is a Fully-convolutional Siamese neural network, which has two images as inputs and outputs histogram corresponding to the likelihoods of possible displacements. The image shows the method without any fusion - both branches of the network have CNN with the same weights, and the inputs are only in the RGB domain.

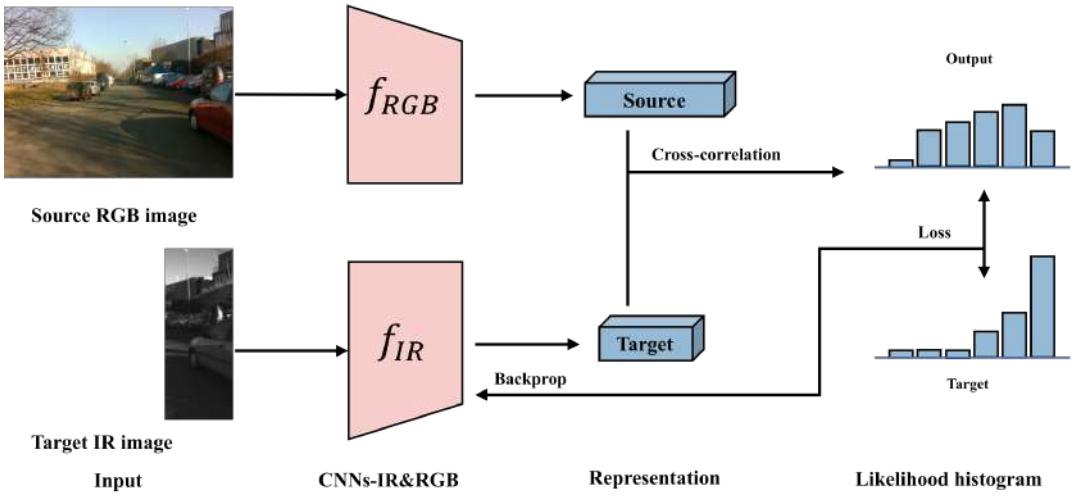


Fig. 3: Diagram of the training pipeline for the infrared model. The CNNs do not share weights and have different input types. During training, the existing model CNN_{RGB} has fixed weights and only the part with the new model CNN_{IR} is updated. The target vector is constructed on fly based on the known position of random crop during training.e This newly obtained model is then used for fusion.

Using this notation, the last fusion method can be written as:

$$\mathcal{L}_{fusion3} = \mathcal{L}_{IR} \cdot \mathcal{L}_{RGB} \cdot \mathcal{L}_{cross}. \tag{9}$$

This method is computationally more expensive but exploits the most information about the scene in both visible and IR spectra.

D. Final displacement estimation

The output histograms from these four methods can be used to find the horizontal displacement directly. We denote the final displacement between source (map) images and target (live) images for method M as δp_M. The pixel displacement is outputted as:

$$\delta p_M = argmax(\mathcal{L}_M), \tag{10}$$

where $\mathcal{L}_M$ denotes the histogram obtained by one of the fusion methods or the original method without fusion.

IV. Experiment

In this section, we first describe the dataset construction, experimental setup and its details. Then, the results of comparative experiments are presented to demonstrate the performance of the proposed methods.

A. Datasets

To show the viability of our method, we use two different datasets. One dataset is used to train the neural network

processing the IR images, and the second is used to evaluate various fusion methods. All the images are resized to 512×384 pixels. The training dataset comprises 5000 pairs of corresponding IR and RGB images captured using Intel Realsense D435i. We scheduled the data collection to take the images in various lighting conditions (sun glare, night). Note that this camera has slightly different placement (a few centimetres) of RGB and IR sensors, which results in slightly different pixel positions of close objects. The CNN squeezes the width by a factor of eight, which makes the embeddings robust to these minor inaccuracies.

The second dataset was captured by a robot operating at a parking lot next to the Škoda factory. The robot traverses the same path many times during the day while collecting the rosbags. The robot is equipped with the same camera type as we used to collect the training data. Similarly, the camera simultaneously captures the IR and RGB images as in the test data. To emulate the VT&R position uncertainty, only the odometry of the robot is used to extract images every one meter. The dataset is constructed from 15 traversals, yielding 1725 pairs of IR and RGB images (920 pairs in normal daylight conditions, 460 pairs with sun glare, and 345 pairs at night). Finally, we form pairs from traversals with drastically different environmental conditions to create quadruples of IR and RGB images taken at the same locations, resulting in three datasets - sun glare vs daylight, daylight vs night and sun glare vs night. Some examples of the quadruples are shown in Figures 4 and 5.

B. Experimental setup

To evaluate our proposed method, we conducted experiments comparing all the methods using the Škoda dataset. We tested the ability to find horizontal displacement Δp. The metrics to evaluate the quality of the estimator are absolute error (AE) and standard deviation (SD). The AE represents the absolute difference between the calculated displacement and the ground truth in pixels. AE is used to evaluate the methods' performance, while SD is used to evaluate the stability of the methods' performance. In addition, we used the pairwise T-test to show that the performance improvement of the fusion method over the non-fusion method is statistically significant.

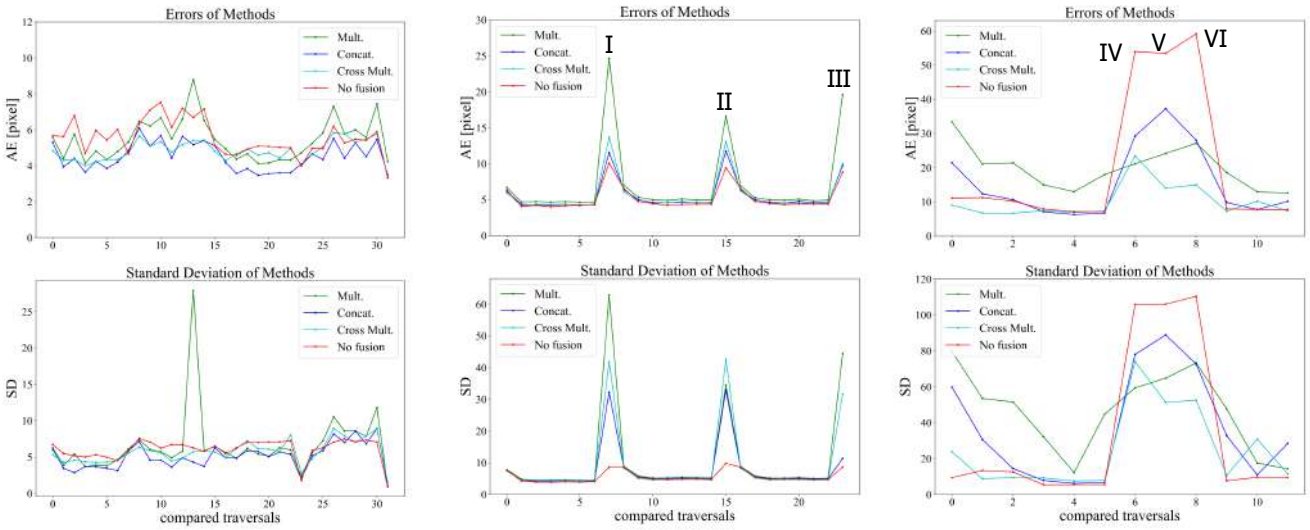
C. Experimental results

Results of the experiments are visualised in Figures 4(a), 4(b) and 4(c). The quantitative evaluation of the methods is done in Table I. It is visible that without fusion, the sun glare can significantly reduce the performance of displacement estimation. We show that all presented fusion methods that use both the IR and RGB images can reduce the influence of sun glare on the quality of the estimate. It is also visible that there is little to no performance decrease for other challenging scenarios, such as day&night difference. This is also supported by presented p-values, which show that for the scenarios with sun glare, there is a statistically significant difference between expected error, while for the day&night scenarios, the p-values suggest that the difference in the performance is rather insignificant. Note that there is a theoretical limit for the method's precision (≈ 4 pixels) arising from different sizes of CNN's input and output. Even though the scenarios are pretty challenging, the results for the two of them are close to the capabilities of the estimator. It is also possible that the ground truth annotations contain inaccuracies in terms of a few pixels because the lenses for IR and RGB are not precisely in the same position. That is one of the reasons why we perform statistical tests, which further support our claims about the performance increase.

Overall, cross multiplication performs best in our experiments and significantly outperforms all the methods in the sun glare vs night scenario. We believe that in this particular scenario, the Sun glare and reflections can be interpreted as bright objects in a night image, and the term $\mathcal{L}_{croos}$ in equation 9 can help to resolve this ambiguity. However, all the fusion methods help alleviate the issues arising from sun glare thanks to the additional information in the IR image. The neural networks used for creating the representations from the image are relatively shallow and lightweight for easy deployment on a mobile robot. In terms of computational requirements, the multiplication and concatenation are comparable. In contrast, cross multiplication is the most demanding due to the term $\mathcal{L}_{cross}$, which requires the calculation of two additional cross-correlations of the image representations. This overhead is not significant because most of the computation time ($\approx 90\%$) takes the forward pass of CNN.

TABLE I: This table shows the performance of methods in three changing illumination conditions. We report the average error (MAE) in pixels, standard deviation (SD) for all methods and the p-value of pairwise T-test for the presented fusion methods.

Method	sun glare & daylight			dark night & daylight			sun glare & dark night		
	MAE	**SD**	**p-value**	**MAE**	**SD**	**p-value**	**MAE**	**SD**	**p-value**
No fusion	5.57	± 6.06	-	**5.25**	± **5.59**	-	20.39	± 33.39	-
Multiplication	5.46	± 6.58	0.27	7.08	± 10.77	10^{-8}	19.87	± 45.91	0.78
Concatenation	**4.53**	± **5.07**	10^{-97}	5.57	± 7.81	0.09	15.58	± 36.43	0.003
Cross Multiplication	4.84	± 5.68	10^{-25}	5.64	± 9.30	0.13	**10.11**	± **24.78**	10^{-12}



(a) sun glare vs daylight (b) daylight vs night (c) sun glare vs night

Fig. 4: Horizontal displacement estimation quality of the compared methods. The mean absolute errors in pixels are in the top row and standard deviations are in the bottom. The x-axes of the graphs indicate the index of the traversal pair. The Roman numbers I-VI indicate particularly difficult conditions, shown in Figure 5.

(a) Daylight versus night. These three pairs of sample images correspond to the peaks in Figure 4(b), where the cross multiplication method shares similar performance with the non-fusion method in displacement estimation.

(b) Sun glare versus night. These three pairs of sample images correspond to the peaks in Figure 4(c), where the fusion methods significantly improve the displacement estimation.

Fig. 5: Example image pairs captured in particularly difficult conditions indicated by Roman numbers in Figure 4.

V. Discussion

We hypothesize that the existing model f_{RGB} can be used to train a new neural network using inputs obtained from different sensors (IR, depth, event-based) with the same modality (image). One of the contributions of this paper is the training scheme, which enables the incorporation of different sensory inputs into the VT&R pipeline without the necessity of collecting long-term datasets.

The significant advantage of the training scheme is that the robustness to the environment changes can be translated from the existing model f_{RGB} to the new model, which has a different type of input. We demonstrate the feasibility of this process in the domain of IR images. The collected training dataset only contains information on bridging the RGB and IR domains. However, the trained network f_{IR} still shows robustness to significant changes in the scene appearance and can improve the performance of the displacement estimate.

VI. Conclusion

We present a self-supervised learning pipeline to train a model for infrared images, which is robust to environmental changes, without the necessity of collecting long-term datasets. Further, we benchmark three methods that can fuse the IR model with the existing RGB model and improve the accuracy of horizontal displacement estimation, which is a crucial subtask of appearance-based VT&R. We test the performance of presented fusion methods on three challenging datasets. We show that the fusion method can outperform the original method without fusion with high statistical significance. Our method can be applied to improve the robustness of VT&R frameworks to drastic illumination changes.

In the future, we will investigate deployment of similar learning pipelines in scenarios, where miniature bio-hybrid robots with vision-only sensors operate in environments with adverse visibility [34]. These scenarios include not only search and rescue missions [35], but also robot navigation in and exploration of social insect colonies [36], [37].

References

[1] H. Choset *et al.*, *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[2] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *Journal of intelligent and robotic systems*, 2008.

[3] T. Krajník *et al.*, "Simple yet stable bearing-only navigation," *Journal of Field Robotics*, vol. 27, no. 5, 2010.

[4] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of field robotics*, vol. 27, no. 5, 2010.

[5] T. Krajník, F. Majer, L. Halodová, and T. Vintr, "Navigation without localisation: reliable teach and repeat based on the convergence theorem," in *Int. Conf. on Int. Robots and Systems (IROS)*, 2018.

[6] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE T-RO*, 2016.

[7] T. Krajník *et al.*, "Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments," *IEEE T-RO*, 2017.

[8] J. Spencer *et al.*, "Same features, different day: Weakly supervised feature learning for seasonal invariance," in *CVPR*, 2020.

[9] G. Broughton *et al.*, "Robust image alignment for outdoor teach-and-repeat navigation," in *ECMR*, 2021.

[10] Z. Rozsypálek *et al.*, "Semi-supervised learning for image alignment in teach and repeat navigation," in *37th ACM/SIGAPP Symposium on Applied Computing*, 2022, pp. 731–738.

[11] P. Khosla *et al.*, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18 661–18 673, 2020.

[12] Z. Rozsypálek *et al.*, "Contrastive learning for image registration in visual teach and repeat navigation," *Sensors*, 2022.

[13] Y. Chen and T. D. Barfoot, "Self-supervised feature learning for long-term metric visual localization," *IEEE RAL*, 2023.

[14] J. Ma, Y. Ma, and C. Li, "Infrared and visible image fusion methods and applications: A survey," *Information Fusion*, 2019.

[15] M. Warren, M. Paton, K. MacTavish, A. P. Schoellig, and T. D. Barfoot, "Towards visual teach and repeat for gps-denied flight of a fixed-wing uav," in *Field and Service Robotics: Results of the 11th International Conference*. Springer, 2018, pp. 481–498.

[16] F. Dayoub and T. Duckett, "An adaptive appearance-based map for long-term topological localization of mobile robots," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3364–3369.

[17] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1645–1661, 2013.

[18] D. M. Rosen, J. Mason, and J. J. Leonard, "Towards lifelong feature-based mapping in semi-static environments," in *ICRA*, 2016.

[19] T. Krajník, T. Vintr, S. Molina, J. Fentanes, G. Cielniak, O. Mozos, G. Broughton, and T. Duckett, "Warped hypertime representations for long-term autonomy of mobile robots," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, 07 2019.

[20] A. Anoosheh, T. Sattler, R. Timofte, M. Pollefeys, and L. Van Gool, "Night-to-day image translation for retrieval-based localization," 05 2019, pp. 5958–5964.

[21] T. Krajník *et al.*, "Image features for visual teach-and-repeat navigation in changing environments," *Robotics and Auton. Systems*, 2017.

[22] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020.

[23] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[24] S. Gidaris *et al.*, "Learning representations by predicting bags of visual words," in *CVPR*, 2021.

[25] L. Tang, X. Xiang, H. Zhang, M. Gong, and J. Ma, "Divfusion: Darkness-free infrared and visible image fusion," *Information Fusion*, 2023.

[26] X. Zhang, P. Ye, H. Leung, K. Gong, and G. Xiao, "Object fusion tracking based on visible and infrared images: A comprehensive review," *Information Fusion*, vol. 63, pp. 166–187, 2020.

[27] Z. Zhang and R. S. Blum, "A categorization of multiscale-decomposition-based image fusion schemes with a performance study for a digital camera application," *Proceedings of the IEEE*, vol. 87, no. 8, pp. 1315–1326, 1999.

[28] S. Li, B. Yang, and J. Hu, "Performance comparison of different multi-resolution transforms for image fusion," *Information Fusion*, vol. 12, no. 2, pp. 74–84, 2011.

[29] S. Li, H. Yin, and L. Fang, "Group-sparse representation with dictionary learning for medical image denoising and fusion," *IEEE Transactions on biomedical engineering*, 2012.

[30] T. Xiang, L. Yan, and R. Gao, "A fusion algorithm for infrared and visible images based on adaptive dual-channel unit-linking pcnn in nsct domain," *Infrared Physics & Technology*, vol. 69, pp. 53–61, 2015.

[31] W. Kong, L. Zhang, and Y. Lei, "Novel fusion method for visible light and infrared images based on nsst–sf–pcnn," *Infrared Physics & Technology*, vol. 65, pp. 103–112, 2014.

[32] A. Sinha, H. Chen, D. Danu, T. Kirubarajan, and M. Farooq, "Estimation and decision fusion: A survey," *Neurocomputing*, 2008.

[33] Y. Zhao, Y. Yin, and D. Fu, "Decision-level fusion of infrared and visible images for face recognition," in *2008 Chinese Control and Decision Conference*. IEEE, 2008, pp. 2411–2414.

[34] H. D. Nguyen, V. T. Dung, H. Sato, and T. T. Vo-Doan, "Efficient autonomous navigation for terrestrial insect-machine hybrid systems," *Sensors and Actuators B: Chemical*, 2023.

[35] T. Rouček *et al.*, "Darpa subterranean challenge: Multi-robotic exploration of underground environments," in *Modelling and Simulation for Autonomous Systems*. Springer, 2020.

[36] N. R. Franks, J. A. Podesta, E. C. Jarvis, A. Worley, and A. B. Sendova-Franks, "Robotic communication with ants," *Journal of Experimental Biology*, 2022.

[37] M. Stefanec *et al.*, "A minimally invasive approach towards "ecosystem hacking" with honeybees," *Frontiers in Robotics and AI*, 2022.

White-box and Black-box Adversarial Attacks to Obstacle Avoidance in Mobile Robots

Iñaki Rañó Anders Lyhne Christensen

***Abstract*— Advances in artificial intelligence (AI) play a major role in the adoption of robots for an increasingly broader range of tasks. However, as recent research has shown, AI systems, such as deep-learning models, can be vulnerable to adversarial attacks where small but carefully crafted changes to a model's input can severely compromise its performance. In this paper, we present two methods to find adversarial attacks against autonomous robots. We focus on external attacks against obstacle-avoidance behaviour where an attacker — a robot — actively perturbs the sensor readings of a goal-seeking victim robot. In the first (white-box) method, we model the interaction between the victim and attacker as a dynamical system and generate a series of open-loop control signals for the attacker to alter the victim's behaviour. In the second (black-box) method, the assumption that the attacker has full knowledge of the system's dynamics is relaxed, and closed-loop control for the attacker is learnt through reinforcement learning. We find that both methods are able to find successful attacks against the victim robot and thus constitute viable techniques to assess the robustness of autonomous robot behaviour.**

I. Introduction

The increased use of robots poses a significant challenge in terms of security and privacy [1], [2]. Security includes resilience against malicious actors who aim to disrupt the robots' operation or perniciously change their behaviour. Adversarial attacks have already been successfully demonstrated against deep learning models [3], [4] and other learning mechanism [5]. Significant research efforts are currently being devoted to hardening such models to make them robust against adversarial attacks [6]. However, as we show in this paper, robots relying on well established, non-learned control can be equally vulnerable to adversarial attacks. We present two methodologies to generate adversarial attacks to robot behaviour than can be implemented by a second physical robot that interferes with the sensors of the first. We show that it is possible to control the behaviour of a robot, *the victim*, by physically interfering with its sensors through an attacking robot, *the attacker*. The first method is an alternative approach to optimal control that uses only initial conditions instead of boundary conditions, making the search for an attack simpler. It can generate attacker's behaviour fast and provides an empirical way to quickly find whether an attack is possible, however, its applicability is limited by the assumptions made. On the other hand, as we will show, attacks can also be generated using reinforcement learning which leads to more general attacker behaviour at the expense of longer training time.

In this study, the victim robot implements obstacle avoidance via potential field methods (PFM) [7], a well established mechanism with well known limitations [8]. Although many excellent alternatives to obstacle avoidance exist [9], [10], [11], [12], the mathematical formalisation of the state transition equation to generate a dynamic model of obstacle avoidance is outside the scope of this work, hence, for simplicity, we will stick to the PFMs. In our experimental scenarios, the victim of the attack performs obstacle avoidance in a static environment trying to reach its predefined goal location. The attacker robot is situated in the same environment and must force the victim to a false target area by occupying positions that affects the victim's behaviour. In the reminder of this paper, *goal* refers to the victim's predefined goal location, whereas *target* refers to the attacker's desired destination for the victim. Our experimental results show that adversarial attacks to PFM for obstacle avoidance are possible in mobile robots, and we present two alternatives to generating these attacks, namely using (i) an open-loop strategy, and (ii) a closed-loop non-linear controller.

The rest of the paper is organised as follows. Section II presents the underlying assumptions and the two methodologies for an attacking robot to learn how to move in order to change the victim's trajectory so that it reaches the attacker's target. We focus on attacking PFM for obstacle avoidance but both methodologies can be applied to other behaviours. The simulated results for the two methodologies are presented in a series of scenarios in Section III. Finally, Section IV presents our conclusions and future research directions.

II. Attacks to Robot Behaviour

This section presents two methods to generate adversarial attacks to robot behaviour. We cast the attack design as an optimisation problem where the objective of the attacker is to optimise a function of the trajectory of the victim and the target. Specifically, the objective of the attacker is to drive the victim to a predefined target area instead of the victim's own goal location. The first is a white-box approach, Section II-A, which leads to an open-loop control signal for the attacker, while the second is a black-box approach, Section II-B, which builds a non-linear controller using an extended state of the victim-attacker system.

A. White-box open-loop attacks

This section begins with a general formulation of the problem of attacking robot behaviour modelled as a dynamical

I. Rañó is with the Electronics and Computing Dept. of the University of Santiago de Compostela, Spain, ignacio.rano@usc.es. A.L. Christensen is with the SDU Biorobotics Unit of the University of Southern Denmark, Denmark. e-mail: andc@mmmi.sdu.dk.
979-8-3503-0704-7/23/$31.00

system, and then moves on to instantiate attacks against PFM for obstacle avoidance. Let us denote the victim's state as $\mathbf{x}_v$ and assume the evolution of its state is given by a C^1 class vector function which includes the state of the attacker as an independent variable. We denote the attacker's state by $\mathbf{x}_a$ and assume its dynamics can be shaped by a control signal $u(t)$. We will further assume the attacker has access to the victim's state. Under these assumptions the dynamic behaviour of the victim and the attacker can be stated as:

$$\begin{aligned}\dot{\mathbf{x}}_v &= \mathbf{F}_v(\mathbf{x}_v, \mathbf{x}_a)\\ \dot{\mathbf{x}}_a &= \mathbf{F}_a(\mathbf{x}_a, \mathbf{x}_v, \mathbf{u}(t)),\end{aligned} \tag{1}$$

where $\mathbf{F}_v(\cdot)$ and $\mathbf{F}_a(\cdot)$ are, respectively, the functions defining the dynamics of the victim and the attacker. It is worth noting that the dynamics of the victim must depend on the state of the attacker so that it can influence the victim's dynamics. Furthermore, we will assume the victim is unaware of the attack so that it takes no countermeasure. The dynamics of the attacker will depend on its own state and the state of the victim, which we assumed known to the attacker. If we consider that the control input $u(t)$ is parameterised through a vector Φ, i.e. $u(t) = u(t, \Phi)$, and assume that the attack starts at $t = 0$ with a horizon t_f, we can define the following error function:

$$E(\Phi) = \frac{1}{2}\int_0^{t_f} |\mathbf{t}_a - \mathbf{x}_v(t)|^2 dt, \tag{2}$$

where $\mathbf{t}_a$ is the centre of the target area to which the attacker should drive the victim. The error depends on the parameters through the system's dynamics (1) and the problem is to find $\mathbf{\Phi}^* = \arg\min_{\Phi} E(\Phi)$, i.e. the optimal parameters of the attacker's input which minimises the error subject to the constraints defined by the dynamics of the system eq. (1). Although this can be achieved through optimal control [13], we approach the problem as a direct minimisation of $E(\Phi)$ using a gradient descent algorithm. Such an approach can be also used to learn the dynamics of the victim if $\mathbf{F}_v(\cdot)$ is known up to a set of parameters [14]. The gradient of the error (2) w.r.t. the parameters Φ is:

$$\nabla_\Phi E(\Phi) = -\int_0^{t_f} (\mathbf{t}_a - \mathbf{x}_v(t)) \nabla_\Phi \mathbf{x}_v(t) dt, \tag{3}$$

where $\nabla_\Phi E(\Phi)$ and $\nabla_\Phi \mathbf{x}_v(t)$ are the gradients of the error and the victim's state w.r.t. the parameters (Φ) of the control input of the attacker.

The challenge to calculate the gradient, eq. (3), is to obtain $\nabla_\Phi x_v(t)$ as the way the victim's trajectory changes with Φ is not known. However, if we calculate the derivatives of eqs. (1) w.r.t. Φ we get:

$$\begin{aligned}\nabla_\Phi \dot{\mathbf{x}}_v &= \nabla_{\mathbf{x}_v}\mathbf{F}_v \nabla_\Phi \mathbf{x}_v + \nabla_{\mathbf{x}_a}\mathbf{F}_v \nabla_\Phi \mathbf{x}_a\\ \nabla_\Phi \dot{\mathbf{x}}_a &= \nabla_{\mathbf{x}_a}\mathbf{F}_a \nabla_\Phi \mathbf{x}_a + \nabla_{\mathbf{x}_v}\mathbf{F}_a \nabla_\Phi \mathbf{x}_v + \nabla_\Phi \mathbf{F}_a,\end{aligned} \tag{4}$$

where $\nabla_{\mathbf{x}_v}\mathbf{F}_v$ and $\nabla_{\mathbf{x}_a}\mathbf{F}_v$ are the Jacobians of the victim's dynamics w.r.t. the victim and attacker's states, $\nabla_{\mathbf{x}_a}\mathbf{F}_a$ and $\nabla_{\mathbf{x}_v}\mathbf{F}_a$ are the Jacobians of the attacker's dynamics w.r.t. the attacker and victim's state, and $\nabla_\Phi \mathbf{F}_a$ is the Jacobian of the attacker's dynamics w.r.t. the parameters of the input $\mathbf{u}(t, \Phi)$. If the derivatives of the state variables $\dot{\mathbf{x}}_v$ and $\dot{\mathbf{x}}_a$ change smoothly with time (t) and parameters (Φ), we can exchange the order of the gradient and time derivative in eqs. (4). Then defining the matrices $D_v = \nabla_\Phi \dot{\mathbf{x}}_v$ and $D_a = \nabla_\Phi \dot{\mathbf{x}}_a$ we can rewrite equation (4) as:

$$\begin{aligned}\dot{D}_v &= \nabla_{\mathbf{x}_v}\mathbf{F}_v D_v + \nabla_{\mathbf{x}_a}\mathbf{F}_v D_a\\ \dot{D}_a &= \nabla_{\mathbf{x}_a}\mathbf{F}_a D_a + \nabla_{\mathbf{x}_v}\mathbf{F}_a D_v + \nabla_\Phi \mathbf{F}_a\end{aligned} \tag{5}$$

Simultaneously integrating eqs. (1) and (5), we obtain the gradient $\nabla_\Phi \mathbf{x}_v(t) = D_v$ needed to find $\nabla_\Phi E$, and hence optimise the error (2) using gradient descent. In this way, the dynamic constraints of the system are accounted for through the integration of (5) with zero initial conditions as the system's initial state does not depend on the parameters.

To instantiate the formulation above to a PFM for obstacle avoidance, in the rest of this section we assume both robots (attacker and victim) operate in 2D according to a single integrator model ($\dot{\mathbf{p}} = \mathbf{u}$). Let's assume the victim starts at position $\mathbf{p}_v(0) \in \Re^2$ and has goal $\mathbf{t}_v \in \Re^2$ in an environment containing obstacles, while the attacker starts at position $\mathbf{p}_a(0) \in \Re^2$. Obviously these points should not be inside any obstacle grown with the Minkowski sum of their bodies, nor should they be so close to one another that the robots' bodies overlap. Both robots have to perform obstacle avoidance and, since both follow the single integrator model, we assume the dynamics of the system is given by:

$$\begin{aligned}\dot{\mathbf{p}}_v &= \mathbf{F}_v^o(\mathbf{p}_v, \mathbf{p}_a) + \mathbf{F}_v^g(\mathbf{p}_v)\\ \dot{\mathbf{p}}_a &= \mathbf{F}_a^o(\mathbf{p}_a) + \mathbf{F}_a^g(\mathbf{p}_a, \mathbf{p}_v) + \mathbf{u}(\Phi),\end{aligned} \tag{6}$$

where $\mathbf{F}_v^o(\cdot)$ and $\mathbf{F}_a^o(\cdot)$ are the repulsive forces of the obstacles for the victim and attacker, respectively, $\mathbf{F}_v^g(\cdot)$ and $\mathbf{F}_a^g(\cdot)$ are the attractive forces of the goal for the victim and the attacker, and $\mathbf{u}(\Phi)$ is the parametric input to the attacker dynamics. While the victim has its obstacle avoidance goal, we can set the goal of the attacker to be the current position of the victim. In general, if the input $\mathbf{u}(\Phi)$ is not bounded, the attacker can generate large actions attack the victim potentially leading to collisions with obstacles. However, one can select a maximum action u_M smaller than the maximum norm of the contributions of the obstacles making the trajectories of the attacker obstacle free, i.e. imposing a constraint on the maximum allowed input $\mathbf{u}(\Phi)$. We will rely on this approach also in our second mechanism to generate adversarial attacks.

B. Black-box Attacks via Reinforcement Learning

Although the approach presented in Section II-A can find an input to drive the victim to the attacker's target, it entails an assumption unrealistic in practice, namely that the attacker knows the dynamics of the victim $\mathbf{F}(\cdot)$. To relax this assumption, attacks can be generated through reinforcement learning (RL) [15] where the additional input to the attacker's dynamics $\mathbf{u}$, the policy in this case, depends on the states of the attacker ($\mathbf{x}_a$) and the victim ($\mathbf{x}_v$). For the obstacle avoidance attacks, we created an RL state $\mathbf{s}_t$ with

the positions of the victim and the attacker's target relative to the attacker's position and also included the velocity of the victim, which can be estimated from the sequence of position $\mathbf{p}_v$. The dynamics of the system then becomes:

$$\begin{aligned}\dot{\mathbf{p}}_v &= \mathbf{F}_v^o(\mathbf{p}_v, \mathbf{p}_a) + \mathbf{F}_v^g(\mathbf{p}_v) \\ \dot{\mathbf{p}}_a &= \mathbf{F}_a^o(\mathbf{p}_a) + \mathbf{F}_a^g(\mathbf{p}_a, \mathbf{p}_v) + \mathbf{u}(\mathbf{s}_t),\end{aligned} \tag{7}$$

where $\mathbf{s}_t = [\mathbf{p}_v - \mathbf{p}_a, \mathbf{t}_a - \mathbf{p}_a, \mathbf{v}_v]$. Although the RL problem can be defined with a state $\mathbf{s}_t$ that does not include the victim's velocity, our tests showed that including this information greatly helped to speed up the learning process, while using $\mathbf{v}_v$ does not necessarily imply knowledge about the victim's dynamics. Furthermore, the dynamics of the attacker could be fully learnt, i.e. without superposing the obstacle avoidance mechanism, but that would entail including some type of range sensing in the attacker as part of the state ($\mathbf{s}_t$). This in turn means a more complex learning mechanism with more inputs and more training data since the attacker would have to learn to avoid obstacles while performing the attack. Given that the attacker dynamics already includes a term for obstacle avoidance, the reward function to optimise in the RL problem was simply defined to be a function of the distance from the victim to the target position of the attacker:

$$r(\mathbf{s}_t) = \begin{cases} r_M & \text{if } |\mathbf{p}_v - \mathbf{t}_a| \leq \epsilon \\ -\alpha|\mathbf{p}_v - \mathbf{t}_a| & \text{if } |\mathbf{p}_v - \mathbf{t}_a| > \epsilon, \end{cases} \tag{8}$$

where ϵ defines an area around the attacker's target, $r_M > 0$ is a positive reward and α is a scaling factor. This reward function penalises with a negative reward the distance between the victim and the attacker's target and gives a positive reward when the victim enters a region of radius ϵ centred at $\mathbf{t}$, which also indicates the end of the RL episode.

III. Experiments of Adversarial Attacks

In this section, we show results of simulated experiments with the two approaches presented in Section II. The attacker and the victim run potential field-based obstacle avoidance with different parameters, specifically the attacker parameters allow it to move faster and to get closer to obstacles. Both the victim and the attacker can perceive the locations and sizes of nearby obstacles.

A. Open-Loop Attacks

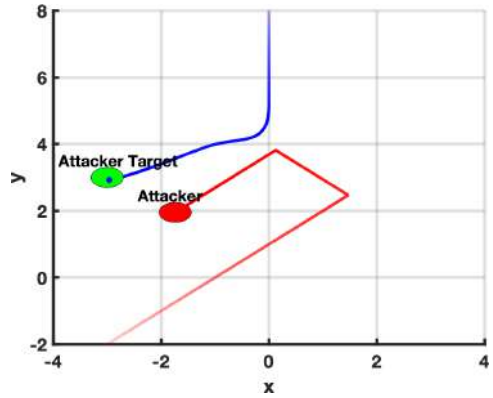
This section presents simulations where the attacker computes an open-loop control signal to combine with the PFM to drive the victim towards a target area using the methodology presented in Section II-A. The equations to compute the gradient of the error (3), i.e. equations (1) and (5) were integrated simultaneously using Euler's method with a step size of $\Delta t = 0.05$. The parameters Φ used are the discretised control signal of the attacker $\mathbf{u}(t)$ as a piece-wise constant function, $u(t_k)$, which is initially set to zero, i.e. $\Phi = \mathbf{0}$ as initial guess for the control commands. Initialising Φ to a random vector would likely not bring any benefit, while other initialisations could improve the optimisation but are difficult to guess. In all simulations shown in this section, the starting position of the victim was $\mathbf{x}_v(0) = [0, 8]$ and its goal was $\mathbf{t}_v = [0, -8]$. The maximum speed of the victim was set to 1m/s, while the attacker had a maximum speed 20% higher, which we found experimentally to give the attacker enough time to approach and drive the victim towards the attacker's target.

Figure 1 shows the trajectories of the victim and the attacker for a configuration where the initial position of the attacker was $\mathbf{x}_a(0) = [-3, -2]$ and the target to drive the victim to is $\mathbf{t}_a = [-3, 3]$. Figure 1(a) shows the resulting attack in a scenario without obstacles, where the red and blue trajectories corresponds to the attacker and the victim, respectively. The target is represented by a green circle while the final position of the attacker corresponds to the red circle. The transparency level of the trajectories represents the time evolution, i.e. the beginning of the trajectory has a higher transparency and same level of transparency in the two trajectories corresponds to same time step. As it can be seen in the figure, the optimisation process found a temporal sequence of attacker velocities to drive the victim to the target area, and the whole trajectory lasted around 9 seconds. The trajectory (see Figure 1(a)) is just a sequence of straight lines driving the victim to the attacker's target.

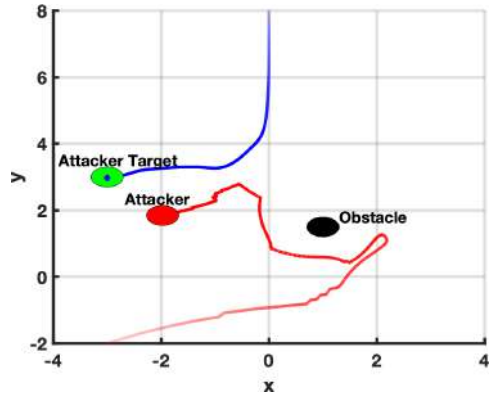
For the next scenario we placed one obstacle in the attacker's way. The resulting trajectory is shown in Figure 1(b), and despite the presence of the obstacle, the attacker is able to drive the victim to its target, although in this case, the attacker required 12 seconds to complete the attack. Figures 1(c) and 1(d) show the result of attacking trajectories for the scenario with two and three obstacles, respectively, where obstacles were added in new positions of the arena to further disturb the trajectories of the attacker and the victim. While the second obstacle affects the trajectory of the attacker, the third obstacle only affects the trajectory of the victim, yet in both cases the attacker successfully finds appropriate trajectories to drive the victim to its target. The time it took to complete the attack was 13 and 11 seconds, respectively. All the time horizons for the attacks were empirically set for each scenario.

B. Attacks with Reinforcement Learning

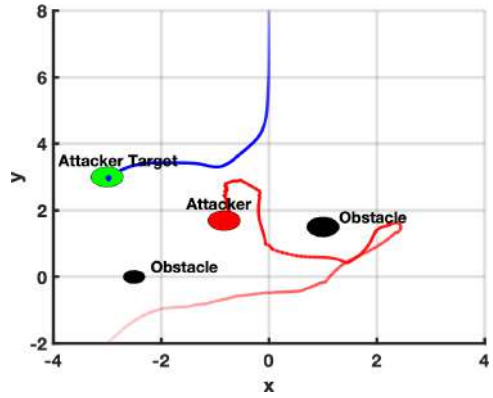
In this section, we present simulation results for the RL approach discussed in Section II-B. For all the experiments shown here the attacker's policy was represented by a neural network with two hidden layers (30×30 units) with hyperbolic tangent output, and a linear output layer which corresponds to the additional input to the attacker's dynamics $\mathbf{u}(s_t)$ in eq. (7). The network architecture was selected through an empirical evaluation of different network sizes. The final velocity of the attacker was limited through its maximum speed of 1.2 m/s. The networks were trained using the Proximal Policy Optimisation (PPO) algorithm [16] where the hyper-parameters (batch size and epochs) were empirically changed from scenario to scenario to optimise the results for each scenario. A scenario is defined by an environment with a fixed number and location of obstacles (no obstacles, one and two obstacles). For each scenario we experimented with three configurations of the initial



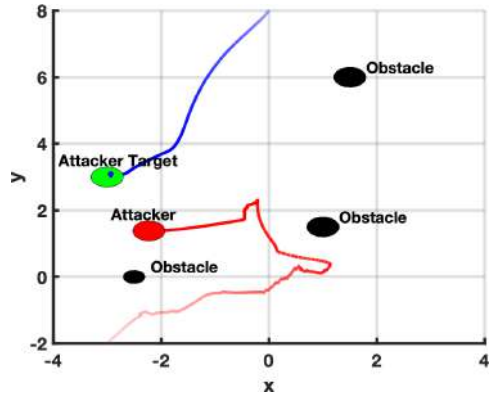
(a) No obstacles



(b) One obstacle



(c) Two obstacles



(d) Three obstacles

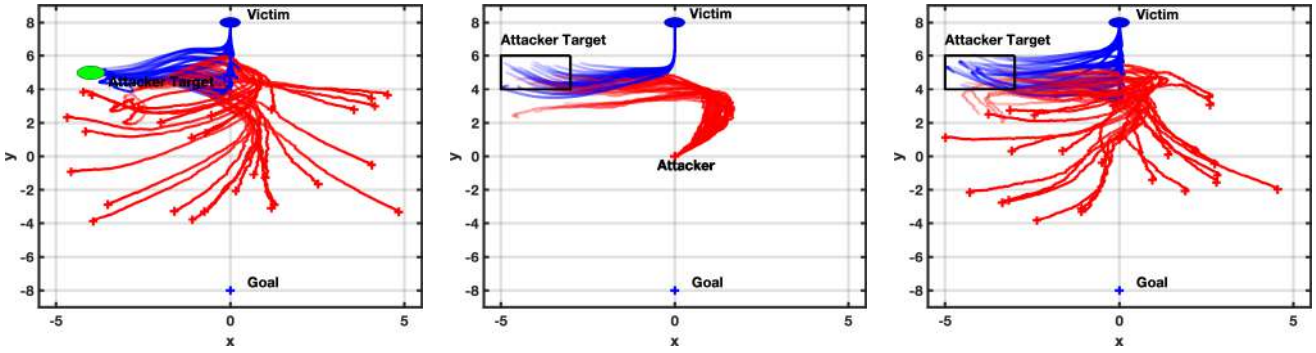
Fig. 1. Examples of trajectories adversarial attacks as an optimisation problem in the control input (open-loop) of the attacker (victim's goal $\mathbf{t}_v = [0, -8]$)

position and target of the attacker: (i) a fixed position for the attacker's target with random initial positions of the attacker inside some area, (ii) a random position for the attacker's target inside some area with fixed attacker initial position, and (iii) random attacker target and initial position. The starting position of the victim was $\mathbf{x}_v = [0, 8]$ and its goal $\mathbf{t}_v = [0, -8]$.

We trained 15 different policies for each of the three scenario and three configurations resulting in a total of $15 \times 3 \times 3 = 135$ policies. Once trained, the success rate of each policy was calculated over 1000 trajectories (with random positions of the attacker, the target or both depending on the configuration) and counting the number of trajectories which drove the victim to the target area. Although some of the trained policies had a success rate of zero, specially for the most challenging scenario with two obstacles, the lowest success rate among the easiest scenario (no obstacle) was 97% (i.e. 97 out of 100 attacks with that policy were successful).

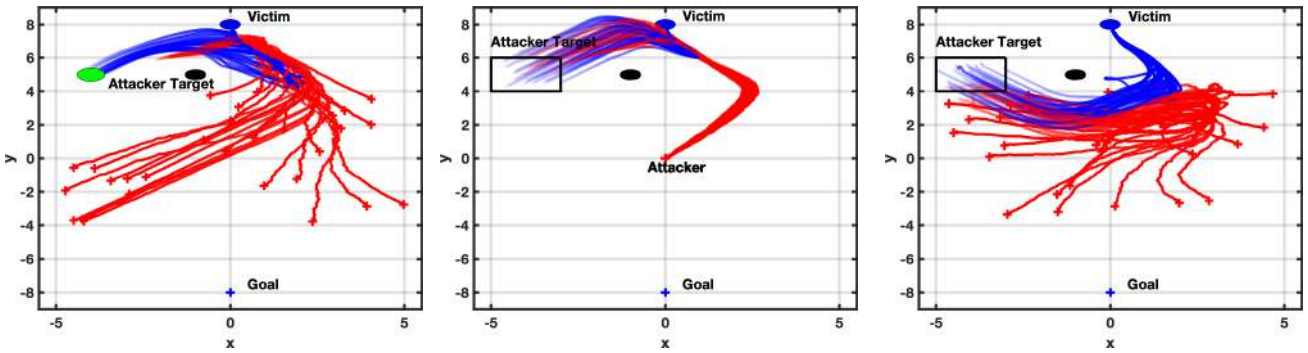
Figure 2 shows samples of 50 trajectories of attacks for each of the three configurations without obstacles (first scenario), i.e. random position of the attacker (Fig. 2(a)), random position of the target (Fig. 2(b)), and random attacker and target (Fig. 2(c)). The blue and red lines correspond respectively to the victim's and attacker's trajectories. The starting position of the victim is shown with a blue circle and its goal with a blue '+'. The initial positions of the attacker are shown with a red '+'. In Figures 2(b) and 2(c), the black rectangle depicts the area where the random target of the attacker was selected. As the figures show, the trajectories of the attacker and victim are highly prototypical, and the strategy of the attacker is to move towards the right of the environment to drive the victim towards the target on the left side. In terms of success rate, the best policies for all configurations lead to a 100% success, which means that for all the sample trajectories in the scenario without obstacles, the attacker drove the victim to the target.

In view of the trajectories shown in Figure 2, we experimented with placing an obstacle along the trajectories of the attacker, see Figure 3, and trained this new scenario for the three configurations. Figures 3(a), 3(b) and 3(c) show 50 sample trajectories for these configurations of attacker and target defined in the scenarios above. Interestingly, in the first two scenarios (where only the target or attacker's initial position are random) the trajectories pass on one side of the obstacle, while in the last scenario the trajectories pass on the other side, which could mean there is more than one possible strategy to perform the attack. The strategy of the attacker to move towards the right side of the environment is still present since its target is on the left side, and, in this way, the attacker drives the victim towards the left. It is worth noting that the rightmost points of the trajectories of the attacker are further to the right than in the no obstacle scenario since the victim moves to the right to avoid the obstacle. As for the success rates of the policies in these scenario, the best policies for the three configurations reached between 98% and 100% success in the 1000 random trials performed.



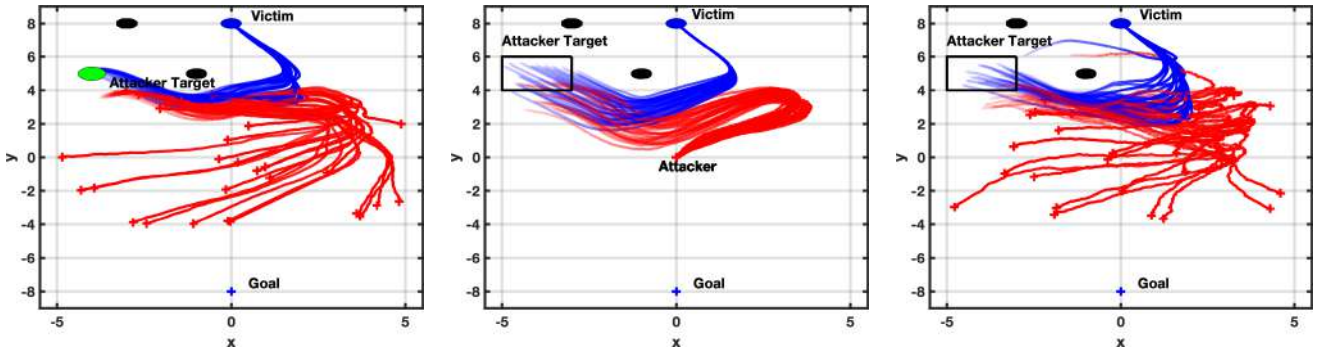
(a) Random attacker's position (b) Random target (c) Random target & attacker's position

Fig. 2. Sample trajectories of adversarial attacks using a neural policy for scenarios with no obstacles



(a) Random attacker's position (b) Random target (c) Random target & attacker's position

Fig. 3. Sample trajectories of adversarial attacks using a neural policy for scenarios with one obstacle



(a) Random attacker's position (b) Random target (c) Random target & attacker's position

Fig. 4. Sample trajectories of adversarial attacks using a neural policy for scenarios with two obstacles

Figure 4 shows the random test trajectories for the scenario with two obstacles, where the second obstacle was added to perturb the trajectories in the configurations shown in Figures 3(a) and 3(b), forcing the attacker to let the victim approach its goal and then pushing it back towards the attacker's target. As Figures 4(a) and 4(b) show, this strategy was successfully learnt by the policies, yet interestingly one of the simulated trajectories in Figure 4(c) drives the victim between the two obstacles. The success rates of the policies in this scenario were significantly reduced with one, four and five policies failing to drive the victim to the attacker's target for the three configurations. The best policies achieved a success rate between 98% and 100%. From the success rates achieved by the 15 learnt policies across scenarios one could infer that the more obstacles in the environment the more difficult is to learn a successful attacking policy, although successful policies can be found to create close to perfect attacks in all the scenarios tested.

IV. Conclusions and Future Work

This paper proposes two methodologies to generate adversarial attacks to robot behaviour with PFM-based obstacle avoidance as a case study. Although this can be seen as

a simplistic example since real robots use combinations of planning and obstacle avoidance, it might be possible to find attacks for such behaviours too. Our simulations showed that attacks to this obstacle avoidance strategy can be found even though good policies are less frequently found in more complex environments (environments with more obstacles). Although the proposed methodologies worked for our case study, an outstanding open question is how far these attacks can go, i.e. which other robot behaviours are vulnerable to attacks? If so, can adversarial attacks to robot behaviour be avoided or at least detected?

Our next objective is to apply these methodologies to generate adversarial attacks to other obstacle avoidance methods, such as those designed for unicycle type robots and deploy them in real robots, where sensor noise and other effects might play a key role on the feasibility of the attacks.

ACKNOWLEDGEMENTS

This work was supported by the Agencia Estatal de Investigación (Spain) (AEI /PID2020-119367RB-I00), the Xunta de Galicia - Consellería de Cultura, Educación, Formación Profesional e Universidades (ED431C 2022/19) and the European Union (European Regional Development Fund).

REFERENCES

[1] B. Nassi, A. Shabtai, R. Masuoka, and Y. Elovici, "Sok-security and privacy in the age of drones: threats, challenges, solution mechanisms, and scientific gaps," *arXiv preprint arXiv:1903.05155*, 2019.

[2] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, "A review on safety failures, security attacks, and available countermeasures for autonomous vehicles," *Ad Hoc Networks*, vol. 90, p. 101823, 2019.

[3] D. Heaven, "Why deep-learning AIs are so easy to fool," *Nature*, vol. 574, no. 7777, pp. 163–166, 2019.

[4] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[5] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," *arXiv preprint arXiv:1905.10615*, 2019.

[6] K.Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial attacks and defenses in deep learning," *Engineering*, vol. 6, no. 3, pp. 346–360, 2020.

[7] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles*. Springer, 1986, pp. 396–404.

[8] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 1991.

[9] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window Approach to collision avoidance," *IEEE Robot. and Autom. Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[10] J. Borenstein and Y. Koren, "The Vector Field Histogram - Fast obstacle avoidance for mobile robots," *IEEE Trans. on Robot. and Autom.*, vol. 7, no. 3, pp. 278–288, 1991.

[11] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Proc. of the Intl. Conf. on Robot. and Autom.* IEEE, 1996, pp. 3375–3382.

[12] J. Mínguez and L. Montano, "Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios," *IEEE Trans. on Robot. and Autom.*, vol. 20, no. 1, pp. 45–59, 2004.

[13] D. Kirk, *Optimal Control Theory: An Introduction*. Dover, 1970.

[14] K. Schittkowski, *Numerical Data Fitting in Dynamical Systems. A Practical Introduction with Applications and Software*. Springer, 2002.

[15] R. Sutton and A. Bartop, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

Evaluating Techniques for Accurate 3D Object Model Extraction through Image-based Deep Learning Object Detection and Point Cloud Segmentation*

Alicia Mora, Alberto Mendez and Ramon Barber

***Abstract*— Accurate 3D object model extraction is essential for a wide range of robotics applications, including grasping and object mapping, which require precise knowledge of objects' shape and location to perform optimally. However, high accuracy can be challenging to achieve, particularly when working with real-world data where factors like occlusions, clutter and noise can greatly influence results. Several techniques can be found in literature for integrating 2D deep learning and point cloud segmentation. Nevertheless, comparative studies on these algorithms are very limited. In contrast, this paper evaluates methods for obtaining 3D object models using a combination of deep learning object detection and point cloud segmentation. We compare a number of existing techniques, some of which have been improved for performance, on real-world data. More specifically, the paper examines four methods for 3D object extraction: two for bounding box object detection, one for instance segmentation and a fourth method that involves estimating an object mask in the image inside the bounding box. We compare these techniques qualitatively and quantitatively using several criteria, providing insights into their strengths and limitations.**

I. INTRODUCTION

The ability of robots to perceive and comprehend their environment is gaining importance as applications for them spread throughout society. One essential aspect for this is the capacity to extract 3D object models. These models enable robots to understand the geometry and spatial arrangement of objects in their surroundings. This information is particularly important in tasks where precise data about the objects shape and position is necessary. For instance, it allows to create semantic maps where the precise location of objects determines the zones at which robots will be capable of interacting with people [1]. Another example is estimating the 6-DoF grasp from a partial object view for a gripper [2]. In both cases, an error in the 3D object model extraction has a direct effect on the methods results. However, accurately extracting 3D object models from real-world data is a challenging task, especially in the presence of factors such as occlusions or cluttered backgrounds. In this work, we explore the potential of combining deep learning object detection with point cloud segmentation to extract accurate 3D object models. This study's objective is to assess the performance of various existing techniques while looking into their applicability to robotics.

*This work was supported by RoboCity2030 DIH-CM project (S2018/NMT-4331, RoboCity2030 Madrid Robotics Digital Innovation Hub)

RoboticsLab, Universidad Carlos III de Madrid, Leganés, Spain. almorav@ing.uc3m.es, albmende@pa.uc3m.es, rbarber@ing.uc3m.es

(a) (b)

Fig. 1. Visual representation of the presented problem: (a) original point cloud, (b) segmented point cloud, where the desired object (bottle) is colored in orange. The box occluding the object is successfully filtered.

Although there are methods that directly detect 3D objects in point clouds such as [3], they are computationally expensive and require big amounts of labeled data for training. Furthermore, since their utilization is not as prevalent compared to image object detection, there is a scarcity of pre-trained models readily accessible. Of the few models that are available, many are designed for autonomous driving applications, so they are not directly applicable for robotic applications. That is why the combination of image object detection and its subsequent projection onto the corresponding point cloud offers great advantages for applications like navigation, mapping or grasping.

A fundamental task in computer vision is object detection, and there are several methodologies for spotting and locating objects in pictures. In this paper, we investigate two main methodologies: bounding box detection and instance segmentation. While bounding box detection provides a rectangular bounding box around the detected object, instance segmentation segments each image pixel separately. Even though bounding box detection is computationally less expensive than instance segmentation, it requires more point cloud segmentation processing to extract 3D object models accurately, given that the box does not precisely adjust to the object shape. Instance segmentation, on the other hand, provides more detailed object localization and segmentation but it also has some drawbacks, including the need for larger dataset creation, greater computational requirements and more challenging training than bounding box detection.

Point cloud segmentation is the second step required for obtaining precise 3D object models. In this study, we investigate ways to segment point clouds enhanced by the prior detection of the object in the corresponding image. To do this, the point cloud projection onto the image plane is used to determine which points of the cloud correspond to

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

the detected object. Depending on the detection technique, a different local point cloud will be obtained after projection, so different filtering techniques will be required for the final 3D model extraction. In the case of bounding box detection, segmentation techniques will be necessary to remove points corresponding to background areas or occlusions, since the box does not fit the exact object shape. In the case of instance segmentation, special attention will be paid to points on the mask edge, as they may be outside objects. These approaches are evaluated and compared in this work to determine their effectiveness for real-world applications in terms of accuracy and time. An example is shown in Fig. 1, where a bottle is detected and segmented successfully despite occlusions.

II. REVIEW: 3D OBJECT MODEL EXTRACTION

In this section, we review several techniques proposed in literature for integrating image-based object detection and point cloud segmentation. These works have been divided according to their application: grasping and mapping. In both cases, we can find strategies based on the two main object detection techniques: bounding box and instance segmentation. Furthermore, they share point cloud segmentation and filtering techniques. Hence, our objective is to gather their main features to subsequently test their performance.

A. Grasping Applications

3D object models allow robots to approach and grasp objects with dexterity and accuracy. Without this information, a robot may struggle to determine the best approach to grasp an object, leading to suboptimal performance or even failure.

Several works have proposed to combine bounding box object detection and point cloud segmentation to solve this issue. Authors in [4] propose a bin picking solution where YOLOv2 first recognizes objects and then the point cloud data is segmented using the bounding box as a mask. Point cloud data is projected onto the image frame and points inside the box are selected. The fact of selecting points within the box that do not belong to the object is not taken into account. In [5], this fact is considered for grasping objects with a humanoid robot. Points outside of the robot manipulation range are removed, as well as the horizontal plane corresponding to the object supporting plane. However, other factors such as occlusions are not considered.

Other works propose the use of instance segmentation instead. Studies on using robots for harvesting like [6], [7] detect fruits using instance segmentation and apply the resulting mask to extract the corresponding point cloud, which is then fitted into a sphere model to estimate grasping. Similarly, in [8] 3D object models are estimated using instance segmentation and their point clouds are fitted into either a plane of a cylinder model. Additionally, ICP is applied for pose refinement. Authors in [9] apply the same technique for pick-and-place tasks and include a shape completion method to obtain a more accurate 3D model. Finally, authors in [10] use GrabCut to select the foreground object of an image cutout coming from a bounding box detection, better fitting the object shape and approaching the way instance segmentation works.

B. Mapping Applications

The 3D object model estimation for mapping provides the opportunity to improve the perception that robots have of their environment. For instance, it enables the creation of higher-level maps including semantic information [11]. Some works like [12] determine object locations using the center of the estimated bounding box to obtain depth. However, this point does not always belong to the object itself, so obtaining a complete 3D model of the object could improve this estimation.

Several works rely on initially detecting objects using bounding boxes to segment point clouds accordingly. Authors in [13] propose an object-aware map where 3D object models are included, fusing information from multiple view angles as explained in [14]. They first use the bounding box as a mask to crop the point cloud, which is then segmented using Locally Convex Connected Patches (LCCP). This algorithm, presented in [15], segments the point cloud into small blocks through supervoxel segmentation, which are later clustered into larger objects using a region growth algorithm based on convex-concave relationships. The biggest cluster is selected as the desired object. However, this could cause errors such as choosing areas that do not belong to the object, such as ground or background regions, in case they occupied more space. In [16], plane models are extracted from the cropped point cloud for visual semantic SLAM on a UAV. Other works like [17], [18] propose to remove ground points from the cropped point cloud using RANSAC before applying Euclidean filtering. The main problem of these methods is how to appropriately select the distance threshold, which will highly influence the segmentation performance. Authors in [19] propose to segment the point cloud before cropping it using the bounding box. Then, clusters are projected into the image plane. Those containing an area above a certain threshold inside the bounding box are selected as part of the detected object. This method has the advantage of merging multiple clusters belonging to the same object.

Regarding instance segmentation, multiple works make use of this technique for mapping. In [20], the resulting mask is used to crop the point cloud. No information regarding cloud filtering is provided, which could cause errors in the final object shape estimation. Works presented in [21], [22], [23] generate object-aware semantic maps based on SLAM. In all these cases, a filtering step is applied after cropping the point cloud. In [21], a seeded region growing algorithm is proposed to remove points in the mask contour that do not belong to the object. In [22], DBSCAN and a connected component analysis are combined for the same purpose. In [23], the point cloud is segmented according to estimated normals. Other works like [1] apply other filtering techniques like statistical filtering and Euclidean clustering to create semantic maps, where object points are projected onto the map plane. Special attention should be paid to the performance of these filters, since a small error in the edges

of the instance mask can cause large errors in the 3D object shape estimation.

This overview on 3D object model extraction highlights that the variety of applications has turned out into a variety of algorithms. For a practical comparison of several approaches, we analyze the performance of four methods: region growing, LCCP, GrabCut and instance segmentation.

III. IMPLEMENTED 3D OBJECT SEGMENTATION ALGORITHMS

Of all the reviewed methods, four have been selected to be compared among each other. All of them start with obtaining an image and a point cloud from an RGB-D camera. Three of the methods are based on object detection using bounding boxes, while the fourth one is based on the mask obtained by instance segmentation. In all these cases, a relationship must be established between the 3D point cloud and the 2D image. Subsequently, the procedure for image object detection, as well as the projection of 3D points onto the image plane and the applied segmentation techniques are explained. Code has been developed using the PCL library in C++ [24] and Open3D in Python [25].

A. Object Detection

Object detection is performed in the 2D image after the data capturing stage. In order to achieve this objective, a convolutional neural network (CNN) that works in real-time is used in this work: *YOLOv5* [26]. This object detection can be done in two different ways: bounding box detection and instance segmentation. An example of the outcome from these two methods is shown in Fig. 2. The main difference between the two proposals is that bounding boxes delimit objects using a rectangle, which may lead to imprecise object localization. Meanwhile, instance segmentation delimits objects pixel by pixel, maintaining objects' shapes [27]. This has a significant impact during the CNN training. Instance segmentation requires larger datasets and more training time. Also, object labeling is more difficult and it takes more time per object instance. In this research, it is intended to test different 3D object model extraction algorithms using the two object detection methods to analyze their impact in 3D segmentation and to determine when it is recommended to use bounding box detection or instance segmentation.

(a) (b)

Fig. 2. Object detection methods: (a) bounding box detection, (b) instance segmentation.

B. 2D - 3D Correspondence

In all of the four proposed methods, a correspondence between 2D image pixels and 3D point cloud points is required. More specifically, 3D points need to be projected to the image plane to see whether they correspond to the object region obtained from the object detector (bounding box or mask) or not. For that purpose, the pinhole camera model is used, as shown in Fig. 3.

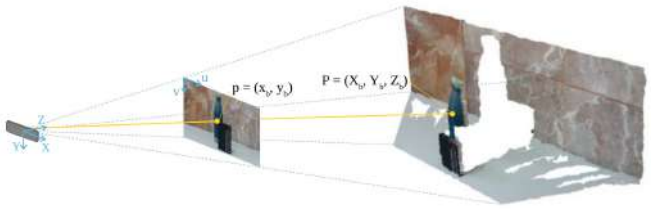


Fig. 3. Point cloud and image correspondence based on the pinhole camera model.

For establishing the relationship between a 3D point $P = (X_b, Y_b, Z_b)$ and a 2D point $p = (X_b, Y_b)$, the following equations are used:

$$x_b = \frac{f_x \cdot X_b}{Z_b} + c_x, \qquad y_b = \frac{f_y \cdot Y_b}{Z_b} + c_y \tag{1}$$

where x_b and y_b are the image coordinates of the point p, X_b, Y_b and Z_b are the point cloud coordinates of the point P, f_x and f_y are the focal lengths of the camera in the x and y directions respectively, and c_x and c_y are the coordinates of the image center.

C. Point Cloud Segmentation

In this section, the four selected segmentation methods are explained, three of them based on the output from bounding box detection and a fourth one based on the mask obtained from instance segmentation.

1) Region growing Algorithm: The region growing-based method is inspired by the work presented in [19]. In the proposed research, the whole point cloud was segmented using an incremental segmentation algorithm and then a criterion was set to see which clusters corresponded to the desired object by checking if they were inside the object bounding box. This criterion was checked from several points of view. However, in our research we are only focused on single view applications, so the method has been modified to be applicable to these situations. The first step is applying the region growing algorithm to the global point cloud. Region growing is a point cloud segmentation technique that groups neighboring points based on similarity criteria such as color, intensity, or distance. The output is a set of clusters corresponding to either objects or object parts. This last case is mostly found in non-convex objects such as chairs, which are typically divided into backrest and seat. Then, each cluster is projected into the image plane using the pinhole model. For each projected cluster, the ratio of cluster points inside the bounding box over the total number of cluster points is calculated. If this value is over 0.9, the cluster

is selected as part of the object. In this way, background and occluding objects are filtered, since it is assumed that their shape cannot be fully or almost fully contained in the bounding box. An example of region growing segmentation is shown in Fig. 4. According to the defined criterion, only the purple cluster would be selected as part of the object.

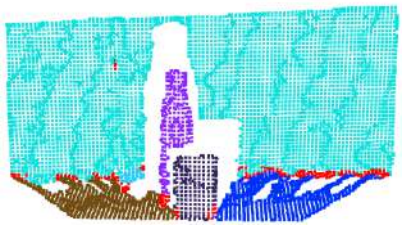
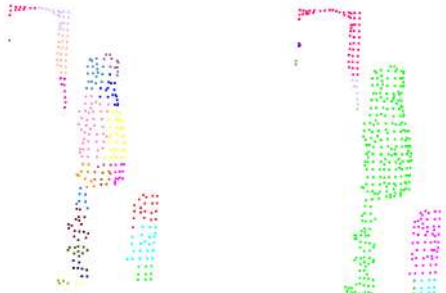


Fig. 4. Region growing segmentation applied on a point cloud. According to the proposed criterion, the purple cluster corresponding to the bottle would be selected. Red points are outliers.

2) LCCP: The LCCP-based method is inspired by the works presented in [13], [14]. In their research, authors proposed to crop the point cloud first by selecting the 3D points inside the object 2D bounding box. Then, the LCCP algorithm was applied on the local point cloud. The LCCP algorithm identifies connected regions with locally similar geometry and appearance, and assigns a unique label to each region based on its convexity and connectedness properties. It relies on two main stages: division into small voxels using supervoxel segmentation and merging voxels by computing an adjacency graph. An example of these two steps is shown in Fig. 5.



(a) (b)

Fig. 5. LCCP stages: (a) supervoxel segmentation, (b) supervoxel merging based on an adjacency graph. The green cluster corresponds to the desired object.

In order to select the cluster corresponding to the object, authors assumed it to be at the center of the bounding box and to occupy most area in the box. However, this assumption cannot be made, specially for non-convex objects. LCCP tends to oversegment objects like chairs, since it pays special attention to convex shapes. Hence, a strategy for selecting more than one cluster is needed in case the object is divided into two or more parts. For that reason, we propose to select clusters assuming that they are at the center of the point cloud (instead of the center of the bounding box) and that they are greater than other clusters. In this way, small clusters corresponding to occlusions are removed because of their size and large background clusters are also filtered because they are too far from the center.

3) GrabCut: The GrabCut-based method is inspired by the work presented in [10]. The objective is achieving similar 2D object detection results as with the instance segmentation method but using a CNN that performs bounding box detection. The proposal consists of two main processes: 2D object extraction and point cloud segmentation. Object extraction is performed using OpenCV's GrabCut [28]. The algorithm uses a Gaussian Mixture Model (GMM) to model the pixel color distribution on an image. Then, a graph is built based on it, where its weights depend on pixel similarity. Finally, a min-cut algorithm generates the binary mask that delimits the object using a minimal cost function. This process is repeated until convergence is achieved. Once an image from the camera is received, it is cropped using the bounding box. It is in this crop where GrabCut is applied so that the object shape is better defined. Then, like in the previous methods, the points in the point cloud corresponding to that region are selected. Given that the GrabCut mask is not as precise as the instance segmentation one and that occlusions may not have been filtered, an additional step is added to remove unwanted data. A density-based clustering algorithm, more specifically DBSCAN [29], segments the point cloud into several clusters according to a neighborhood distance threshold. Then, for each cluster the mean distance to the center of the picture is calculated, and the cluster with the minimum distance is selected as the final point cloud object. Fig. 6 shows a representative example of the proposed stages.

(a) (b) (c)

Fig. 6. GrabCut-based method stages: (a) the image is cropped using the bounding box delimitation, (b) GrabCut is applied to remove backgroud pixels, (c) 3D points inside the mask are selected and filtered to remove background and occlusions.

4) Instance Segmentation: The instance segmentation-based method is the most straight forward proposal. The pinhole camera model is again applied to see which points from the 3D point cloud are inside the calculated mask, which has the shape of the object. However, the output needs to be filtered in case that the mask does not perfectly fit the object shape. It could happen that the mask boundary is outside the object, so points that are far from the object would be selected, as shown in Fig. 7. For that reason, the

mask is first eroded using a disk-shaped kernel of radius 3. Then, an outlier removal filter is applied in order to be sure that only points corresponding to the object are selected.

(a) (b)

Fig. 7. Instance segmentation errors: (a) general point cloud view, (b) close-up view. If the segmented cloud is not filtered, boundary points on the mask could correspond to other objects. In this case, they correspond to an occluding box.

IV. EVALUATION

For the purpose of testing the accuracy of the proposed methods, their results are evaluated with several metrics. First, data is captured from a varied number of objects, corresponding to both workspaces for manipulation and larger objects for navigation and mapping. Then, after applying the proposed methods, the segmentation quality is assessed by comparing results against human labeled data. Results are hereunder presented, both qualitatively and quantitatively.

A. Dataset

In order to test the presented algorithms, a dataset was required. For that purpose, a set of aligned RGB images and point clouds have been recorded using an RGB-D camera. The selected hardware is a RealSense D-435i. Regarding software, ROS has been chosen as the link between the hardware and the algorithms. ROS provides a standardized interface for obtaining and processing data from RGB-D cameras, simplifying the development of applications that use this type of sensor data. With respect to the selected objects, the main intention is to verify how the methods behave with both small and large objects. The goal is to observe if there is a differentiation between both, in order to recommend their application for manipulation, navigation, or both. Hence, the dataset has been divided into workspace objects and larger objects. A total of 26 frames were recorded, with an image resolution of 640×480 px and its corresponding aligned point cloud. A summary of all the collected samples is shown in Table II. It must be specified that each sample contains only one object of interest, but there may be more than one point of view for the same object in separated samples. The dataset is publicly available[1].

TABLE II
DATASET SUMMARY

Workspace objects		
Obj. type	# objects	# samples
Knife	1	1
Book	2	4
Monitor	1	2
Bottle	2	5

Larger objects		
Obj. type	# objects	# samples
Chair	3	8
Washbasin	1	1
Bag	1	1
Fridge	1	1
Sofa	2	2
Toilet	1	1

B. Quantitative Results

Regarding the quantitative evaluation of the proposed methods, four different metrics have been applied. The first one is execution time, with the aim of checking if methods are valid for real-time applications. The code has been executed on a 12th Gen Intel(R) Core(TM) i7-12700H CPU. The second metric is intersection over union (IoU), which quantifies the overlap between two 3D bounding boxes, one corresponding to the ground truth point cloud and the other one to the output from the specified method. For this purpose, each point cloud is delimited by its corresponding 3D bounding box. It is a unitless value scaled between 0 and 1. The third metric is Chamfer distance (CD), another similarity metric calculated as the sum of the distances between each point in one cloud and its nearest neighbor in the other cloud. Finally, the distance between the center of the ground truth point cloud and the estimated one is also measured. These metrics were selected to quantify the performance of each method as they are the most popular ones in state-of-the-art works. With this we aim to facilitate further comparisons of our proposals with others. Results

[1]https://www.kaggle.com/datasets/aliciamorav/object-segmentation-dataset

TABLE I
PERFORMANCE METRICS FOR THE FOUR PROPOSED METHODS

	Time (s)	IoU	CD (m)	Distance (m)	
RG	0.0599 ± 0.0105	0.5808 ± 0.2623	4.9550 ± 10.2449	0.0327 ± 0.0426	workspace objects
	0.4415 ± 0.2291	0.4512 ± 0.2243	997.7513 ± 841.9354	0.2085 ± 0.1225	larger objects
LCCP	**0.0058 ± 0.0044**	**0.6042 ± 0.3363**	7.7770 ± 10.8200	0.0489 ± 0.0872	workspace objects
	0.0383 ± 0.0217	0.4551 ± 0.2591	704.7949 ± 522.7520	0.2250 ± 0.1773	larger objects
GC	1.0789 ± 0.1589	0.5076 ± 0.2330	5.0067 ± 2.8707	0.0326 ± 0.0373	workspace objects
	2.3043 ± 1.3372	0.4525 ± 0.2566	600.1952 ± 490.6789	**0.1934 ± 0.1631**	larger objects
INST	0.0070 ± 0.0027	0.5246 ± 0.2391	**4.4195 ± 3.5965**	**0.0263 ± 0.0302**	workspace objects
	0.0124 ± 0.0083	**0.4763 ± 0.2038**	**459.2447 ± 425.6846**	0.2195 ± 0.1367	larger objects

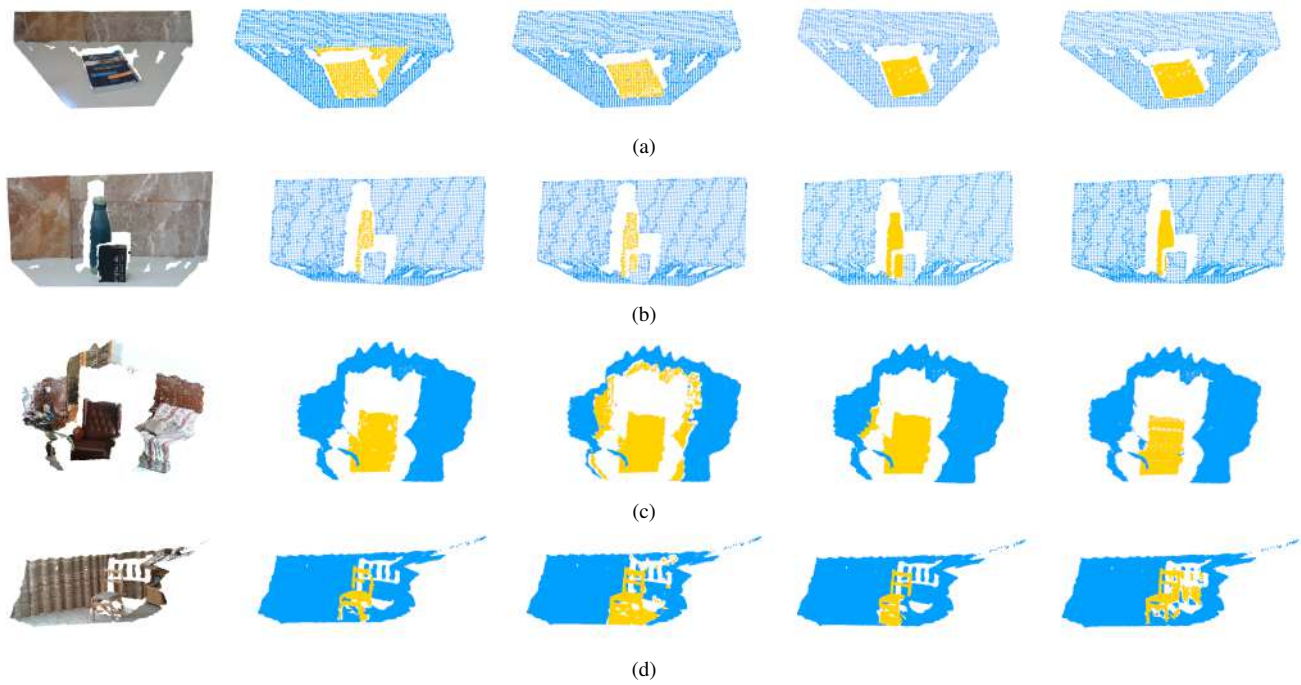


Fig. 8. Methods' main limitations when challenging conditions appear. Two workspace objects and two larger objects have been selected as representative examples: (a) thin book placed on a table, (b) bottle occluded by a box, (c) sofa on a cluttered environment, including occlusions, (d) chair with holes on its back. Results are presented from left to right in the following order: initial colored point cloud, RG, LCCP, GC and INST.

are collected in Table I, where the mean values as well as standard deviations are provided. Results have been divided for the two types of selected data: workspace objects and larger objects. The best value for each metric has been marked in bold.

By taking a look at time, it can be seen that GrabCut (GC) takes longer to execute than the rest of the methods for both object types. This is a key factor for selecting a segmentation method, since some applications may require real-time computations. Due to this fact, the most appropriate methods would be LCCP and instance segmentation (INST), since in our case all execution times were under the rate at which the camera captures data (30 fps). Regarding accuracy metrics, there is a clear differentiation in the methods performance with respect to the object type. All methods perform better with workspace objects in comparison to larger objects. Overall, instance segmentation provides the best results in both cases. In the case of workspace objetcs, RG and LCCP provide better results for IoU, but they are worse for CD and distance. Even so, these are minor differences. In the case of larger objects, only GC outperforms INST in distance. Although in this case the differences are also minor in IoU and distance, according to CD, INST is significantly better than the other options.

C. Qualitative Results

A visual representation of the methods' performance is shown in Fig. 8, where the most common errors for each method have been selected. Pictures are organized as follows: each row corresponds to a different object and each column corresponds to a different method. Column 1 is the initial colored point cloud and columns 2, 3, 4 and 5 are the results from RG, LCCP, GC and INST, respectively. By looking at the visual output obtained with each method, we can detect their main limitations.

Regarding workspace elements, Fig. 8(a) shows the example of an object that protrudes very little from the surface that supports it. More specifically, it is a book resting on a table. The first tested method, RG, is not capable of separating the object from the background into different clusters, so the object is defined with every point that is inside the bounding box. The rest of the methods, LCCP, GC and INST are able to clearly define the object shape. In the case of Fig. 8(b), a bottle with a box occluding it can be found. In this case, LCCP and GC have problems separating the different elements of the scene, since they include box points as part of the bottle. RG and INST differentiate both elements.

With respect to larger objects, two situations are shown: a complex-shaped object in a cluttered environment and an element with internal holes that allow the capture of background information through them. Fig. 8(c) shows the first case, where a sofa is detected. It has multiple elements around and it is additionally partially occluded by a table. In this case, the occlusion is correctly filtered by the four methods. However, the only methods that correctly segment the object are RG and INST. GC includes a part of another nearby object and LCCP includes a gib amount of background data. Finally, Fig. 8(d) shows a chair with holes on its back. RG and GC are capable of removing background points. The first one additionally does not include floor points, whereas GC does. LCCP does not remove floor points and includes background data. INST is capable of removing floor points

but background points are included in the model since they are also part of the object mask.

The described situations directly affect grasping and mapping performances. In the case of workspace objects like Fig. 8(a) and 8(b), grasping points would not be accurate because the point cloud does not perfectly fit the object shape. In the case of larger objects such as Fig. 8(c) and 8(d), their estimated location and dimension would be incorrect, even leading to conflicts between multiple objects that could overlap in the final map.

V. CONCLUSIONS

In this work, several 3D object model extraction methods have been presented. According to our results, instance segmentation provides the fastest and most accurate results. Due to its limitation in terms of dataset elaboration and training time, methods based on bounding box detection could be chosen. For workspace objects, the most accurate method is LCCP, while for larger objects, the use of GrabCut is recommended whenever it is not necessary to work in real time. Overall, it can be stated that point cloud segmentation via 2D object detection is a promising approach for obtaining accurate models.

As future work, we intend to combine the proposed segmentation strategy using YOLO and the corresponding point cloud with an object reconstruction strategy that will allow us to complete the hidden area of the detected objects. This will facilitate the correct functioning of the inclusion of real-time models for mapping or grasping tasks, since object shapes will be more accurate.

REFERENCES

[1] X. Qi, W. Wang, Z. Liao, X. Zhang, D. Yang and R. Wei, R., Object semantic grid mapping with 2D LiDAR and RGB-D camera for domestic robot navigation, Applied Sciences, 2020, vol. 10, no 17, p. 5782.

[2] A. Alliegro, M. Rudorfer, F. Frattin, A. Leonardis and T. Tommasi, End-to-end learning to grasp via sampling from object point clouds, IEEE Robotics and Automation Letters, 2022, vol. 7, no 4, p. 9865-9872.

[3] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley and C. Stachniss, Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data, IEEE Robotics and Automation Letters, 2021, vol. 6, no 4, p. 6529-6536.

[4] K. T. Song, Y. H. Chang, and J. H. Chen, 3D vision for object grasp and obstacle avoidance of a collaborative robot, IEEE/ASME AIM. IEEE, 2019. p. 254-258.

[5] L. Tian, N. M. Thalmann, D. Thalmann, Z. Fang and J. Zheng, Object grasping of humanoid robot based on YOLO, Advances in Computer Graphics: 36th CGI, 2019, Calgary, AB, Canada, June 17–20, 2019, Proceedings 36. Springer International Publishing, 2019. p. 476-482.

[6] H. Kang, H. Zhou, X. Wang and C. Chen, Real-time fruit recognition and grasping estimation for robotic apple harvesting, Sensors, 2020, vol. 20, no 19, p. 5670.

[7] G. Lin, Y. Tang, X. Zou and C. Wang, Three-dimensional reconstruction of guava fruits and branches using instance segmentation and geometry analysis, Computers and Electronics in Agriculture, 2021, vol. 184, p. 106107.

[8] G. J. Sun and H. Y. Lin, Robotic grasping using semantic segmentation and primitive geometric model based 3D pose estimation, 2020 IEEE/SICE SII. IEEE, 2020. p. 337-342.

[9] M. Gualtieri, and R. Platt, Robotic pick-and-place with uncertain object instance segmentation and shape completion, IEEE robotics and automation letters, 2021, vol. 6, no 2, p. 1753-1760.

[10] Z. Li, B. Xu, D. Wu, K. Zhao, M. Lu and J. Cong, A mobile robotic arm grasping system with autonomous navigation and object detection, 2021 ICCAIS. IEEE, 2021. p. 543-548.

[11] P. T. Wu, C. A. Yu, S. H. Chan, M. L. Chiang and L. C. Fu, Multi-Layer Environmental Affordance Map for Robust Indoor Localization, Event Detection and Social Friendly Navigation, 2019 IEEE/RSJ IROS. IEEE, 2019. p. 2945-2950.

[12] Y. Zhang, G. Tian, X. Shao, S. Liu, M. Zhang and P. Duan, Building metric-topological map to efficient object search for mobile robot, IEEE Transactions on Industrial Electronics, 2021, vol. 69, no 7, p. 7076-7087.

[13] L. Wang, R. Li, J. Sun, L. Zhao, H. Shi, H. S. Seah and B. Tandianus, Object-Aware Hybrid Map for Indoor Robot Visual Semantic Navigation. En 2019 IEEE ROBIO. IEEE, 2019. p. 1166-1172.

[14] L. Wang, et al., Multi-view fusion-based 3D object detection for robot indoor scene perception, Sensors, 2019, vol. 19, no 19, p. 4092.

[15] S. C. Stein, F. Wörgötter, M. Schoeler, J. Papon, J., and T. Kulvicius, Convexity based object partitioning for robot applications, 2014 IEEE ICRA. IEEE, 2014. p. 3213-3220.

[16] H. Bavle, P. De La Puente, J. P. How and P. Campoy, VPS-SLAM: Visual planar semantic SLAM for aerial robotic systems, IEEE Access, 2020, vol. 8, p. 60704-60718.

[17] X. Xu, L. Zhang, J. Yang, C. Cao, Z. Tan and M. Luo, Object detection based on fusion of sparse point cloud and image information, IEEE Transactions on Instrumentation and Measurement, 2021, vol. 70, p. 1-12.

[18] Z. Liao, W. Wang,X. Qi and X. Zhang, RGB-D object SLAM using quadrics for indoor environments, Sensors, 2020, vol. 20, no 18, p. 5150.

[19] Y. Nakajima and H. Saito, Efficient object-oriented semantic mapping with object detector. IEEE Access, 2018, vol. 7, p. 3206-3213.

[20] R. Mascaro, L. Teixeira and M. Chli, Volumetric Instance-Level Semantic Mapping Via Multi-View 2D-to-3D Label Diffusion, IEEE Robotics and Automation Letters, 2022, vol. 7, no 2, p. 3531-3538.

[21] K. Liu, Z. Fan, M. Liu and S. Zhang, Object-aware Semantic Mapping of Indoor Scenes using Octomap, 2019 CCC. IEEE, 2019. p. 8671-8676.

[22] S. Lin, J. Wang, M. Xu, H. Zhao and Z. Chen, Topology aware object-level semantic mapping towards more robust loop closure, IEEE Robotics and Automation Letters, 2021, vol. 6, no 4, p. 7041-7048.

[23] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart and J. Nieto, Volumetric instance-aware semantic mapping and 3D object discovery, IEEE Robotics and Automation Letters, 2019, vol. 4, no 3, p. 3037-3044.

[24] R. B. Rusu and S. Cousins, RUSU, Radu Bogdan; COUSINS, Steve. 3d is here: Point cloud library (pcl). En 2011 IEEE ICRA. IEEE, 2011. p. 1-4.

[25] Q. V. Zhou, J. Park and V. Koltun, ZHOU, Qian-Yi; PARK, Jaesik; KOLTUN, Vladlen. Open3D: A modern library for 3D data processing. arXiv preprint arXiv:1801.09847, 2018.

[26] G. Jocher, et. al., ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration, 2021, Zenodo.

[27] A. M. Hafiz and G. M. Bhat, A survey on instance segmentation: state of the art, International journal of multimedia information retrieval, 2020, vol. 9, no 3, p. 171-189.

[28] C. Rother, V. Kolmogorov and A. Blake, "GrabCut" interactive foreground extraction using iterated graph cuts. ACM TOG, 2004, vol. 23, no 3, p. 309-314.

[29] M. Ester, H. P. Kriegel, J. Sander & X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, KDD, 1996.

TAICHI algorithm: Human-Like Arm Data Generation applied on Non-Anthropomorphic Robotic Manipulators for Demonstration*

Blanca Lopez[1], Adrian Prados[1], Luis Moreno and Ramon Barber

***Abstract*— In household settings, Learning from Demonstration techniques can enable end-users to teach their robots new skills. Furthermore, it may be necessary for the demonstrations to be accessible through a straightforward setup, such as a single visual sensor. This study presents a pipeline that uses a single RGB-D sensor to demonstrate movements taking into account all the key points of the human arm to control a non-anthropomorphic arm. To perform this procedure, we present the TAICHI algorithm (Tracking Algorithm for Imitation of Complex Human Inputs). This method includes detecting key points on the human arm and mapping them to the robot, applying Gaussian filtering to smooth movements and reduce sensor noise, and utilizing an optimization algorithm to find the nearest configuration to the human arm while avoiding collisions with the environment or the robot itself. The novelty of this method lies in its utilization of key points from the human arm, specifically the end-effector and elbow, to derive a similar configuration for a non-anthropomorphic arm. Through tests encompassing various movements performed at different speeds, we have validated the efficacy of our method and confirmed its efficiency in replicating the desired outcomes on the robot's end-effector and joints.**

I. INTRODUCTION

The use of mobile robotic manipulators in dynamic environments, such as domestic scenarios, requires the capacity to acquire new skills to adapt to the changes of the environments in which the robot operates. Techniques such as Learning from Demonstration (LfD) [1] present a feasible alternative to traditional programming, allowing end-users to program the robot without the need of an expert and adapting it to their environment. One of the most important factors in this type of method is the generation of demonstrations. To carry out this process, a large number of techniques are available, which can be divided into two groups: direct demonstrations and indirect demonstrations [2].

Direct demonstrations encompass those techniques where the data collection process requires the use of the robot. Kinesthetic learning [3] is based on the physical interaction between the teacher and the robot's body, allowing a series of data to be generated directly with the robot's own sensors. Another method within direct demonstrations is the use of teleoperation [4]. The data collection process can be accomplished using various devices, such as joysticks, tactile sensors, or wearable devices. These devices enable robot control and data acquisition through the robot's internal sensors, but

*This work was supported by RoboCity2030 DIH-CM project(S2018/NMT-4331, RoboCity2030 Madrid Robotics Digital Innovation Hub)

Robotics Lab, Universidad Carlos III de Madrid, Leganés, Spain. e-mail: bllopezp@ing.uc3m.es, aprados@pa.uc3m.es, {moreno, rbarber}@ing.uc3m.es.[1]**Both authors contributed equally.**

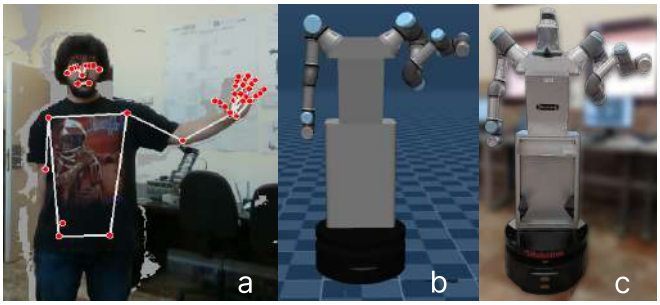


Fig. 1. Human demonstrations applying TAICHI algorithm. (a) Pose from human demonstrator, (b) Pose generated by the method in MujoCo simulation environment, (c) Pose in the real ADAM robot.

without the need for direct physical contact. Indirect demonstrations [5] encompass those techniques where no contact with the robot is required and the data collection process can be performed in a separate environment to that of the robot. These approaches are highly recommended for teaching high Degrees of Freedom (DoF) or non-anthropomorphic robots [1]. Within indirect demonstrations, and focusing on manipulation tasks, a highly used approach consists of learning how to replicate human movements. For accurate tracking of people, research centres often make use of motion capture systems (MoCap), formed by several cameras or vision systems [6]. These systems have clear disadvantages, such as the use of large spaces to mount the camera system or the need of using body markers as references. This prevents their direct application in real environments such as home scenarios. In this kind of setting, the use of simple visual sensors such as cameras presents a very convenient alternative, allowing for comfortable working.

By employing these sensors, it becomes possible to extract essential characteristics of the human body, which will then be utilized in the imitation process. Once the human movement is observed, it must be translated into suitable robot motion. However, in the context of non-anthropomorphic robotic arms, this transfer process is not straightforward. Typically, what can be demonstrated through human movements is solely the desired trajectory of the end-effector (EE). These data allow to transform EE motion to joint motions using Inverse Kinematics (IK) for the specific arm model. This method is not only used for LfD but is also commonly used for teleoperation of robotic arms [7]. Relying just on the end-effector as a tracking system for a non-anthropomorphic arm poses the risk of collisions with the environment since the remaining arm joints are not directly controlled. This is particularly hazardous in redundant arms, where multiple IK solutions are possible. Moreover, if the recorded trajectory

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

contains noise, it adversely affects the robot motion in terms of tracking accuracy and smoothness.

In this paper, we propose a pipeline for generating motion demonstration data, considering not only the end-effector (EE) of the human arm but also identifying the most suitable elbow configuration for a mobile robot equipped with non-anthropomorphic arms. To achieve this, our approach focuses on observing human movements using a single RGB-D sensor, which captures the position of key points on the human arm (see Fig. 1). We employ a Gaussian filtering method to reduce noise in the collected data, followed by an algorithm that utilizes a cost function to optimize the configurations of the robotic arm. To evaluate the effectiveness of our method, we conduct various types of movements and assess the goodness of fit in the relation to the recorded data, as well as the performance of the robot's motion on the end-effector. Our contributions are summarized as follows:

- Detection of relevant points of the human arm (wrist and elbow) using a single RGB-D camera and generation of an algorithm that optimises the position and orientation of a non-anthropomorphic arm to match the most human-like structure.
- Use of Gaussian filtering to smooth the collected human movements, considering demonstrations of different shapes and speeds.
- Implementation of the demonstration pipeline applicable in both MATLAB and Python, which includes human tracking, smoothing the obtained data and generating the robot trajectories and movements both in Matlab and MujoCo based simulators.
- Carrying out simulated and real tests on the ADAM mobile manipulator robot (see Figure 1c).

II. RELATED WORK

The process of tracking a person using depth sensors is widely used to capture natural human-related movements. One of the most common approaches to perform this process relies on solely tracking the hand palm. This procedure is applicable to both teleoperation and imitation. In teleoperation [8], a comparison is presented between human hand tracking systems based on data gloves and systems based on the use of optical hand tracking sensors like Leap Motion. In imitation [9], a 3D tracking of the human palm based on Fuzzy fusion is applied to estimate the configuration of the rest of the human arm. Another method based on palm tracking is presented in [10], where a vision-based data acquisition of the KUKA IIWA is presented by applying MediaPipe for hand coordinates extraction to obtain the orientation.

The use of RGB-D sensors for full-body tracking is also very common to be applied for this purpose. In [11], an ASUS Xtion PRO and OpenPose are used for body estimation, enabling movement control for bimanipulation tasks in the CENTAURO robot. Other approximations are presented in [12], where a motion capture system and data post-processing are used for characterization of a full upper limb robot. A similar idea is presented in [13], where using a Kinect camera and a skeletonization process of the human are used to teleoperate a low-cost arm. Regardless of whether the whole body tracking is used or not, all of the aforementioned methods are applied directly to non-redundant anthropomorphic manipulators. Consequently, tracking specific critical points of the human arm, such as the elbow, is not essential as anthropomorphic arms exhibit similar behavior to the human arm. Thus, by appropriately processing the end-effector data alone, satisfactory results can be achieved.

Only few works have been presented regarding the application of human data acquisition through sensors such as cameras directly to non-anthropomorphic manipulators. In the work presented in [14], the authors propose an initial solution to address this problem. They utilize an RGB-D camera along with OpenPose for motion capture, enabling end-effector processing for data acquisition. This data is then employed to control a UR3 arm by employing an analytic inverse kinematics (AIK) process. Other works such as [15] present a similar idea using BodyPoseNet for body feature extraction for dual parallel manipulation, taking into account the position of the opposite arm. Additionally, the works discussed in [16], [17] also employ an RGB-D system for data acquisition in the context of an UR3 arm. These studies focus on applying filtering techniques to refine the raw data obtained from the camera. By employing these filtering methods, they successfully generate smoother and more human-like movements in the robotic arm. However, none of these methods consider other key points of the human arm to determine the most appropriate human-like robotic configuration. Instead, they often rely on default configurations. Additionally, these methods do not take into account the layout of the surrounding environment or potential collisions with it, as the discussed robotic arms are not mounted on a real mobile robot.

In contrast to the aforementioned methods, our proposed approach involves extracting key points from the entire human arm to control a non-anthropomorphic robotic arm using a single RGB-D camera. Furthermore, we employ Gaussian filtering techniques to enhance the smoothness and accuracy of the collected human-like data. To achieve configurations that closely resemble the human arm, we combine an analytic inverse kinematics (AIK) method with an optimization process for the elbow position. This integrated approach takes into consideration both the robot's singularities and potential collisions with environmental elements, resulting in more realistic and safe arm configurations.

III. METHOD

The subsequent section introduces our approach, referred to as TAICHI (Tracking Algorithm for Imitation of Complex Human Inputs). This method is specially developed for indoor environments, such as residential houses, where deploying an extensive human tracking system to generate a training dataset for Learning from Demonstration (LfD) systems applied to non-anthropomorphic robotic arms may not be feasible. The general scheme of this algorithm is shown in Fig 2. The algorithm is divided into four main stages. The first stage involves extracting essential key points

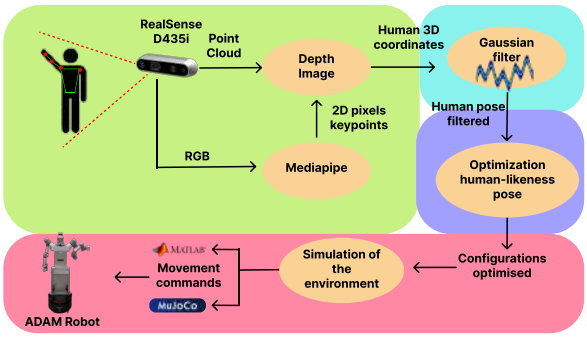


Fig. 2. General scheme of the TAICHI algorithm. The pipeline is made up of four different stages: human position tracking (green), filtering of the camera data (blue), optimisation of the arm configurations (purple) and the simulation or sending to the real ADAM robot (red).

of the human arm while performing a trajectory, specifically the shoulder, elbow, and wrist positions, as well as several key points of the hand. To accomplish this, we have developed a system that utilizes a RealSense D435i RGB-D camera. By leveraging Mediapipe [18], [19], we can extract the 2D positions of these significant landmarks of the human arm, along with their corresponding depth information. This allows us to accurately determine the 3D location of these points relative to the user's shoulder.

Once the data is obtained, it is necessary to filter out intrinsic noise from the camera and smooth the captured human movements. For this purpose, we have implemented a Gaussian filter [20], which acts on the computed position and orientation of the human elbow and wrist. The filtered data is then passed to the optimization algorithm, which aims to determine the most human-like position for each configuration of the human arm. This algorithm takes into account both the joint limits of the arm and the physical constraints of the robot's body to avoid collisions.

After obtaining the arm configurations, the algorithm enables the representation of the results in both the simulators implemented in Matlab and in MuJoCo [21]. Furthermore, these configurations can be directly transferred to the ADAM robot [22], which serves as the platform for the application of this work. The algorithm has been designed in such a way that anyone can easily modify the code in order to be applied to other non-anthropomorphic manipulator models and has been implemented in both Python and Matlab and is available in `https://github.com/AdrianPrados/TAICHI`. Each of the stages are explained in detail below.

A. Human tracking and data extraction

To acquire human data through demonstrations, it is crucial to capture users' movements. For this purpose, the information obtained from the RGB-D camera undergoes processing using MediaPipe, an open-source framework that facilitates 2D person detection and extraction of key points to analyze their movements. Determining depth information is also essential to obtain 3D positional and orientation data, and this is accomplished by utilizing the point cloud provided by the sensor. Once this information is obtained, it becomes necessary to shift the reference frame from the camera to that of the robotic arm base. This process involves breaking down human tracking into three stages: 2D pose estimation, 3D correspondence, and reference frame transformation.

The first stage consists on detecting people and estimating their pose in the 2D plane corresponding to the RGB image. MediaPipe estimates the user's pose via 33 markers. These markers are then analyzed in conjunction with the depth information to extract the estimated pose of each marker. In this particular case, we are interested in extracting the poses of the left arm. Therefore, we focus on and save three markers: the shoulder, elbow, and wrist, along with a fourth marker (right shoulder) for reference. The shoulder points are used to estimate the central point of the person and to correct the orientation of the human body, ensuring that the body is always facing the camera. In this research, the robot is thought to employ a gripper to manipulate objects. Consequently, the orientation of the end-effector is determined by the hand plane formed by three additional specific marker points: wrist point W, index finger methacarpophalangeal (mcp) I and pinky finger mcp P of the left hand, as shown in Fig. 3a. These points allow us to generate two vectors,

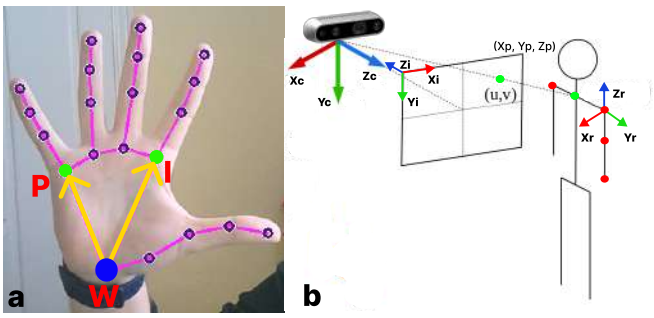


Fig. 3. (a) Palm plane for end-effector orientation extraction referred to image frame, where the blue point represents $\vec{N}$, (b) camera frame (Xc, Yc, Zc), image frame (Xi, Yi, Zi) and robot arm frame (Xr, Yr, Zr). The line joining the camera origin and the pixel (u, v) is defined by *rs2DeprojectPixelToPoint*, that obtains the 3D point (Xp, Yp, Zp) for each of the keypoints (red points).

$\vec{WI} = \vec{I} - \vec{W}$ and $\vec{WP} = \vec{P} - \vec{W}$ that can be used to obtain the normal vector (blue point in Fig. 3a) using the cross product and normalising its value as follows:

$$\vec{N} = \vec{WI} \times \vec{WP}, \ \hat{N} = \frac{\vec{N}}{N} \tag{1}$$

The second objective involves converting each 2D point of the human arm, measured in pixels, into a 3D point in meters based on the camera reference frame. To achieve this, the pinhole camera model is utilized in conjunction with point cloud data to convert from pixel-based to spatial information. The function *rs2DeprojectPixelToPoint* from the Python wrapper *PyRealsense2* is employed to accomplish this task. This function is responsible for generating a 3D vector that represents the ray passing through the (u, v) pixel coordinates. The outcome of this process is the intended 3D point in relation to the camera reference frame. Fig. 3b illustrates a schematic explanation of this method. The proper alignment of data from both the RGB image and the point cloud enables this task to be carried out. The final

task involves transforming the 3D points from the camera reference frame to that of the robot arm base, which aligns with the reference frame of the human shoulder. These are represented in Fig. 3b. Furthermore, a re-scaling process is essential to establish a correspondence between the robot arm and the human arm. This involves generating a correction factor based on the user's arm length, which allows mapping the positions of the wrist and elbow to ensure that the maximum range of the human arm matches the maximum range of the robotic arm.

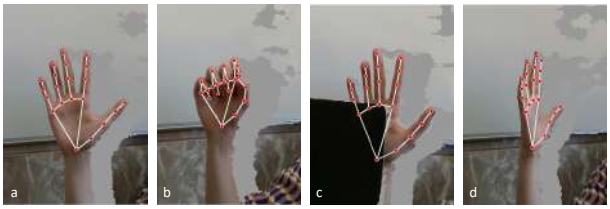


Fig. 4. Different hand poses captured through MediaPipe. (a) Hand fully open, (b) Fist, (c) Hand partially occluded, (d) Tilted hand.

It is worth noting that methods relying on RGB-D sensors for human tracking can be sensitive to occlusions, which may obstruct relevant body parts. In this study, if the captured data deviates significantly from the expected data considering the user's arm length and the intended trajectory, occlusions are likely responsible for information loss. In such cases, the positions of the wrist and elbow are estimated using a similar re-scaling projection process. Regarding the detection of human hand landmarks, it is worth mentioning that the MediaPipe framework offers a robust method for capturing the 3D positions of these key points. As depicted in Fig.4, even in challenging scenarios such as when the hand is in a fist position, partially occluded, or tilted, the algorithm is capable of accurately estimating the feature points. This capability allows us to compute the hand orientation, as previously discussed. However, it is important to monitor these situations closely, as they can indeed potentially degrade the quality of the captured data, especially when abrupt changes in human positions are recorded.

B. Gaussian Smoothing

Pre-processing and filtering data generated by demonstrations before being used for trajectory generation allows for noise smoothing of the acquired data. This filtering allows not only to improve the data by eliminating possible sources of error or noisy data, but is also beneficial when mapping to the robot arm. This is because continuous and even paths generate smoother responses on the robotic arms, which avoid over-oscillations when establishing the response to the human input data. In this work, a Gaussian filter has been applied for signal processing. This filter is a convolution operator that is used to remove noise from a signal. In this sense it is similar to the mean filter, but it uses a different kernel which stands for the shape of a Gaussian hump. These filters are characterized by narrow bandwidths, sharp cutoffs, and low overshoots. The main advantages against other methods such as Kalman Filters or Low pass filters are its simplicity, straightforward implementation and the absence of a complex, dynamic model. Focused on data acquisition using RGB-D cameras, Gaussian filters are a preferable option due to their ability to remove high frequency noise without negatively affecting low frequency components and preserve important data details. The Gaussian filter is based on the following Gaussian function:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{2}$$

where $-\infty \leq x \leq +\infty$, σ represents the standard deviation and μ represents the mean. The filter is applied by the function $gaussianFilter(input, \sigma)$ to both the elbow and wrist position and orientation values, for each of its components. The application of this filter for the EE position is depicted in Fig. 5. Empirically it has been observed that the best σ

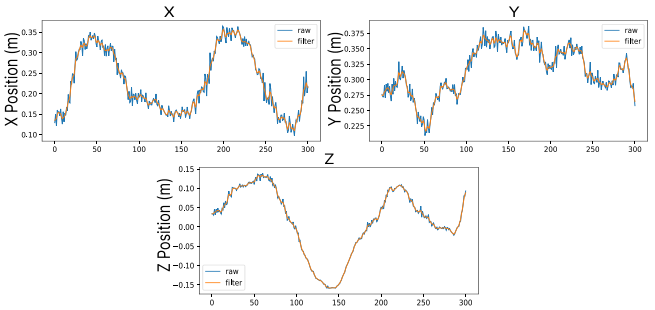


Fig. 5. Components for EE with respect to the human shoulder. The Gaussian filter (orange lines) smooths the values taken by the camera (blue lines) eliminating the peaks derived from sensor noise.

values for our application are in the range $\sigma = [0.5, 1.3]$. For values lower than 0.5 the filter does not generate any filtering, and for values higher than 1.3 the filter starts to eliminate important information, especially for the wrist orientation. For our experiments, a constant value of $\sigma = 1$ has been set.

C. Posture optimization

Once the filtered position and orientation data for the human arm are available, it is time to obtain the most human-like configurations for the non-anthropomorphic arm from these captured data. For this process, we have followed the concepts and methodologies presented in [23], adapting and applying them specifically to a non-anthropomorphic model like the UR3 arm. The applied human-robot mapping method pursues the minimisation of the distance between the elbow of the human arm and the rest of the joints of the robotic arm, except for the shoulder and the end-effector. A simplified schematic of this method can be seen in Fig. 6.

This method makes use of a combination of both an analytical inverse kinematics (AIK) of the non-anthropomorphic arm to obtain the 8 possible solutions and an optimization model based on direct kinematics to obtain the distances for each of the generated configurations. The AIK allows the algorithm to be able to solve the configurations in a very short time and minimise the EE error in position and orientation. The use of the cost function based on the distances to the human elbow allows estimating the most human-like position

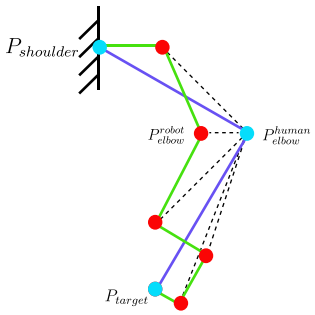


Fig. 6. The method is based on the minimization of the discontinuous lines between human elbow and the rest of the joints (red points) of the robotic arm. Green arm represents the UR3 arm and blue arm represents human arm. In both arms the $P_{shoudler}$ and the P_{target} are the same.

that the arm can reach among the possible options. The cost function in the first place depends on the positions of the EE. Let $X_R = f_R(q_R)$ denote the Forward Kinematics (FK) mapping for a non-anthropomorphic robot with n Degrees of Freedom (DOF), where $q_R \in \mathbb{R}^n$ is the vector of the desired joint angles, and let $X_H \in \mathbb{R}^3$ denote the human end-effector position. Hence, the metric for the EE position goal is defined as:

$$d_{Rp}(q_R) = ||X_R - X_H||^2 \tag{3}$$

The orientation influence is given using the robot EE orientation $h_R = (a_r, b_r, c_r, d_r)$ and the human orientation of the EE $h_H = (a_h, b_h, c_h, d_h)$, both expressed in quaternions. The orientation divergence is then expressed using:

$$\bar{d}_{Ro}(h_R, h_H) = \arccos(a_r a_h, b_r b_h, c_r c_h, d_r d_h) \tag{4}$$

To prevent problems derived from arm singularities, the cost function takes into account the antipodal points in $\mathbb{S}^3$, so the Equation 4 is finally defined as:

$$d_{Ro}(h_R, h_H) = \min(\bar{d}_{Ro}(h_R, h_H), \bar{d}_{Ro}(h_R, -h_H)) \tag{5}$$

In order to obtain the joint position distances, the metrics use the human elbow position as a reference. Let $S_{H_{elbow}} \in \mathbb{R}^3$ be the position of the human elbow in 3D space, and $S_j, j = 1, ..., n$ be the position of each robot joint in 3D space for an specific configuration obtained by AIK. The distance metric (excluding the shoulder and the EE) is given by:

$$D = \sum_{j=1}^{n} ||S_{H_{elbow}} - S_j||^2 \tag{6}$$

Finally, a continuity error $Error_{W1}$ is added to the cost function. This factor enables continuity to prevent abrupt variations in the robot wrist orientation. Hence, the cost will be lower when the configuration of the previous state and the next state are as similar as possible. By combining all the previously explained equations, the final cost function is expressed as:

$$\begin{aligned} F_{RH} = \min(W_p * d_{Rp}(q_R) + W_o * d_{Ro}(h_R, h_H) \\ + W_H * D + Error_{W1}) \end{aligned} \tag{7}$$

where W_p, W_o and W_H are weights for position, orientation and humanity respectively that adjust the relative importance of each factor. Before deriving the cost function for each arm configuration, the algorithm initially assesses two crucial factors. Firstly, it determines if the resulting configuration leads to any collisions with the robot's body or surrounding environment. Secondly, it verifies whether the configuration remains within the arm limits, which are unique to each model and assembly of the robot. If either of these two constraints occur, the resulting configuration will be directly ruled out from the possible solutions. In addition to the latter, it is important to note that by making the robotic arm follow natural human arm configurations, potential singular configurations are inherently discarded.

IV. EXPERIMENTAL RESULTS

In this section, we evaluate the accuracy of the developed TAICHI method applied to our ADAM robot. To conduct the evaluation, muliple users recorded a series of movements in real-time in front of the camera. The algorithm was then executed and the human-likeness of the generated robotic movements, the accuracy of the end effector tracking capabilities and the smoothness of the generated trajectories are analysed. A video with different examples of the experiments is available in `https://youtu.be/rSynqgXa_Yc`.

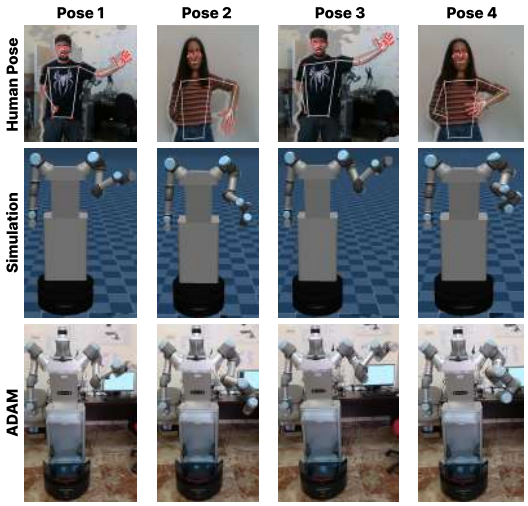


Fig. 7. Examples of how the robot follows human left arm poses.

A. Human-likeness Analysis of Posture

To demonstrate the performance of our method, we conducted various tests aimed at achieving robotic configurations that closely resemble human arm positions. The objective was to leverage the RealSense D435i camera to enable the robot to imitate the user's arm configurations as accurately as possible. Fig.7 illustrates the results of several qualitative tests, demonstrating the effectiveness of the method outlined in Section III in imitating human arm configurations despite the robot's different structure. It can be observed that the developed method enables the generation of positions where the robot arm configuration is as similar as possible to that of the human arm. It is important to acknowledge that achieving an exact match in configuration between both arms is highly challenging due to their structural disparities. Consequently,

it is qualitatively evident that the algorithm takes into account the unique arm structure while pursuing the desired EE goal position and orientation.

B. Accuracy Analysis of End-effector

To evaluate the accuracy of the method in terms of end-effector tracking, various trajectories with different starting points, end points, and velocities were executed. Fig. 8 illustrates an example case where the three-dimensional solution paths for both the human arm and the robotic arm are compared. As shown, the two paths are highly similar.

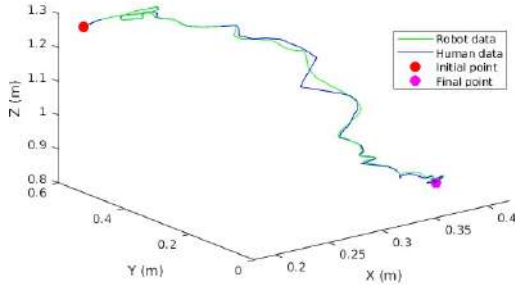


Fig. 8. Generated human (blue) and robot (green) end-effectors trajectories.

There exists a specific region where slight perturbations are present in the human path. These disturbances are caused by inherent physical constraints of the human arm, such as joint limits. The algorithm has the capability to detect and correct these perturbations, ensuring smooth robot movements while maintaining the overall consistency with the human trajectory. Overall, the three-dimensional end-effector trajectories demonstrate a qualitative and accurate tracking of the human data. If a quantitative comparison is made, and the previously shown path is decomposed into its positional and orientation components (see Fig. 10), it is observed that the error in both cases is almost negligible.

The method demonstrates a high degree of positional accuracy, closely matching the human path. The noticeable errors primarily occur at specific points where the algorithm optimizes and filters the errors originating from human data collection. One such example is at index 40 in the Y position. Examining the orientation errors reveals a similar pattern, with the two orientations being nearly identical. The maximum outlier value for orientation error is 0.114 radians.

To assess the algorithm's effectiveness, various tests were conducted involving individuals of different heights, genders, and physical builds. These tests involved performing similar movements at different speeds. The accuracy results obtained from these tests are summarized in Table I.

TABLE I

POSITION AND ORIENTATION ERRORS

	Max Outlier Error				**Mean Error**			
	x (m)	y (m)	z (m)	Θ (rad)	x (m)	y (m)	z (m)	Θ (rad)
Test 1	0.0230	0.0310	0.0241	0.1530	0.0014	0.0018	0.0014	0.0097
Test 2	0.0233	0.0311	**0.0207**	0.1147	0.0024	0.0022	0.0017	0.0084
Test 3	0.0294	0.0177	0.0156	0.0551	0.0050	0.0028	0.0027	0.0062
Test 4	0.0330	**0.0401**	0.0274	**0.2011**	**0.0100**	**0.0052**	**0.0610**	**0.0146**
Test 5	**0.0350**	0.0300	0.0170	0.1316	0.0036	0.0031	0.0030	0.0175
Test 6	0.0260	0.0160	0.0190	0.1543	0.0019	0.0021	0.0032	0.0187

Based on the results presented in Table I, it can be concluded that the TAICHI algorithm effectively tracks the human hand through the robot end-effector (EE) with high accuracy. The algorithm achieves a mean positional error of less than 1 cm and a mean rotational error of less than 0.02 radians. The maximum error values observed are localized to specific points along the path where human joint limits or captured data noise are encountered. However, the algorithm successfully detects and corrects these errors, ensuring a smooth trajectory for the robot. As a result, the positional and orientation errors are consistently maintained below 4 cm and 0.2 radians, respectively.

TABLE II

JERK VALUES FOR HUMAN AND ROBOT MOTIONS

	Human Jerk Values		**Robot Jerk Values**	
	Elbow	Wrist	Elbow	Wrist
Test 1	6.3708	10.4192	6.5617	10.6246
Test 2	4.6500	8.3615	4.7801	8.3400
Test 3	3.7849	5.8892	3.9124	6.0059
Test 4	3.1862	5.2660	3.2540	5.2305
Test 5	3.0215	7.2156	3.1247	7.4561
Test 6	3.5489	6.3384	3.7321	6.5101

In addition, a study of the jerk value for the different trajectories has been carried out. Jerk is estimated as the time derivative of acceleration, and it is an important factor in both suppressing vibration and achieving high accuracy in path generation. In our case study, the jerk values represent the minimum variation in acceleration changes between the human and robot arm movements. The more similar the two results are, the more similar the trajectories of the robotic arm will be and the more similar they can be assumed to be to human trajectories. Table II presents the results of studies for the critical points of the arm (wrist and elbow). The mean value for the difference between the jerk values of the wrist is 0.1289 and 0.1337 for the elbow. Since the mean difference is less than 0.15, it can be assumed that the TAICHI algorithm generates movements similar to those of the human arm.

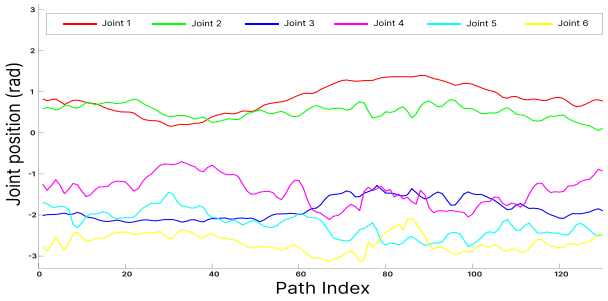


Fig. 9. Smoothness study of robotic arm configurations for Test 1.

C. Smoothness Analysis of Joint Configurations

Finally, we have conducted a quantitative analysis of the smoothness and feasibility of the computed robotic configuration trajectories. This analysis involved examining the continuity of the joint values across different test runs. Fig. 9 presents an example of these results for a specific use case. It is evident that the generated trajectories exhibit a continuous and smooth profile, with no significant changes

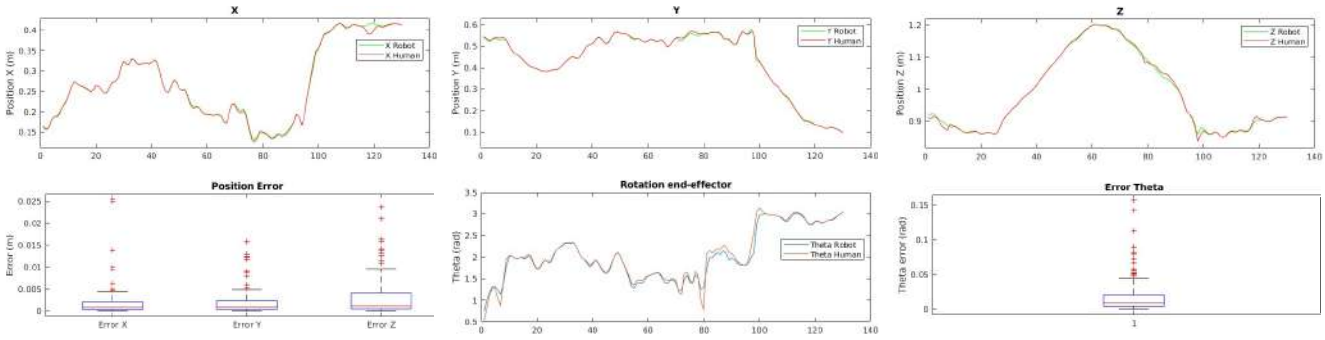


Fig. 10. Accuracy evaluation of end-effector tracking for Test 2.

in the configurations or abrupt changes in sign, which would indicate sudden changes in joint configurations. This guarantees that trajectories computed by the TAICHI algorithm are achievable by a real robotic platform.

V. CONCLUSIONS

In this paper, we have presented the TAICHI system, which comprises an RGB-D sensor, a human position capturing process, an AIK framework, Gaussian filtering, and a configuration optimizer. The effectiveness of the system in data acquisition for non-anthropomorphic arms has been demonstrated, as it successfully obtains human-like configurations while considering arm limitations and achieving minimal end-effector tracking errors. Moreover, the TAICHI system is user-friendly, efficient, and adaptable to various environments, making it suitable for LfD data collection.

For future work, we intend to enhance the tracking system to support bimanipulation tasks. Additionally, we plan to incorporate a tracking system for both hands simultaneously, enabling the collection of grasping data.

REFERENCES

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[2] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun, "Survey of imitation learning for robotic manipulation," *International Journal of Intelligent Robotics and Applications*, vol. 3, pp. 362–369, 2019.

[3] A. Prados, A. Mora, B. López, J. Muñoz, S. Garrido, and R. Barber, "Kinesthetic learning based on fast marching square method for manipulation," *Applied Sciences*, vol. 13, no. 4, p. 2028, 2023.

[4] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5628–5635.

[5] D. Vogt, S. Stepputtis, S. Grehl, B. Jung, and H. B. Amor, "A system for learning continuous human-robot interactions from human-human demonstrations," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2882–2889.

[6] P. Racinskis, J. Arents, and M. Greitans, "A motion capture and imitation learning based approach to robot control," *Applied Sciences*, vol. 12, no. 14, p. 7186, 2022.

[7] Q. Gao, Z. Ju, Y. Chen, Q. Wang, and C. Chi, "An efficient rgb-d hand gesture detection framework for dexterous robot hand-arm teleoperation system," *IEEE Transactions on Human-Machine Systems*, 2022.

[8] C. Mizera, T. Delrieu, V. Weistroffer, C. Andriot, A. Decatoire, and J.-P. Gazeau, "Evaluation of hand-tracking systems in teleoperation and virtual dexterous manipulation," *IEEE Sensors Journal*, vol. 20, no. 3, pp. 1642–1655, 2020.

[9] C.-F. Juang, C.-W. Chang, and T.-H. Hung, "Hand palm tracking in monocular images by fuzzy rule-based fusion of explainable fuzzy features with robot imitation application," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 12, pp. 3594–3606, 2021.

[10] H. Sharma, R. K. Yadav, and B. Amrutur, "Vision-driven tele-operation for robot arm manipulation," in *2022 7th International Conference on Robotics and Automation Engineering (ICRAE)*. IEEE, 2022, pp. 123–129.

[11] E.-J. Rolley-Parnell, D. Kanoulas, A. Laurenzi, B. Delhaisse, L. Rozo, D. G. Caldwell, and N. G. Tsagarakis, "Bi-manual articulated robot teleoperation using an external rgb-d range sensor," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018, pp. 298–304.

[12] C. He, X.-W. Xu, X.-F. Zheng, C.-H. Xiong, Q.-L. Li, W.-B. Chen, and B.-Y. Sun, "Anthropomorphic reaching movement generating method for human-like upper limb robot," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13 225–13 236, 2021.

[13] M. Syakir, E. S. Ningrum, and I. A. Sulistijono, "Teleoperation robot arm using depth sensor," in *2019 International Electronics Symposium (IES)*. IEEE, 2019, pp. 394–399.

[14] J. B. Martin and F. Moutarde, "Real-time gestural control of robot manipulator through deep learning human-pose inference," in *Computer Vision Systems: 12th International Conference, ICVS 2019, Thessaloniki, Greece, September 23–25, 2019, Proceedings 12*. Springer, 2019, pp. 565–572.

[15] Q. Gao, J. Liu, Z. Ju, and X. Zhang, "Dual-hand detection for human–robot interaction by a parallel network based on hand detection and body pose estimation," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 12, pp. 9663–9672, 2019.

[16] M. Dagioglou, A. C. Tsitos, A. Smarnakis, and V. Karkaletsis, "Smoothing of human movements recorded by a single rgb-d camera for robot demonstrations," in *The 14th PErvasive Technologies Related to Assistive Environments Conference*, 2021, pp. 496–501.

[17] A. C. Tsitos and M. Dagioglou, "Handling vision noise through robot motion control in a real-time teleoperation system," in *2022 30th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2022, pp. 624–629.

[18] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, and M. Lee, "A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

[19] A. Mora, A. Prados, A. Mendez, R. Barber, and S. Garrido, "Sensor fusion for social navigation on a mobile robot based on fast marching square and gaussian mixture model," *Sensors*, vol. 22, no. 22, 2022.

[20] S. Moorthy, J. Y. Choi, and Y. H. Joo, "Gaussian-response correlation filter for robust visual object tracking," *Neurocomputing*, vol. 411, pp. 78–90, 2020.

[21] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[22] R. Barber, F. J. Ortiz, S. Garrido, F. M. Calatrava-Nicolás, A. Mora, A. Prados, J. A. Vera-Repullo, J. Roca-González, I. Méndez, and Ó. M. Mozos, "A multirobot system in an assisted home environment to support the elderly in their daily lives," *Sensors*, vol. 22, no. 20, p. 7983, 2022.

[23] M. Liarokapis, C. P. Bechlioulis, P. K. Artemiadis, and K. J. Kyriakopoulos, "Deriving humanlike arm hand system poses," *Journal of Mechanisms and Robotics*, vol. 9, no. 1, p. 011012, 2017.

Multi-Task Learning for Industrial Mobile Robot Perception Using a Simulated Warehouse Dataset

Dimitrios Arapis
Robotics and Operational Technologies
Novo Nordisk A/S
Bagsvaerd, Denmark
dtai@novonordisk.com

Andrea Vallone
Electrical and Photonics Engineering
DTU - Technical University of Denmark
Kgs. Lyngby, Denmark
s192327@student.dtu.dk

Milad Jami
Robotics and Operational Technologies
Novo Nordisk A/S
Bagsvaerd, Denmark
mjam@novonordisk.com

Lazaros Nalpantidis
Electrical and Photonics Engineering
DTU - Technical University of Denmark
Kgs. Lyngby, Denmark
lanalpa@dtu.dk

***Abstract*—Autonomous industrial mobile robots need advanced perception capabilities to operate safely and human-compliantly in shared working environments. To achieve a high-level understanding of the mobile robots' surroundings, this paper investigates Multi-Task Learning approaches to process multiple tasks simultaneously and potentially improve the generalization performance. Our work alleviates the scarcity of datasets that are relevant for industrial settings by introducing and making publicly available a simulated warehouse dataset (WarehouseSIM) covering semantic segmentation, depth estimation and surface normals estimation tasks. We collect and examine numerous MTL task-balancing techniques for industrial mobile robot perception. Our experiments show that MTL methods that have shown superior performance on different computer vision datasets fail to improve over the single-task learning setup in our scenario. This implies that the performance of those approaches is very dependent on the considered dataset, which further highlights the value of introducing new relevant datasets focused on industrial mobile robot environments.**

***Index Terms*—Multi-task learning (MTL), industrial mobile robots, perception, warehouse dataset**

I. Introduction

The usage of industrial mobile robots in production facilities and warehouses has the potential to revolutionize the way work is performed in these environments. This technology can reduce costs and improve safety, ultimately replacing human workers in hazardous or repetitive tasks. Additionally, mobile robots can work around the clock, increasing productivity and throughput. In order to achieve the desired effectiveness, it is imperative that mobile robots exhibit both flexibility and reliability within dynamic settings where they must seamlessly integrate with human workers, lifters, and other autonomous robotic systems, all of which contribute to the complex and continuously changing nature of these environments. A rudimentary level of environmental awareness has been attained for industrial mobile robots, primarily by employing LiDAR technology. Nevertheless, these approaches tend to concentrate on the identification of obstacles rather than a comprehensive understanding of the objects present within the surrounding context. Mobile robots equipped with cameras and machine learning algorithms can potentially perform more complex tasks, such as object detection, and enhance their interaction capabilities. While there are still many challenges to overcome, such as the need for robust and reliable sensing and perception algorithms, the potential benefits of adapting these technologies to industrial mobile robots are clear [1]. Figure

Fig. 1: Typical Novo Nordisk warehouse facility with a custom mobile manipulator (top left), and reference color image of the simulated warehouse facility taken from the WarehouseSIM dataset (bottom right)

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

1 (top left) depicts an industrial mobile robot equipped with a retrofitted module with additional sensors (3D LiDAR and LiDAR camera) and an edge computing unit which has been used to test various perception algorithms in the Novo Nordisk warehouse and production facilities.

The development in this area has been significantly hindered by the lack of publicly available datasets specific to these environments. This scarcity arises due to the closed nature of these facilities, where strict regulations necessitate privacy, proprietary information protection, and adherence to industry-specific safety and security standards. As a result, limited access to comprehensive data from warehouses and production facilities, in combination with the cost of the labeling process, highlights the need for alternative approaches. To advance AI-based perception for mobile robots in these specialized environments we generated a simulated warehouse dataset called WarehouseSIM, an example image of which is depicted in Fig. 1 (bottom right).

The primary objective of this study is to enable industrial mobile robots with limited computational resources with the capability to perform scene understanding through perception. The majority of AI-based perception algorithms require a single task to be optimized, e.g. image classification or depth estimation, and are usually trained individually for the specific task. However, human perception mechanisms have the ability to transfer knowledge across tasks, which is one of the most important indicators of advanced intelligence. This knowledge transfer plays an important role in improving the accuracy of information and allows complex reasoning. Similarly, in machine learning, it is possible to deploy a single model that can learn multiple tasks at once, i.e multi-task learning architectures (MTL), with the goal of improving the generalization performance by processing all tasks simultaneously. Another distinct advantage of MTL architectures over single-task learning architectures is that their shared computations across tasks result in faster and more efficient information processing during inference, which is rather significant for mobile robots. We test common task-balancing techniques—typically used in computer vision tasks—and highlight an important limitation of these techniques in our newly introduced dataset. Specifically, we found that previously used task-balancing techniques failed to improve single-task learning results, calling into question their effectiveness in new datasets. Our findings underscore the importance of carefully selecting and designing multi-task learning strategies and highlight the potential benefits of adopting a more systematic approach to this problem. By addressing this critical challenge, our work has the potential to significantly enhance the automation and efficiency of warehouse robotics.

The *contribution* of this work is threefold: (i) We make public a simulated warehouse dataset, with color image, depth, semantic and surface normals information. Furthermore, (ii) we collect and examine numerous MTL task-balancing techniques for industrial mobile robot perception. Finally, (iii) we provide a benchmark of multiple baseline results, laying the groundwork for future progress and development.

II. Related work

Much of the progress in AI-based perception can be attributed to the advancements made in the field of autonomous vehicles and the broader domain of computer vision [2]. Although mobile robot perception has benefited from these advancements, as it shares common underlying principles and challenges, the necessity for domain-specific development with focused datasets and models with real-time processing capabilities is paramount. Late development in mobile robot perception has pushed the boundaries of the field, moving from basic navigation and obstacle avoidance to higher-level scene understanding [3]–[6]. Graf et al. brought attention to the gap between narrow perception tasks, such as 2D object detection and 2D segmentation, that are typically solved in isolation, versus developments in holistic scene perception algorithms, which require tasks to be solved together [7]. Due to the intricate nature of the MTL problem, several research findings indicate that definitive assumptions cannot be made when designing a multi-task setting [8]–[11]. The primary factor behind this is that when simultaneously training on a shared set of features, the system often struggles to strike a balance between competing optimization objectives. The effectiveness of different approaches can vary based on four key dimensions: the specific tasks involved, the model architectures employed, the available data, and the performance metrics considered. Regarding the tasks involved, some works have focused on identifying task relationships and if they should be trained together. In their study cited as [12], Zamir et al. aimed at creating a computational model that can effectively identify task relationships that are conducive to transfer learning scenarios. In another work from some of the same authors, a computational framework for differentiating which tasks should be trained together and which individually is presented [13]. Similarly, Finn et al. proposed an inter-task affinity metric to measure task relationships [14]. Although considerable effort has been devoted to researching how task relationships affect the performance of each other, this remains a complex problem that has not been solved. In the work of Stadley et al., training an MTL network with surface normals improved all other tasks, hurting, however, the performance of surface normal estimation itself [13]. In [15] focusing on cross-task consistency, Zamir et al. also showed that information on surface normals helped improve the performance of other tasks. Several studies have examined how the architecture of the learning algorithm impacts the magnitude of potential improvement. In [16], a detailed comparison between a standard multi-head split for each task and a multi-task attention network (MTAN) [17] architectures shows that the same weighting strategies affect each architecture differently. A collection of architectures specifically designed for MTL can be found in [18]. Vandenhende et al. provide a quantitative analysis of different MTL architectures applied on NYUD-v2 [19] and PASCAL [20] datasets [9]. Over the past few years, attention has shifted toward developing effective task-balancing techniques. These techniques aim to distribute the

workload evenly across multiple tasks in a MTL scenario, where a machine learning model is trained to perform multiple tasks simultaneously. Vandenhende et al. [9] have categorized task-balancing methods into two categories: indirect and direct methods. Indirect methods, which are predominantly weight-based methods, adjust the weight of individual task losses relative to the total loss [17], [21]–[23]. Direct methods operate on the shared task gradients to balance the learning process among multiple tasks [24]–[28]. Recent work has shown that task-balancing techniques that have previously dominated a specific field, e.g. with a certain model architecture and a specific dataset, fail to generalize in other architectures or tasks, often resulting in worse results [29].

III. Industrial Mobile Robot Dataset

To support the optimization of multi-task learning (MTL) architectures for industrial mobile robots, we have taken the initiative to generate a comprehensive dataset specifically designed for this purpose. *WarehouseSIM* is a dataset made using the Isaac Sim platform which is built based on NVIDIA Omniverse. We utilized different randomization components to create 3 different warehouse scenes. For each scene, the lighting, transformations (object position, scale, orientation) and textures where randomized. We incorporated several mobile robots (MIR100) equipped with a robotic arm (UR5), mirroring the physical robots utilized in our testing facilities. We used these mobile robots to collect data as they navigated in the environments by fixing a virtual camera to their body. Once each scene was generated, we used the synthetic data recorder tool to record data from all sensors. We collected 2125 images with corresponding depth maps, semantic segmentation, instance segmentation, 2D tight bounding box and 2D loose bounding box. Due to the unavailability of generating synthetic surface normals using the Isaac Sim platform, we used the work of Boulch and Marlet [30] to generate the surface normals from the synthetic depth maps. By employing a technique based on local surface fitting, their approach demonstrates strong performance when applied to depth maps containing well-defined features. The histogram representing pixel occurrences of the different objects existing in the WarehouseSIM dataset are visualized in Fig. 3. The mapping between class names and label numbers is presented in Table I. Fig. 2 visualizes the distribution of depth values in the scene, where each pixel is counted and categorized into 1-meter ranges. From the 2125 images of the full dataset, we use 1456 images for training, 206 for testing and 463 for validation. The dataset is accessible online and can be downloaded from the project repository by following the provided link: https://github.com/DTU-PAS/WarehouseSIM.

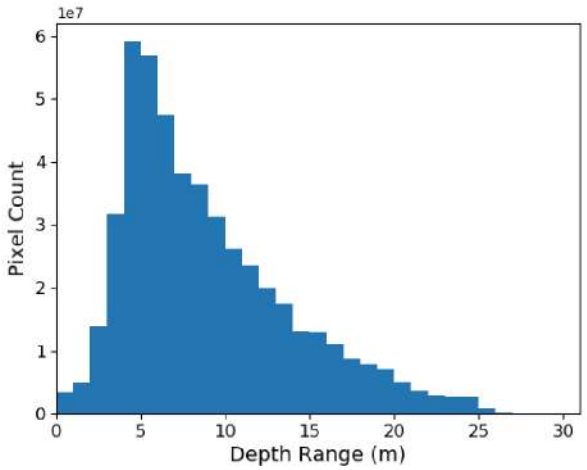


Fig. 2: Depth distribution in the WarehouseSIM dataset counted in pixels within a depth range.

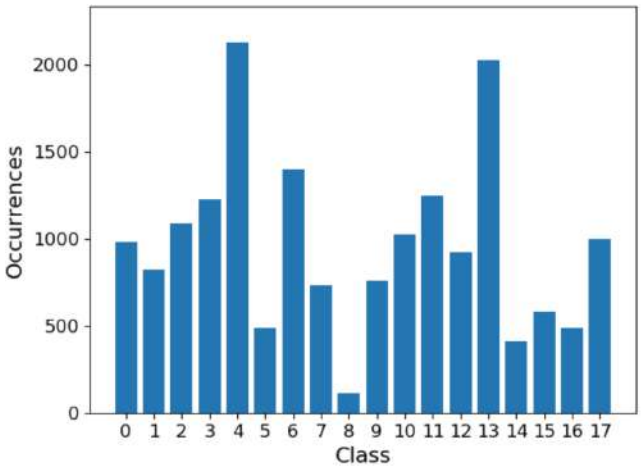


Fig. 3: Classes distribution in the WarehouseSIM dataset counted in occurrences of pixels belonging to a class.

IV. Experimental Setup

A. Tasks

In this research, we focus on the dense prediction tasks: semantic segmentation, depth estimation and surface normals estimation, which all require pixel-level predictions on corresponding RGB images. *Semantic segmentation* requires an image to be divided into different subgroups (segments) that describe different classes. Using a simulated warehouse environment as an example (see Fig. 4a), a label should be assigned to each pixel in the image, resulting in floor, human, mobile robot, racks, and other segments, as seen in Fig. 4b. *Depth estimation* is the task of assigning a depth value to each pixel of an image. The result is a depth map that contains information about the distance of the depicted objects, as seen in Fig. 4c). Finally, surface normals describe vectors perpendicular to the plane at a given 3D point. *Surface normal estimation* is the task of assigning a vector for every pixel of the image (see Fig. 4d).

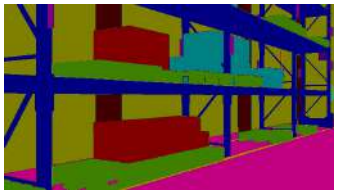
B. Architecture

Following previous practices [16], [23], we use the improved version of DeepLabv3 [31] with a ResNet-18 shared encoder with dilated convolutions, and task-specific heads for each task designed using the Atrous Spatial pyramid Pooling

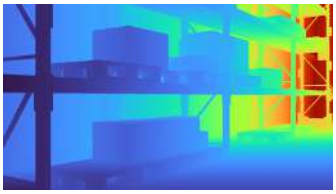
TABLE I: Corresponding classes of the WarehouseSIM dataset

Number	Name	Number	Name
0	Humans	9	Imported mobile robots
1	Purple Boxes	10	Pallets
2	Structural columns	11	Floor markings
3	Shelves	12	Shelf section signs
4	Walls	13	Floor
5	Trolleys	14	Plastic containers
6	Signs	15	Wall cabling
7	Cardboards	16	Electrical enclosures
8	Electrical plugs	17	Fire extinguishers

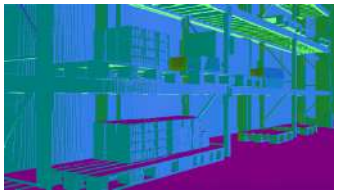
(a) Reference color image



(b) Semantic labels



(c) Normalized depth map



(d) Surface normals data

Fig. 4: WarehouseSIM dataset color image with corresponding depth, semantic and surface normals data

(ASPP) module [32]. In light of the recent advancements in hardware within the realm of robotics and edge computing, models with increased size and complexity will exhibit the capability to be applied in mobile robotic contexts. Therefore, we also tested with ResNet-34 and Resnet-50 encoders but found that the resulting improvement in performance was only marginal, and came at the expense of a significant increase in training time. Considering the significant number of tested strategies and combinations, we ultimately opted to use solely the ResNet-18 encoder for our experiments.

C. Training environment and Parameters

We use a NVIDIA Tesla V100 16 GB GPU, train for 200 epochs with Adam optimizer, a learning rate of 0.0001 and a standard step scheduler reducing the learning rate every 100 steps with a gamma of 0.5. To reduce the computational load we reduce the size of the images from their original resolution to 240×420 pixels.

D. Metrics

For a multi-task learning scenario involving semantic segmentation, depth estimation, and surface normal estimation, different metrics can be used to evaluate the model's performance on each of these tasks. For semantic segmentation, two commonly used metrics are mean Intersection over Union (mIoU) and pixel accuracy (pA). These metrics help to evaluate the accuracy of the model in segmenting objects and identifying their boundaries. For depth estimation, common metrics are absolute error (aE) and relative error (rE). Finally, for surface normal estimation, the mean angle error (mE) and percentage of points with an angle error less than 12.5, 22.5, and 30 degrees ($<$12.5, $<$22.5, $<$30 respectively) are used. Due to the multiple task and metrics per task we use the Δ_{MTL} metric [16] that combines one metric from each task to assess the overall performance of the multi-task learning model. Δ_{MTL} compares the depth aE, segmentation mIoU and surface normals mE errors against the corresponding values of the single trained networks, and produces a single metric defining the percentage of improvement. Negative values of Δ_{MTL} describe a network that is performing worse than the single trained networks.

E. Task Balancing

The need for task-balancing techniques arose due to the varying complexity and data availability of different tasks. If one task dominates the learning process, the model may not be able to learn the other tasks effectively, leading to sub-optimal performance. Therefore, task-balancing techniques have become crucial to achieving optimal performance in MTL scenarios. We test common loss-based weighting techniques, adding individual losses with equal weight (EW), dynamic weight averaging (DWA), uncertainty weighting (UW), geometric loss strategy (GLS) and random loss weighting (RLW). Additionally, we test gradient normalization (GradNorm), gradient surgery (PCGrad), conflict-averse gradient descent (CAGrad), gradient sign dropout (GradDrop) and gradient vaccine (GradVac) for gradient-based task balancing.

V. Experimental Evaluation

A. Single-task training results

We first trained three networks, each as a single-task learning process, focusing at the shared encoder and solely one branch at a time, consequently optimizing for each individual task. Table II displays the three single-task learning results, namely DeepLabv3-SingleS, DeepLabv3-SingleD and DeepLabv3-SingleN, for semantic segmentation, depth estimation and surface normal estimation respectively. The results of the single-task learning networks are used as baselines against which we evaluate the MTL strategies.

B. Multi-task task-weighting balancing results

With the aim of evaluating commonly employed methods, we initiated the MTL experimentation phase by weighting the losses of the individual tasks and subsequently the total loss of each experiment, with results reported in Table III. We tested the effectiveness of the learning process by training across all three tasks with an equal weighting strategy for each task-specific term of the loss (DeepLabv3-EW). The results of EW indicate that handling the task-specific losses equally does not enhance performance on individual tasks and leads to a decrease of 5.38% as determined by the corresponding

TABLE II: Baseline results for single-task trained network

Single-task								
Method	Semantic Segmentation		Depth Estimation		Surface Normal Estimation			
	mIoU↑	pA↑	aE↓	rE↓	mE↓	<12.5↑	<22.5↑	<30↑
DeepLabv3-SingleS	0.7936	0.9799	-	-	-	-	-	-
DeepLabv3-SingleD	-	-	0.2449	0.0404	-	-	-	-
DeepLabv3-SingleN	-	-	-	-	5.289	0.8647	0.9187	0.9389

TABLE III: Comparison of task-weighting balancing methods on the WarehouseSIM dataset

Multi-task									
Method	Semantic Segmentation		Depth Estimation		Surface Normal Estimation				Total
	mIoU↑	pA↑	aE↓	rE↓	mE↓	<12.5↑	<22.5↑	<30↑	Δ_{MTL}↑
DeepLabv3-EW	0.7812	0.9792	0.2463	0.0398	5.461	0.8609	0.9164	0.9372	-5.38
DeepLabv3-DWA	0.7791	0.9792	0.2455	**0.0390**	5.461	0.8615	0.9162	0.9368	-5.34
DeepLabv3-GLS	0.7862	**0.9798**	0.2578	0.0425	**5.354**	**0.8639**	**0.9177**	**0.9378**	-7.42
DeepLabv3-RLW	0.7783	0.9792	0.2487	0.0406	5.508	0.8603	0.9159	0.9368	-7.61
DeepLabv3-UW	**0.7884**	0.9796	**0.2467**	0.0411	5.378	0.8634	0.9173	0.9376	**-3.05**

TABLE IV: Comparison of gradient-based balancing methods on the WarehouseSIM dataset

Multi-task									
Method	Semantic Segmentation		Depth Estimation		Surface Normal Estimation				Total
	mIoU↑	pA↑	aE↓	rE↓	mE↓	<12.5↑	<22.5↑	<30↑	Δ_{MTL}↑
DeepLabv3-EW-GradNorm	**0.7866**	0.9792	0.2568	0.0447	5.319	0.8642	0.9180	0.9382	-6.31
DeepLabv3-EW-CAGrad	0.7777	0.9786	0.2535	0.0435	**5.301**	**0.8650**	**0.9188**	**0.9387**	-5.73
DeepLabv3-EW-GradDrop	0.7810	0.9792	**0.2437**	0.0404	5.526	0.8606	0.9159	0.9366	-5.58
DeepLabv3-EW-GradVac	0.7804	0.9792	0.2468	**0.0396**	5.455	0.8619	0.9164	0.9368	-5.58
DeepLabv3-EW-PCGrad	0.7782	**0.9792**	0.2463	0.0402	5.418	0.8632	0.9171	0.9374	**-4.95**

Δ_{MTL} metric. Dynamic weight averaging (DeepLabv3-DWA) gives very similar results with a best Δ_{MTL} of -5.34%. DWA requires a temperature parameter (T) to be defined, which controls the elasticity of weighting. The recommended value of temperature (T) in previous works is 2 [17], [18]. In our case, for T=2, we experienced low elasticity in task-weighting, as the three individual weighting values were confined to a narrow range of 10% throughout the training. In practice, due to the low elasticity, the DWA setting performed almost identically to the EW strategy, across all training epochs. We experimented with reducing the temperature parameter to increase the elasticity, and subsequently the allowed scaling of weights but got worse results. Geometric loss strategy (DeepLabv3-GLS) and random loss weighting (DeepLabv3-RLW) produced analogous results, worse than the aforementioned strategies (Δ_{MTL} of -7.42% and Δ_{MTL} of -7.61% respectively). Uncertainty weighting (DeepLabv3-UW) did enhance the performance significantly resulting in a Δ_{MTL} of -3.05%, yet not managing to take advantage of the added information of all tasks to produce a positive Δ_{MTL}.

C. Multi-task gradient-based balancing results

Next, we assess the effectiveness of gradient-based balancing methods and document the results obtained when employing these techniques on tasks that are weighted equally. Gradient normalization (DeepLabv3-EW-GradNorm) performs worse than the equal weighting baseline (DeepLabv3-EW)—Δ_{MTL} of -6.31% opposed to Δ_{MTL} of -5.38%—showing that normalizing the gradients on all tasks is not an effective strategy for our scenario. Similar results can be seen for conflict-averse gradient descent (DeepLabv3-EW-CAGrad), gradient sign dropout (DeepLabv3-EW-GradDrop) and gradient vaccine (DeepLabv3-EW-GradVac) with Δ_{MTL} of -5.73%, -5.58% and -5.58% respectively. CAGrad focuses on minimizing conflicts by updating gradients from only one task that has minimal interference with the others, GradDrop encourages diverse gradient directions through sparsity by randomly dropping the sign information of a portion of gradients, and GradVac injects task-specific gradient noise to promote positive transfer. The only gradient-based balancing technique that improves upon the equal weighting baseline is gradient surgery (DeepLabv3-EW-PCGrad)—Δ_{MTL} of -4.95% opposed to Δ_{MTL} of -5.38%—showing that projecting conflicting gradients to a mutually beneficial subspace does indeed benefit the learning process in our scenario. Although it is feasible to combine gradient-based balancing techniques with task-weighting balancing techniques, our experimental results did not demonstrate any notable enhancements across the tested combinations.

VI. Conclusion

In this work, we have presented a detailed analysis of common multi-task learning techniques in the new setting of mobile industrial robots. We start by producing a simulated warehouse dataset with color images and corresponding information for three tasks, i.e. semantic segmentation, depth estimation and surface normal estimation. The WarehouseSIM[1] dataset is made publicly available and can be used for both

[1]The dataset download link, source code and additional information on this project are available at: https://github.com/DTU-PAS/WarehouseSIM

single-task training and multi-task learning. Next, we test various task-balancing techniques used in computer vision. Throughout our experiments we demonstrate that these techniques fail to improve upon single-task learning, questioning their ability to be used in new scenarios, such as industrial mobile robot perception. Future work includes augmenting the synthetic dataset by incorporating additional scenes and introducing greater variability, leveraging novel aspects of the simulation platform like incorporating mobile human entities to enhance the scenario, and integrating supplementary tasks such as object detection, human pose estimation, and trajectory estimation. Finally, we plan on revisiting task balancing strategies to find solutions not tied to specific datasets and tasks that will consistently produce improved metrics across all tasks.

REFERENCES

[1] S. Cebollada, L. Payá, M. Flores, A. Peidró, and O. Reinoso, "A state-of-the-art review on mobile robotics tasks using artificial intelligence and visual data," *Expert Systems with Applications*, vol. 167, p. 114195, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742030926X

[2] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja, "Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues," *Array*, vol. 10, p. 100057, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590005621000059

[3] S. Garg, N. Sünderhauf, F. Dayoub, D. Morrison, A. Cosgun, G. Carneiro, Q. Wu, T. Chin, I. D. Reid, S. Gould, P. Corke, and M. Milford, "Semantics for robotic mapping, perception and interaction: A survey," *CoRR*, vol. abs/2101.00443, 2021. [Online]. Available: https://arxiv.org/abs/2101.00443

[4] S. Bultmann, R. Memmesheimer, and S. Behnke, "External camera-based mobile robot pose estimation for collaborative perception with smart edge sensors," 2023.

[5] L. Gao, J. Ding, W. Liu, H. Piao, Y. Wang, X. Yang, and B. Yin, "A vision-based irregular obstacle avoidance framework via deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 9262–9269.

[6] D. Arapis, M. Jami, and L. Nalpantidis, "Bridging depth estimation and completion for mobile robots reliable 3d perception," in *Robot Intelligence Technology and Applications 7*, J. Jo, H.-L. Choi, M. Helbig, H. Oh, J. Hwangbo, C.-H. Lee, and B. Stantic, Eds. Cham: Springer International Publishing, 2023, pp. 169–179.

[7] F. Graf, J. Lindermayr, C. Odabasi, and M. F. Huber, "Toward holistic scene understanding: A transfer of human scene perception to mobile robots," *IEEE Robotics and Automation Magazine*, vol. 29, no. 4, pp. 36–49, 2022.

[8] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2022.

[9] S. Vandenhende, S. Georgoulis, M. Proesmans, D. Dai, and L. V. Gool, "Revisiting multi-task learning in the deep learning era," *CoRR*, vol. abs/2004.13379, 2020. [Online]. Available: https://arxiv.org/abs/2004.13379

[10] T. Gong, T. Lee, C. Stephenson, V. Renduchintala, S. Padhy, A. Ndirango, G. Keskin, and O. H. Elibol, "A comparison of loss weighting strategies for multi task learning in deep neural networks," *IEEE Access*, vol. 7, pp. 141 627–141 632, 2019.

[11] D. Xin, B. Ghorbani, A. Garg, O. Firat, and J. Gilmer, "Do current multi-task optimization methods in deep learning even help?" *ArXiv*, vol. abs/2209.11379, 2022.

[12] A. R. Zamir, A. Sax, W. B. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," *CoRR*, vol. abs/1804.08328, 2018. [Online]. Available: http://arxiv.org/abs/1804.08328

[13] T. Standley, A. Zamir, D. Chen, L. J. Guibas, J. Malik, and S. Savarese, "Which tasks should be learned together in multi-task learning?" *CoRR*, vol. abs/1905.07553, 2019. [Online]. Available: http://arxiv.org/abs/1905.07553

[14] C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, and C. Finn, "Efficiently identifying task groupings for multi-task learning," *CoRR*, vol. abs/2109.04617, 2021. [Online]. Available: https://arxiv.org/abs/2109.04617

[15] A. R. Zamir, A. Sax, T. Yeo, O. F. Kar, N. Cheerla, R. Suri, Z. Cao, J. Malik, and L. J. Guibas, "Robust learning through cross-task consistency," *CoRR*, vol. abs/2006.04096, 2020. [Online]. Available: https://arxiv.org/abs/2006.04096

[16] S. Liu, S. James, A. J. Davison, and E. Johns, "Auto-lambda: Disentangling dynamic task relationships," *CoRR*, vol. abs/2202.03091, 2022. [Online]. Available: https://arxiv.org/abs/2202.03091

[17] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," *CoRR*, vol. abs/1803.10704, 2018. [Online]. Available: http://arxiv.org/abs/1803.10704

[18] B. Lin and Y. Zhang, "LibMTL: A python library for multi-task learning," *arXiv preprint arXiv:2203.14338*, 2022.

[19] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," vol. 7576, 10 2012, pp. 746–760.

[20] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge." *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijcv/ijcv88.html#EveringhamGWWZ10

[21] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," *CoRR*, vol. abs/1705.07115, 2017. [Online]. Available: http://arxiv.org/abs/1705.07115

[22] S. Chennupati, G. Sistu, S. K. Yogamani, and S. A. Rawashdeh, "Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning," *CoRR*, vol. abs/1904.08492, 2019. [Online]. Available: http://arxiv.org/abs/1904.08492

[23] B. Lin, F. YE, Y. Zhang, and I. Tsang, "Reasonable effectiveness of random weighting: A litmus test for multi-task learning," *Transactions on Machine Learning Research*, 2022. [Online]. Available: https://openreview.net/forum?id=jjtFD8A1Wx

[24] Z. Chen, V. Badrinarayanan, C. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," *CoRR*, vol. abs/1711.02257, 2017. [Online]. Available: http://arxiv.org/abs/1711.02257

[25] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *CoRR*, vol. abs/2001.06782, 2020. [Online]. Available: https://arxiv.org/abs/2001.06782

[26] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu, "Conflict-averse gradient descent for multi-task learning," *CoRR*, vol. abs/2110.14048, 2021. [Online]. Available: https://arxiv.org/abs/2110.14048

[27] Z. Chen, J. Ngiam, Y. Huang, T. Luong, H. Kretzschmar, Y. Chai, and D. Anguelov, "Just pick a sign: Optimizing deep multitask models with gradient sign dropout," *CoRR*, vol. abs/2010.06808, 2020. [Online]. Available: https://arxiv.org/abs/2010.06808

[28] Z. Wang, Y. Tsvetkov, O. Firat, and Y. Cao, "Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=F1vEjWK-lH_

[29] V. Kurin, A. D. Palma, I. Kostrikov, S. Whiteson, and M. P. Kumar, "In defense of the unitary scalarization for deep multi-task learning," *CoRR*, vol. abs/2201.04122, 2022. [Online]. Available: https://arxiv.org/abs/2201.04122

[30] A. Boulch and R. Marlet, "Fast and robust normal estimation for point clouds with sharp features," *Computer Graphics Forum*, vol. 31, no. 5, pp. 1765–1774, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2012.03181.x

[31] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. [Online]. Available: http://arxiv.org/abs/1706.05587

[32] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018. [Online]. Available: http://arxiv.org/abs/1802.02611

Artifacts Mapping: Multi-Modal Semantic Mapping for Object Detection and 3D Localization

Federico Rollo[†,‡,§], Gennaro Raiola[†,‡], Andrea Zunino[†,‡], Nikolaos Tsagarakis[‡], Arash Ajoudani[‡]

***Abstract*— Geometric navigation is nowadays a well-established field of robotics and the research focus is shifting towards higher-level scene understanding, such as Semantic Mapping. When a robot needs to interact with its environment, it must be able to comprehend the contextual information of its surroundings. This work focuses on classifying and localising objects within a map, which is under construction (SLAM) or already built. To further explore this direction, we propose a framework that can autonomously detect and localize predefined objects in a known environment using a multi-modal sensor fusion approach (combining RGB and depth data from an RGB-D camera and a lidar). The framework consists of three key elements: understanding the environment through RGB data, estimating depth through multi-modal sensor fusion, and managing artifacts (*i.e.*, filtering and stabilizing measurements). The experiments show that the proposed framework can accurately detect 98% of the objects in the real sample environment, without post-processing, while 85% and 80% of the objects were mapped using the single RGBD camera or RGB + lidar setup respectively. The comparison with single-sensor (camera or lidar) experiments is performed to show that sensor fusion allows the robot to accurately detect near and far obstacles, which would have been noisy or imprecise in a purely visual or laser-based approach.**

I. Introduction

To boost navigation autonomy and contextual awareness of mobile robots in unstructured environments, geometric information collected from the surroundings and the associated semantic data play key roles. The latter, in particular, includes qualitative environment information that can contribute to improving the robot's autonomy for navigation, task planning and manipulation, and simplifying human-robot interaction (HRI). This problem is tackled in the *Semantic Mapping* field, which aims to organize objects into classes and compute their pose and shape in a specific fixed reference frame. In this way, the environmental geometric information is supported by high-level features which increase the robot's awareness of the environment. In our specific case, we deal with the object detection and localization problem, which nowadays is widely investigated. For instance, in the last Darpa Subterranean Challenge[1], the main objectives were multi-robot exploration and object mapping in unknown environments, and the overall score was calculated based on the number of correctly detected and localized objects on the map.

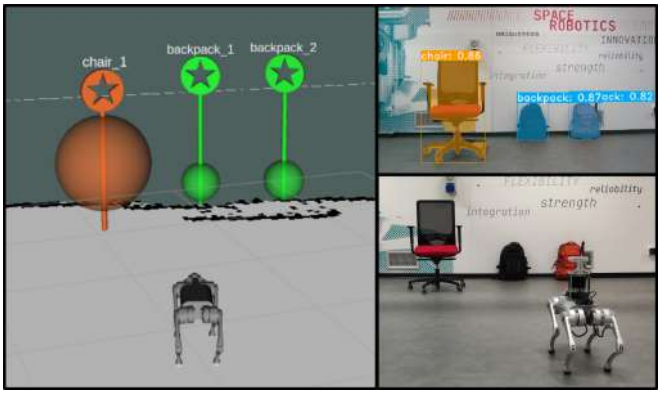


Fig. 1. An example of the framework during an experiment. On the left, is the visual application where objects are shown with a landmark and a spherical region of interest for the location in the Rviz visualization tool. On the top right, is the instance segmentation inference of the image taken from the robot camera while on the bottom right is the external representation of the experimental scene.

Different works were proposed to cope with the semantic mapping problem. Most recent results in robotics are facing the problem of using only RGB data and some interactive structures to be compliant with dynamic environments [1] while others rely on RGB-D data exploiting older algorithmic strategies (*e.g.* PnP algorithm) [2]. In autonomous driving, the RGB camera and lidar sensor fusion for semantic understanding is a currently tackled problem [3]. For a broader evaluation of the literature review see Sect. II.

Independently of the approaches used in robotics literature, the first thing which stands out is that most of them rely only on camera sensors. Cameras can give lots of dense information to the user especially if paired with depth data. However, their accurate depth range is within a few meters, leading to heavy depth measurement errors as the distances increase, especially if the robot is moving. This is particularly true for outdoor and vast indoor environments (*e.g.*, warehouses), where depth cameras are limiting and object semantic mapping remains a major challenge for far distances. In these cases, lidar sensors are an essential camera partner, allowing to have precise depth measurements for a wider distance range. Rather, in autonomous driving, the lidar and the RGB camera are nowadays commonly used but depth cameras are not considered due to their low resolution in the wide outdoor areas commonly faced in driving scenarios.

Another aspect not considered in most of the robotics examined works is that they do not account for limited re-

[†]Intelligent and Autonomous Systems, Leonardo Labs, Genoa, Italy
[‡]HHCM & HRII, Istituto Italiano di Tecnologia, Genoa, Italy
[§]Industrial Innovation, DISI, Università di Trento, Trento, Italy
Authors' e-mail: {*name.surname*}*.ext@leonardo.com*

[1]Darpa Subterranean Challenge: https://www.subtchallenge.com/

979-8-3503-0704-7/23/$31.00 ©2023 European Union

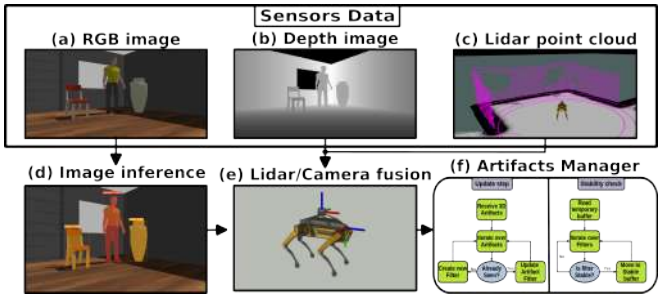


Fig. 2. This figure represents the whole Artifacts Mapping pipeline. The top block groups the sensors' data readings: (a) camera RGB image, (b) camera depth image and (c) lidar point cloud. At the bottom, there are (d) the RGB image inference performed with a Deep Neural Network for instance segmentation, (e) the multi-modal sensor fusion for detection and localization which uses as input the camera depth, the lidar point cloud and the Neural Network inference, and (f) a representation of the artifacts manager state-machines used to handle the sensor fusion detections and stabilize them.

sources applications which should run on embedded devices (*e.g.* Nvidia Jetson Nano[2]). Furthermore, Semantic Mapping is often used in the context of grasping or augmented reality scenarios while this work proposes an application for detecting and localizing objects (a.k.a artifacts) for high-level navigation tasks.

In our work, we aim to merge robotics and autonomous driving applications' strengths and present a modular architecture for semantic mapping[3]. We provide a multi-modal (camera-lidar) online semantic mapping framework which can fuse sensor information in real-time depending on the object distance and sensor's accuracies. We use image semantic information to enrich objects' filtered and stabilised positions to have precise object localization. The artifacts' shapes are simplified as spheres but they will be improved in future development. Our work relies on external geometric-based navigation frameworks such as SLAM algorithms or other localization algorithms (*e.g.*, AMCL [4]).

The proposed application demonstrates good accuracy for both near and far objects thanks to the camera-lidar depth fusion which, as far as the authors know, was not examined in other robotic or autonomous driving semantic mapping works. The application operates online also on low resources embedded systems (see Sect. IV-B) which strengthens the contributions of this paper. Moreover, we developed a Rviz[4] application which improves the user experience (UX) for visualization and interaction with the objects and the robot (see Fig. 1). The authors will provide on-demand a Docker application[5] as an added contribution, for running the artifacts mapping applications in simulation or on a robot (see Sect III-A-III-B and Sect. III-C).

The paper is divided as follows. In Sect. II a literature review of some recent works in semantic mapping is presented. The framework developed for this work is explained in Sect. III. We can distinguish the framework pipeline as two perception modules and a manager one. The first perception module performs 2D object detection while the second aims to estimate 3D artifacts position by fusing camera and lidar depth information (see Sect. III-A). The last module is needed to stabilize the perception estimations and to filter out noisy outliers (see Sect. III-B). An application of the presented framework (see Sect. III-C) is proposed based on two steps: *(i)* the robot can autonomously classify and localize objects on a map and save them in a specified format, *(ii)* the robot can load the artifacts as way-points on the map and the user can interactively select them to command the robot moving in that place to successively accomplish various kind of tasks such as manipulation, grasping, inspections or others. In Sect. IV the experiments to validate the framework are evaluated and discussed, and in Sect. V the conclusion and some future improvements are provided.

II. Related works

In literature, the semantic mapping problem was addressed using several approaches both in robotics and autonomous driving fields. Different surveys were presented that analysed this topic from various points of view. In [5] the authors explored the semantic mapping application in a human-robot collaboration scenario in an indoor environment while in [6] the semantic SLAM problem is presented in a general fashion analysing the works also in terms of perception, robustness, and accuracy. In [7], the less recent semantic mapping works are reviewed (*i.e.*, before 2014). This survey is a good reference to analyse the first development for the semantic mapping problem which yielded the more recent applications.

Among the modern semantic mapping approaches presented in robotics literature in the last decade, some first successful examples are [8] and [9]. In [8] the authors presented a monocular SLAM system that uses a SURF [10] feature extractor to check correspondencies and reconstruct the object's geometry. Instead, the authors in [9] showed an object-oriented 3D SLAM based on an ICP [11] object pose refinement and demonstrated that the introduction of semantic objects in the SLAM loop improves performances. the authors in [12] developed a monocular SLAM-aware object recognition system based on multi-view object proposals and efficient feature encoding methods giving as output a semi-dense semantic map. In [13] the authors proposed a framework which directly manages 3D objects. They use a Kinect[6] camera to reconstruct the 3D environment from different points of view and classify them while estimating their pose. In [14] the Data Associated Recurrent Neural Networks (DA-RNN) is introduced, which is an RNN for semantic labelling of RGB-D videos. The network output

[2]Nvidia Jetson Nano: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[3]Artifacts Mapping Youtube videos: https://www.youtube.com/playlist?list=PLdibjJfM06zugiWd-yUcdGH-SRWKTA3nQ

[4]rviz: http://wiki.ros.org/rviz

[5]The authors will grant access to a Docker image with the compiled application upon acceptance of the paper, based on the Freeware license.

[6]Microsoft Kinect camera https://en.wikipedia.org/wiki/Kinect

is fused with the KinectFusion algorithm [15] to merge semantic and geometric data. In [16] a Convolutional Neural Network (CNN) is used along with the ElasticFusion SLAM algorithm [17] to provide long-term dense correspondences between RGB-D video frames even in loopy trajectories. The authors in [18] leveraged ORB-SLAM2 [19] to reconstruct the geometric environment while using Single-Shot multi-box Detector (SSD) [20] along with an unsupervised 3D segmentation algorithm to place objects in the environment.

Moving towards more recent works, in [21] is presented the Contextual Temporal Mapping (CT-Map). They modelled the semantic inference as a Conditional Random Field (CRF) to account for contextual relations between objects and the temporal consistency of their pose. MaskFusion [22] is a real-time object-aware semantic and dynamic RGB-D SLAM algorithm. The greatest difference with respect to its predecessors is that it can cope with dynamic objects by continuously labelling them. Fusion++ [23] performs an object-level SLAM based on a 3D graph map of arbitrary reconstructed objects. They used RGB-D cameras, Mask-RCNN [24] instance segmentation and the Truncated Signed Distance Function (TSDF) to perform the semantic reconstruction. In [25] is presented an approach that incrementally builds a volumetric object-centric map with an RGB-D camera. They used an unsupervised geometric approach with instance-aware semantic predictions to detect previously unseen objects. They then associated the 3D shape locations with their classes if available and integrate them into the map. This approach has limited time performances to be used on a mobile robot because it runs at 1 HZ so it could be impractical in real-time. Conversely, in [26] the authors obtained a real-time dense reconstruction and semantic segmentation of 3D indoor scenes. They used an efficient super-voxel clustering method and conditional random fields (CRF) with higher order constraints from structural and object cues, enabling progressive dense semantic segmentation without any precomputation. The CRF infer optimal segmentation labels from the prediction of a deep neural network and runs in parallel with a real-time 3D reconstructor which utilizes RGB-D images as input. In [27] an open-source C++ library for metric-semantic visual-inertial SLAM in real-time is presented. They provide a modular code composed of a visual-inertial odometry (VIO) module, a pose graph optimizer, a 3D mesh-building module, and a dense 3D metric-semantic reconstruction module. The authors in [28], used a UAV equipped with a lidar, an RGB camera and a thermal camera to augment 3D point clouds and image segmentation masks while also generating an allocentric map.

One of the last available works which focus on this topic is [1] which presented a semantic mapping framework which uses only RGB data. They did not accomplish only object mapping but they provided a framework that can also distinguish different rooms and buildings. They exploited the 3D dynamic scene graphs [29] to abstract the different layers of inference (*i.e.* object, room and building), to solve problems such as loop closure detection and to cope with the mapping problem. Instead, the authors of [2] used RGB-D cameras to reconstruct an allocentric semantic map. They used a keypoint-based approach for pose estimation using a CNN keypoint extractor trained on synthetic data. Object poses were recovered from keypoint detections in each camera viewpoint with a variant of the PnP algorithm. The outputs obtained from the multi-camera system were then fused using weighted interpolation.

In autonomous driving, the multi-sensor fusion problem for 3D object detection is faced in [3] which uses lidar and RGB camera sensors to estimate the objects positions in the environment through ground estimation and depth completion. They use an end-to-end approach to train their multi-task network. The authors in [30] build a semantic map with a laser-based semantic segmentation of the point cloud not requiring any camera data. In [31], the authors provided a lidar-based SLAM for the geometric mapping and then use a CRF to fuse and optimize the camera semantic labels to obtain the semantic map. Instead, in [32], the camera and lidar data are used to build a probabilistic semantic octree map considering all the uncertainties of the sensors involved in the process. The authors in [33] presented one of the latest works in autonomous driving semantic mapping. They use an RGB camera and a lidar to perform semantic segmentation, direct sparse visual odometry and global optimization to include GNSS data in the mapping process.

Our review of the state-of-the-art indicated that most of the works on robotics platforms rely only on camera measurements and the experiments are limited to small indoor environments. Instead, in the autonomous driving scenario the camera-lidar fusion is already used for semantic tasks but they rarely use depth cameras, their lidars are generally more powerful (*i.e.*, they have 128-row lidars compared to the 16 ones commonly used in robotics) and they test the application in driving outdoor scenarios which offer different challenges with respect to robotic indoor once. Hence, with our work, we aim to stress the fact that RGB-D cameras and lidars are complementary sensors also in robotic semantic applications. For the semantic mapping application, we stated that with both sensors we can correctly localize objects at different distance ranges, improving detection accuracy.

III. Artifact Mapping Framework

In this section, the whole framework is presented as a conjunction of two blocks: Sect. III-A for object perception and Sect. III-B for object managing. In Sect. III-C the provided UI application is illustrated.

A. Artifacts detection and position estimation

The perception part can be conceptually divided into two components: *(i)* 2D object segmentation, *(ii)* 3D object position estimation using camera-lidar filtering.

1) 2D object segmentation: In this phase, a deep neural network [34] is used to infer from RGB images (see Fig. 2a) some predefined objects' classes and their masks. During the navigation, the robot takes pictures of the environment using the camera mounted on it. The pictures are passed

into an instance segmentation deep neural network which outputs the classification labels and masks (*i.e.*, a binary image having 1 where the object is found and 0 elsewhere) for each object recognized on the image (see Fig. 2d). The outputs are grouped and passed to the next module which will convert 2D data into 3D ones. An optional feature provided in this module is the possibility to filter out classes in real-time upon request. In this way, the robot can map different objects online depending on the requirements proposed. Other implementation aspects will be further explained in Sect. IV.

2) 3D object position estimation using camera-lidar filtering: This module fuses RGBD camera and lidar measurements to have a precise estimate of the objects' positions in the environment. The input is composed of the classification labels and masks found in the previous module, and depth information extracted from the camera (see Fig. 2b) and the lidar (see Fig. 2c). Sensors depth measurements are first analyzed separately in the following.

The depth image obtained from the camera (see Fig. 2b) is filtered using the recognized objects masks through element-wise matrix multiplication. The output, containing only the depth data of the object plus some sensor noise and environment outliers, is used to build a 3D point cloud projecting the 2D image points in the 3D space using the formula in the equation:

$$\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{p_x}{f_x} \\ 0 & \frac{1}{f_y} & \frac{p_y}{f_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} z_C \ , \tag{1}$$

where x_C, y_C, z_C are the 3D point coordinates with respect to the camera, u, v are the pixels on the image plane and f_x, f_y, p_x and p_y are the camera intrinsic parameters (focal distances and sensor's centre). Note that z_C is the depth measured by the camera depth sensor.

The obtained point cloud is filtered using a voxel grid downsampling filter[7] to reduce the number of points and, consequently, a radius outlier filter[8] is applied to remove the outliers induced by sensors noises and inference imperfections. The final point cloud is then used to compute the camera artifact centroid X_C as the mean of its points.

The 3D lidar centroid estimation is computed as follows. Projecting the 3D lidar points (see Fig. 2c) in the 2D detected masks images using Eq. 2, we are able to extract the object points of interest from the point cloud (*i.e.,* the points which have the 2D projection inside the mask).

$$z_L \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_L^C & T_L^C \end{bmatrix} \begin{bmatrix} x_L \\ y_L \\ z_L \\ 1 \end{bmatrix} \ , \tag{2}$$

where $R_L^C \in \mathbb{R}_{3x3}$ and $T_L^C \in \mathbb{R}_{3x1}$ are the rotation matrix and the translation vector between the lidar and the camera,

[7]voxel grid downsampling filter: https://pointclouds.org/documentation/tutorials/voxel_grid.html

[8]radius outlier removal: https://pointclouds.org/documentation/tutorials/remove_outliers.html

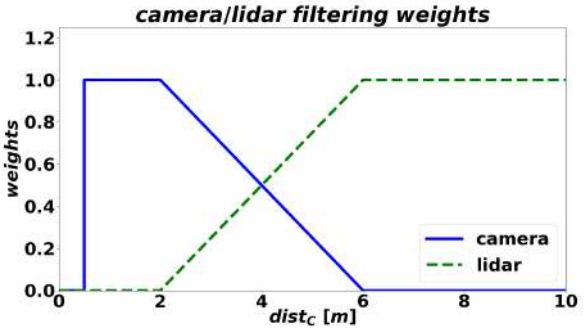


Fig. 3. An example of the contribution weights of camera and lidar for sensor fusion. The camera weight is in blue while the lidar one is in dashed green. In the specific example, we considered the specifications of a generic RGB-D camera you can find on the market: $min_C = 0.5$, $acc_C = 2.0$ and $max_C = 6.0$.

x_L, y_L, z_L are the 3D centroid position with respect to the lidar and the other parameters are the same of Eq. 1.

The extracted point cloud, representing the noisy artifact, will be then filtered using a radius outlier filter similar to the one used for the camera. Both radius filter parameters are directly dependent on the number of point cloud points because different distances and sizes of objects affect the point-cloud density and consequently the filtering. Finally, the mean of the point cloud is computed to obtain the lidar artifact centroid X_L.

Once both centroid measurements are available, they are fused in the artifact centroid X following the rules in the equation:

$$X = \begin{cases} 0 & \text{If } dist_C < min_C \\ X_C & \text{If } min_C \leq dist_C \leq acc_C \\ \xi X_C + (1-\xi) X_L & \text{If } acc_C \leq dist_C \leq max_C \\ X_L & \text{If } dist_C > max_C \ , \end{cases} \tag{3}$$

where $dist_C$ is the euclidean distance between the 3D point estimates and the camera, min_C and max_C are the minimum and maximum distances the depth camera can perceive, acc_C is the distance within which the camera can have accurate enough measurements to be used alone for the object localization (the camera information are generally provided by the sensors vendors), $X_L \in \mathbb{R}^3$ and $X_C \in \mathbb{R}^3$ are the lidar and camera 3D centroid estimates and $\xi \in [0,1] \in \mathbb{R}$ is the fusion weight represented by the blue slope of the segments between acc_C and max_C in Fig. 3 and it is computed as follows:

$$\xi = -\frac{1}{max_C - acc_C}(dist_C - acc_C) + 1 \tag{4}$$

Using the filtered camera and lidar point clouds, a rough 3D radius estimation ρ of the objects is performed. The camera radius ρ_C and the lidar radius ρ_L are computed as the mean of the two bigger dimensions along the X, Y and Z point cloud axis. the final radius ρ is computed following the same centroid fusion rules of Eq. 3 substituting X with ρ, X_C with ρ_C and X_L with ρ_L.

Also, the view angle ϕ of the artifact with respect to the robot is computed. Such an angle is rotated with respect to the map reference frame for implementation reasons with equation:

$$\phi = atan2(r_{21}, r_{11}) + atan2(y_r, x_r) \ , \tag{5}$$

where the r_{ij} is the entry at row i and column j of the rotation matrix $R_r^m \in \mathbb{R}_{3x3}$ between the map m and the robot r and x_r, y_r are the x, y positions of the artifact centroid with respect to the robot base. The two addends of Eq. 5 represent respectively the heading angle between the robot and the map and the angle between the robot and the 3D centroid.

B. Artifacts manager for data association

The manager (see Fig. 2f) is needed to filter out outliers and to stabilize artifact position estimations provided by the sensor fusion module. This process is generally known as data association[5][14]. The manager is composed of two modules: *(i)* object position filtering and *(ii)* object position stabilization which runs asynchronously in parallel.

1) Position filtering: Using a temporary data structure, the *temporary buffer*, we store and filter the perceived artifacts. Once the manager receives the 3D artifacts position estimations from the perception module (see Sect. III-A), it checks if the artifacts were already seen before (*i.e.,* the distance between one of the already seen artifacts and the current one is less than its 3D radius). If this is the case then the artifact in the temporary buffer is updated. Otherwise, for each not previously seen artifact received, the manager creates a new artifact instance in the temporary buffer. These instances have their own moving average filter which estimates the average of the artifact centroid position and its radius with Eq. 6 and computes a variance based on the distances between the position and the moving average in the filter horizon with Eq. 7.

$$\mu = \frac{1}{N} \sum_{\chi \in \Omega_N} \chi \tag{6}$$

$$\sigma = \frac{1}{N} \sum_{\chi \in \Omega_N} ||\chi - \mu||^2 \ , \tag{7}$$

where $N \in \mathbb{N}$ is the number of measurement in the moving average set Ω_N of 3D points, $\chi \in \mathbb{R}^3$ represent the current 3D position measurement, $\mu \in \mathbb{R}^3$ is the 3D mean position and $\sigma \in \mathbb{R}$ represent the variance of the filter.

2) Position stabilization: This module checks the stability of the artifacts in the temporary buffer and stores stable artifacts in another similar structure, the *stable buffer*. If an artifact in the temporary buffer is stable, the stabilizer moves the artifact from the temporary buffer to the stable one. An artifact is considered stable when its moving average filter variance σ is less than half its 3D artifact radius ρ and at least half the average filter set Ω_N is filled. This means that we have enough stable object position estimations and the object position average can be used for fixing the object position on the map.

At the end of the Artifacts Mapping application, an additional data association step is performed. The artifacts belonging to the same class which overlay each other on the XY plane are merged into a single artifact. This step reduces the duplicated object which sometimes appears on the map due to different point-of-view measurements and occlusions. After that, the stable artifacts buffer is saved in a yaml file which could be loaded into the user interface application presented in the next section.

C. User Interface for goal sending

A User Interface (UI) application based on a Rviz plugin (see Fig. 1) was developed to provide an intuitive visualization of the artifacts on the map, to send commands to the robot for moving near an artifact of interest and to delete artifacts which the user do not need or are wrong. Such artifacts can be loaded from the yaml file obtained with the artifacts mapping application. Through the UI application, the user can send *nav_msgs/goal* ROS messages which can be used by the robot to move towards the object (*e.g.*, using the ROS navigation stack as we do, see Sect. IV). The user can interact with the artifacts by simply right-clicking on them on Rviz and selecting the action *Go To* or *Delete*. Being the artifacts centroid position inside the artifacts shapes, the goal is moved in front of the artifact so that the robot stops before colliding with the object. The other available option is artifact deletion. If the user notices that an artifact is wrongly identified (classification or position) then the user can delete it and, once the UI application is closed, the loaded yaml file is updated with the remaining artifacts.

IV. Experiments

The experiments are performed both in simulation and using a real robot in a laboratory environment. The experimental setup is the same: some chosen objects are randomly positioned in the experiment area and the robot, following a predefined path, maps the predefined objects it encounters. This strategy is chosen because the objective is the validation of the artifacts mapping accuracy during an application, for example during a patrol. In other application scenarios, *e.g.* search and rescue, our framework could run in parallel with an exploration algorithm and the robot could trigger the exploration module every time an object of interest is encountered to obtain a precise localization.

In the experiments, we compare the data fusion with the mono-sensors application (i.e. using only an RGB-D camera or only the lidar) to demonstrate that the data fusion highly improves the detection accuracy and decreases the errors. For each environment setup, the experiments are repeated three times, one for each *sensors configuration*: only camera, only lidar, and both.

This work focuses only on semantic mapping and does not account for the robot localization which is assumed to be given. Additional errors in mapping resulting from localization are not considered in the final evaluation even if

TABLE I
DETECTION RESULTS OF THE SIMULATION AND REAL EXPERIMENTS.

	Simulation			Real		
	Camera	*Lidar*	*Fusion*	*Camera*	*Lidar*	*Fusion*
Correct detection	386	391	**416**	86	81	**99**
Wrong localization	12	13	**7**	10	14	**2**
Duplication	19	24	**15**	10	14	**6**
Wrong classification	**0**	**0**	**0**	7	11	**6**
Total detections	417	428	433	113	120	113
Total objects	422			101		

they negatively affect our application. Moreover, is important to notice that quadrupedal robots' movements are jerky and the sensors can suffer from that.

We set the parameters min_C, acc_C and max_C of Eq. 3 as 0.3, 4, 6 respectively based on the camera hardware information provided by the camera vendors (Intel Realsense).

The final validation performance is based on the number of objects which the robot can correctly find over the number of total objects. Also, the number of correctly-detected objects over the total number of detections is evaluated. The object is considered *found* if the difference between the estimated position and the real one is less than the real object radius and the associated class label is correct. The errors are categorized as duplicated objects, wrong localization and wrong classification. The duplications occur when there are more artifacts on a single object. they could be caused by the wrong artifacts radius computation due to occlusions or distinct point of view detection (*i.e.,* viewed from different perspectives: front and behind). The localization is considered wrong if the artifact's estimated position is outside the real object shape while the classification is erroneous if the artifact's class label is not correct.

For the simulation, the Whole-body Locomotion Framework (WoLF)[35] is used on a notebook with an *Intel® Core™ i9-11950H* processor and an *NVIDIA Geforce RTX 3080 Laptop* GPU. In the real scenario, a Unitree Go1[9] quadrupedal robot equipped with a RoboSense RS-Helios16 lidar[10], an Intel RealSense D455[11] and three Nvidia Jetson[12] (two Jetson Nano 4GB and one Nvidia Xavier NX) are used for the evaluation. The experiments are performed with the instance segmentation algorithms Yolact++ [34] and YolactEdge [36] trained on COCO [37] data set.

A. Simulation Experiments

Gazebo[13] simulator is used to simulate the robot in two different environments: the office[14] and Maze worlds where a predefined number of objects are positioned randomly at each iteration. The chosen objects for the simulation evaluation are *vase*, *couch*, *plant* and *person*. Specifically, in the office world, there are 5 vases, 12 couches, 6 plants and 11 persons while in the Maze world, there are 15 vases, 13 couches, 12 plants and 12 persons. The robot path is chosen randomly in advance using some waypoints on the map. In total, for each *sensors configuration*, 10 experiments were conducted, 5 for each environment, using different setups, for a total of 30 experiments.

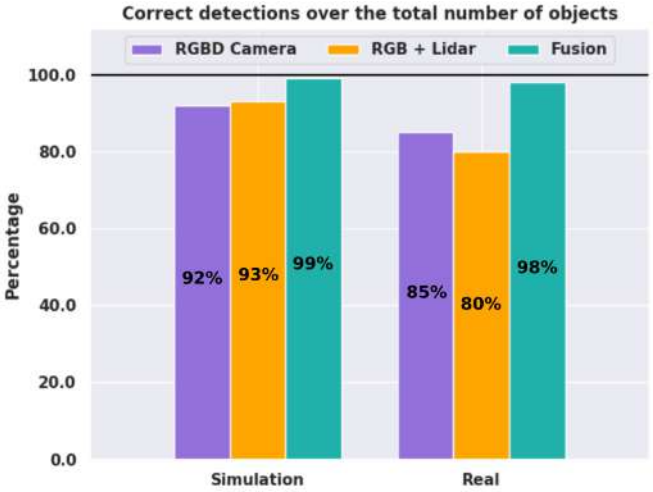


Fig. 4. Percentage of the correctly mapped and labelled objects concerning the total number of objects on the scene. On the left are the simulation results and, on the right, are the real experiments. Each block has three histograms representing the three *sensors configurations* used during the experiments: only RGBD camera, only RGB + lidar, and both.

The results of the simulation experiments are shown in the left part of Fig. 4 in terms of the number of correct detected objects. Specifically, considering the three ordered *sensors configurations* (*i.e.* only camera, only lidar, and both), we obtain the 92%, 93% and 99% of correctly localized and classified objects. Moreover, analysing the total number of detections produced, we obtain the distribution of the detections represented in the left column of Table I and the top part of Fig. 5 for the simulation experiment. Among all the detection produced, considering again in order the three *sensors configurations*, the 92%, 91% and 95% were correct while the remaining 8%, 9% and 5% of them were wrong.

The farthest object correctly detected in simulation during the camera-lidar sensor fusion experiments was at $15.47m$ from the robot, while the nearest was at $1.23m$.

[9]Unitree Go1: https://www.unitree.com/en/go1/
[10]RoboSense RS-Helios16: https://www.robosense.ai/en/rslidar/RS-Helios
[11]Intel RealSense D455: https://www.intelrealsense.com/depth-camera-d455/
[12]Nvidia Jetson: https://www.nvidia.com/it-it/autonomous-machines/embedded-systems/
[13]Gazebo simulator: https://gazebosim.org/home
[14]Clearpath robotics worlds: https://github.com/clearpathrobotics/cpr_gazebo/tree/noetic-devel/cpr_office_gazebo

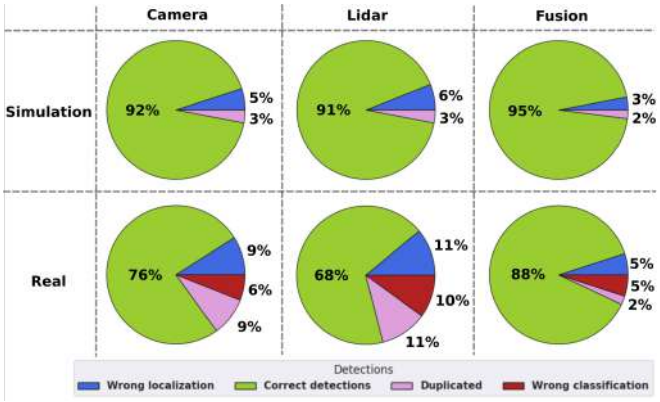


Fig. 5. Distribution of correctly and wrongly detected artifacts among the total generated detections. The pie charts represent the distribution of the correctly detected artifacts in green, the doubled objects in blue, the wrongly localized ones in pink and the wrongly classified ones in red. On the top row are the simulations results while on the bottom are the real ones. For each row, experiments are divided into three columns depending on the *sensors configuration* used during experiments: only camera, only lidar, and both.

B. Laboratory Experiments

The real experiments were carried out in a laboratory setting considering two scenarios, a one-room laboratory environment and a complete floor environment where the robot can move through corridors. In these environments were positioned *umbrellas*, *chairs*, *cabinets*, *backpacks* and *TVs* in variable amounts. For each *sensors configuration*, A total of 6 experiments were conducted, 3 for each environment, for a total of 18 experiments. For each trial, the objects were randomly moved and the illumination changed, *i.e.*, switching off lights or closing shutters.

The results of the laboratory experiments are shown in the right part of Fig. 4 in terms of the number of correct detected objects. Specifically, considering the three *sensors configurations* in order (*i.e.* only RGBD camera, only RGB + lidar, and both), we obtain respectively the 85%, 80% and 98% of correctly localized and classified objects. Moreover, analysing the total number of detections produced, we obtain the distribution of the detections represented in the right column of Table I and the bottom part of Fig. 5 for the real experiment. Among all the detection produced, the 76%, 68% and 88% were correct while the remaining 24%, 32% and 12% of them were wrong.

The farthest object correctly detected during the camera-lidar sensor fusion experiments was at a distance of $10.37m$ from the robot, while the nearest was at $0.98m$.

C. Discussion

The first thing to point out is that the farthest distances of the detected object were greater than $10m$ both in simulation and in real experiments. We take into account this distance to show a qualitative comparison between the lidar and RGB-D measurement in Fig. 6. The figure qualitatively upholds the thesis that a lidar sensor along with the camera is necessary to improve semantic mapping and, in general, other detection algorithms in wide areas. Moreover, from the results obtained

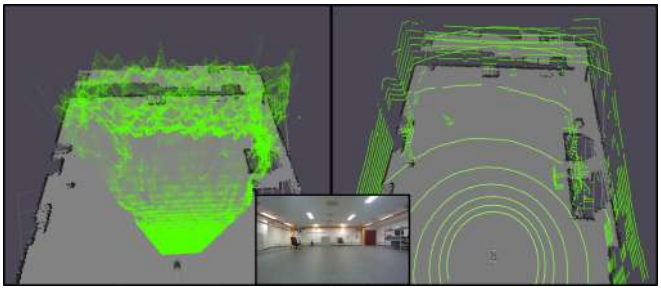


Fig. 6. Qualitative comparison between RGB-D camera (left image) and lidar (right image) point cloud detections at an approximate distance of $10m$ from the wall. At the bottom centre, there is the representation of the scene taken with the robot camera at that time instant. At large distances, the camera data are noisier and less accurate with respect to the lidar one but at small distances cameras provide a denser accurate point cloud while lidar data are sparser. From this comparison can be deduced that a visual-lidar sensor fusion can enhance semantic mapping.

from the experiments, it is clear that in our framework the use of both sensors improves the robustness of the application and decreases the detection errors. These improvements are less evident in a simulation environment where we used almost ideal sensors, *i.e.* the noise representation is not realistic as in Fig. 6. Still, it impacts real scenarios where there is more sensor noise.

The lidar can map far obstacles precisely while the camera introduces lots of errors at high distances. If we adopt only the camera, one solution to avoid erroneous measurements could be to not consider the depth measurement out of the accurate range guaranteed by the device specifications. By the way, by doing this the robot could miss some artifacts if it does not get close enough to them.

The camera, by providing more information at near distances with respect to the lidar, yields more precise centroid computations because it has fewer outliers than the lidar. Lidar outliers can be caused by wrong camera-lidar pose calibration and time synchronization which are essential for these applications especially when the robot moves fast. Instead, with RGBD cameras, the depth and the RGB images are synchronized in time and can be spatially superimposed almost exactly.

It is important to notice that wrong classification errors result from erroneous classifications in the pre-trained instance segmentation neural network which can be caused by illumination, reflections or other environmental conditions. They are here considered because the image inference is a module of the proposed pipeline but such errors can be decreased using more powerful neural networks.

V. Conclusion

We presented a framework which uses multi-modal sensors fusion to tackle the semantic mapping problem which is a rare setup in robotics applications. We fuse the lidar and RGB-D camera sensor readings to achieve better accuracy both for near and far objects as opposed to camera-only systems which lose accuracy for distant objects or lidar-only which lack high-level texture understanding of the environment.

We proposed a UI application to interact with the artifacts map obtained during the mapping application. This application is useful to perform autonomous high-level decision-making tasks because it exposes the object's class and location to the robot and the user.

The experiments showed that our application can correctly detect, localize and map the 98% of the objects present in the scene at different distances providing a small number of detection errors and good localization accuracy. The comparisons with the single-sensor scenario (only camera or only lidar) proved that sensor fusion is essential for wide areas and high-accuracy applications.

There are different future improvements we planned for this framework: (i) evolve the algorithm to an independent graph-based SLAM system, (ii) use 3D semantic point clouds with oriented bounding boxes and dimension information for better visualization and object understanding, (iii) deal with dynamics obstacle.

REFERENCES

[1] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3d scene graph construction and optimization," *Robotics: Science and Systems XVIII*, 2022.

[2] J. Hau, S. Bultmann, and S. Behnke, "Object-level 3d semantic mapping using a network of smart edge sensors," *arXiv preprint arXiv:2211.11354*, 2022.

[3] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[4] W. Xiaoyu, L. Caihong, S. Li, Z. Ning, and F. Hao, "On adaptive monte carlo localization algorithm for the mobile robot based on ros," in *Chinese Control Conf (CCC)*, 2018.

[5] A. Achour, H. Al-Assaad, Y. Dupuis, and M. El Zaher, "Collaborative mobile robotics for semantic mapping: A survey," *Applied Sciences*, 2022.

[6] L. Xia, J. Cui, R. Shen, X. Xu, Y. Gao, and X. Li, "A survey of image semantics-based visual simultaneous localization and mapping: Application-oriented solutions to autonomous navigation of mobile robots," *International Journal of Advanced Robotic Systems*, 2020.

[7] I. Kostavelis and A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, 2015.

[8] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. M. M. Montiel, "Towards semantic slam using a monocular camera," in *2011 IEEE/RSJ international conference on intelligent robots and systems*, 2011.

[9] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013.

[10] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, 2008.

[11] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, 1992.

[12] S. Pillai and J. Leonard, "Monocular slam supported object recognition," *arXiv preprint arXiv:1506.01732*, 2015.

[13] K. Tateno, F. Tombari, and N. Navab, "When 2.5 d is not enough: Simultaneous reconstruction, segmentation and recognition on dense slam," in *2016 IEEE international conference on robotics and automation (ICRA)*, 2016.

[14] Y. Xiang and D. Fox, "Da-rnn: Semantic mapping with data associated recurrent neural networks," *arXiv preprint arXiv:1703.03098*, 2017.

[15] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE international symposium on mixed and augmented reality*, 2011.

[16] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and automation (ICRA)*, 2017.

[17] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph," in *Robotics: Science and Systems*, 2015.

[18] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid, "Meaningful maps with object-oriented semantic mapping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[19] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, 2017.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*, 2015.

[21] Z. Zeng, Y. Zhou, O. C. Jenkins, and K. Desingh, "Semantic mapping with simultaneous object detection and localization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[22] M. Runz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2018.

[23] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level slam," in *2018 international conference on 3D vision (3DV)*, 2018.

[24] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask r-cnn," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[25] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto, "Volumetric instance-aware semantic mapping and 3d object discovery," *IEEE Robotics and Automation Letters*, 2019.

[26] Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Real-time progressive 3d semantic segmentation for indoor scenes," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[27] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[28] S. Bultmann, J. Quenzel, and S. Behnke, "Real-time multi-modal semantic fusion on unmanned aerial vehicles," in *2021 European Conference on Mobile Robots (ECMR)*, 2021.

[29] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans," *arXiv preprint arXiv:2002.06289*, 2020.

[30] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "Suma++: Efficient lidar-based semantic slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[31] J. Li, X. Zhang, J. Li, Y. Liu, and J. Wang, "Building and optimization of 3d semantic map based on lidar and camera fusion," *Neurocomputing*, 2020.

[32] J. S. Berrio, M. Shan, S. Worrall, and E. Nebot, "Camera-lidar integration: Probabilistic sensor fusion for semantic mapping," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[33] Q. Cheng, N. Zeller, and D. Cremers, "Vision-based large-scale 3d semantic mapping for autonomous driving applications," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022.

[34] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact++: Better real-time instance segmentation," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[35] G. Raiola, M. Focchi, and E. M. Hoffman, "Wolf: the whole-body locomotion framework for quadruped robots," *arXiv preprint arXiv:2205.06526*, 2022.

[36] H. Liu, R. A. R. Soto, F. Xiao, and Y. J. Lee, "Yolactedge: Real-time instance segmentation on the edge," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, 2014.

Dynamic Human-Aware Task Planner for Human-Robot Collaboration in Industrial Scenario

Alberto Gottardi[1,2,†], Matteo Terreran[1], Christoph Frommel[3],
Manfred Schoenheits[3], Nicola Castaman[2], Stefano Ghidoni[1], Emanuele Menegatti[1]

***Abstract*— The collaboration between humans and robots in industrial scenarios is one of the key challenges for Industry 4.0. In particular, industrial robots offer accuracy and efficiency, while humans have experience and the capability to manage complex situations. Combining these features can enhance the industrial process by avoiding the user manipulates heavy weights and allowing him to dedicate his efforts to tasks where flexibility, quality and experience make the difference in the final product. However, the collaboration between humans and robots raises several new problems to be addressed like safety, tasks scheduling and operator ergonomics. For example, human presence in the robot workspace introduces various elements of complexity into robot planning due to its dynamism and unpredictability. Planning must take into account how to coordinate the tasks between the robot and the human and be quick in re-planning to respond reactively to the operator's trigger. For this purpose, this work proposes a hierarchical Human-Aware Task Planner framework capable of generate a suitable plan to complete the process and manage user interrupts in order to have a constantly updated plan. The method is evaluated in a real industrial scenario and in a specific complex assembly task like the draping of carbon fiber plies.**

***Index Terms*— Human-Aware Task Planner, Human Action Recognition, Human-Robot Collaboration, Dynamic industrial scenario**

I. INTRODUCTION

Human-Robot Collaboration (HRC) in the industrial scenario is one of the most important technological challenges of recent years. The synergy between the robot's abilities, like precision, accuracy, efficiency and repeatability, along with human intelligence, flexibility and experience provides several advantages because it reduces the operator's effort and improves ergonomics during the operations, ensures the production quality and accuracy [1]. To be able to fully benefit from these advantages, while at the same time ensuring user safety when working with the robot, intelligent task coordination between humans and robots is required. A task planner takes over this intelligent coordination of activities. Moreover, this implies that it is necessary to use a planner that takes into account the user.

† Corresponding Author
[1] Intelligent Autonomous System Lab, Department of Information Engineering, University of Padova, 35131 Padua, Italy. gottardial, matteo.terreran, ghidoni, emg@dei.unipd.it
[2] IT+Robotics srl, 36100 Vicenza, Italy. nicola.castaman@it-robotics.it
[3] Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Augsburg, Germany. christoph.frommel, manfred.schoenheits@dlr.de
979-8-3503-0704-7/23/$31.00 ©2023 IEEE

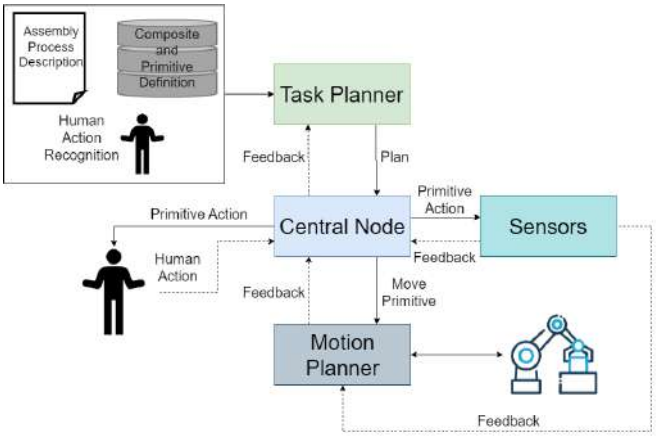


Fig. 1: Dynamic Human-Aware Task Planning Framework for Human-Robot Collaboration.

Since a shared industrial environment between humans and robots is a highly dynamic scenario, the classical task planner approaches are infeasible: they assume that the workspace is deterministic, the state is fully-observable, the robot is the only agent that can change the workspace, and actions are instantaneous. Therefore, to be usable in dynamic environments, task planners must deal with unpredictable and partially uncontrollable situations, especially due to human behaviour [2]. Several approaches have been investigated over the last few years to handle the dynamic scenario: from Artificial Intelligent Planning like PDDL [3] or Markov Decision Process [4] through Finite State Machine [5] to timeline-based approaches [6]. Multi-level programming [7] and Task Allocation [8] solutions are also very common approaches. Finally, game-theory and Reinforcement Learning (RL) models and methodologies are widely applied to multi-agent task scheduling problems [9], [10].

This paper proposes the Dynamic Human-Aware Task Planner framework for HRC in an industrial scenario summarized in Fig. 1. In particular, it focuses on the dynamic scheduling of shared human-robot activities within a manufacturing environment where humans and robots have to collaborate to complete complex tasks like object sorting, production line assembly [11] or draping [12].

Draping is one of the most complex operations in carbon fibre manufacturing. It is carried out by transporting the carbon ply onto the mould and adapting its shape to the mould. Nowadays, this process is completely manual and performed by expert human operators. In addition, the human operator is in charge of transporting the plies from a table

to the mould and then draping it. The EU project DrapeBot[1] aims at developing an HRC system capable of assisting an operator working on the draping of carbon fibre parts. In order to manage that process, a collaborative task planner must be used.

To address the requirements outlined above, we propose a hierarchical task planner that exploits the symbolic description of the process, the definition of Primitives and Composites actions and human commands to generate a suitable and continuously updated plan for the assembly process. In detail, our contribution follows:

- Dynamic Human-Aware Task Planner framework, which is able to compute human and robot activities and handle the user commands to update the plan.
- Using primitive actions to create more complex, so-called composite actions, which contribute to the creation of the final plan.
- Intelligent Action Recognition to trigger activities or robot behaviour not foreseen in the plan but which the expert user wants to perform.

II. RELATED WORKS

In recent years, task planning problems for HRC have been investigated. Existing works like [13], [14] tried to explore the knowledge encoded in the CAD model to extract the product's assembly sequence. Other works focused on sub-problems such as scheduling human and robot actions through Petri Nets [5], [15], or cooperative planning at a symbolic level [16], [17]. These approaches work better only in a classical static environment. Indeed, they cannot handle dynamism, uncertainty and the possibility of the user triggering unforeseen actions as a Human-Aware dynamic scenario requires and as proposed in our approach.

Nikolakis et al. [18] proposed a hierarchical method based on multi-criteria decision-making for an offline task allocation and a dynamic replanning due to unexpected events. Related works which used multi-criteria decision-making framework are [19], [20]. In these works, the authors considered robots and humans as resources. They developed task allocation approaches capable of handling unexpected events but not capable of handling specific commands/actions desired by the user. An unexpected event can be considered as a generic trigger where different events could correspond to a generic reaction. A user's command, instead, is a specific trigger, i.e., each command corresponds to a specific reaction. However, this part is crucial because the operator is an important subject inside the process, and his ability is fundamental to improve the process. Our method proposes solving this gap using the action recognition module connected to the task planner.

Graph-based approaches are described in [21], [22]. The modelling of the process takes place via AND/OR graph that can handle the parallelism of two actions assigned to two different resources. However, they cannot handle the order of precedence constraints typical of assembly tasks and how our approach aims to address and resolve. In other work, instead of using graph-based approaches, the authors exploit the advantages of the Behaviour Tree (BT) [23], [24]. In particular, Lamon et al. [24] have combined the BT approach with a Mixed-Integer Linear Programming (MILP) based role allocation method that allows individual and collaborative roles within the same formulation. However, human uncertainty is not modelled and considered. But, an intelligent system has to consider human intentions in its decision-making rather than force the operator to follow a strict, predefined assembly plan. In our proposed method, the operator can directly interact with the system and force the robot to execute some tasks by the action recognition module. Human intentions are typically modelled through the Partially Observed Markov Decision Process (POMDP) [25]. Approaches which shared similarities with Cramer et al. [25] modelled the collaborative task like hierarchical task network (HTN) [26], [27], [28]. The latter directly employs first-order logic to enable the robot to estimate its partner's goals and anticipate correctly in the presence of human variability and non-deterministic sensing. Another work related to the HTN is [29], where the task planner is able to divide the plan into multiple streams for multiple agents, humans included.

Most of the approaches described above have an implicit representation of the time. Actions are supposed to be instantaneous so that the action effects become true when the action itself is applied and changes the environment or the situation. Therefore, states and goals are not supposed to have a temporal extension such that they hold only for a limited temporal interval, or that they must be achieved within known temporal bounds. The planners that follow this approach are temporal planning, and the main feature is that they synthesize plans by combining causal reasoning with time and resource reasoning [30], [6].

In [6], Umbrico et al. proposed a timeline-based planner called PLATINUm with the ability to deal with temporal uncertainty at the planning and plan execution levels. The same authors then improved that tool by proposing an evolution of it, called TENANT [31], capable of setting objectives, defining tasks and establishing operational constraints, despite the inherent complexity required in planning and robotics. Although these works are evaluated in industrial applications, these approaches are not able to be adapted or rescheduled based on real-time observations by the operator. Indeed, adapting plans on the fly can be difficult, especially when the original plan heavily relies on strict time constraints like in these approaches.

Finally, significant advances in Deep Reinforcement Learning (DRL) have been witnessed in many outstanding large-scale sequential decision-making problems [10], [32]. Hu et al. in [33] exploited the combination of timed-place Petri nets with the deep Q-network with GCN to manage the dynamic scheduling problem of an industrial manufacturing scenario. In the same application, Kim et al. in [34] proposed the RL approach where intelligent agents evaluate the priorities of jobs and distribute them through negotiation.

[1]https://www.drapebot.eu/

Fig. 2: Example of a plan with composite and primitive actions.

III. SYSTEM ARCHITECTURE

This section presents the Dynamic Human-Aware Task Planner framework that handles the entire process and human-robot planning. Fig. 1 depicts an overview of the system proposed.

The Task Planner module coordinates the human and robot activities. It is responsible for creating a continuously updated plan that serves as a guideline for the workflow and will be composed of the sequence of actions to achieve the assigned task. Finally, the Task Planner must manage the human intentions to adapt the computed task plan to meet the collaboration needs dynamically or to handle unexpected situations and use recovery actions to return to a safe state. A simple example of the plan for a draping process is shown in Fig. 2 where it is composed of composite action like *Transport* that represents the activity to transfer the carbon fibre ply from a picking table to the mould, and primitives like *Draping* and *Inspection* that represent the actions performed by the human operator that drapes the ply into the mould and checks that no defects have formed during the previous activity.

The second important module is Action Recognition, which recognises the gestures associated with triggering specific actions. The associated command is sent to the Central Node, which is the module that monitors the operation of the system and sends the commands to the other modules that are in charge of performing the action in the plan. The Central Node manages the information the sensors provide in the workcell. Finally, a state-of-art Motion Planner is involved in order to generate a collision-free trajectory for the robot.

A. Task Planner

The task planner structure is outlined in Fig. 3 and is developed following a hierarchical approach consisting of 3 different layers:

- A low layer consisting of Primitive Actions.
- A middle layer consisting of Composite Actions.
- A high layer consisting of the Plan of the entire process.

Each layer has its own distinctive characteristics and a different level of abstraction with respect to the final task.
In the lower layer, we have the *Primitive actions* which represents the activities to be performed (e.g. *Move*, *Draping*, *Inspection*, etc.). The robot and the human alone could execute this activity, or both agents are required. A series of preconditions and effects characterise a primitive action. The preconditions are verified directly by the primitive itself, while the effects describe the state changes. When one of the preconditions is not satisfied, the current state is invalid and the primitive cannot be sent to execution, i.e. the related activity cannot be performed. Therefore, the primitive itself notifies the Central Node that the state is invalid. The central

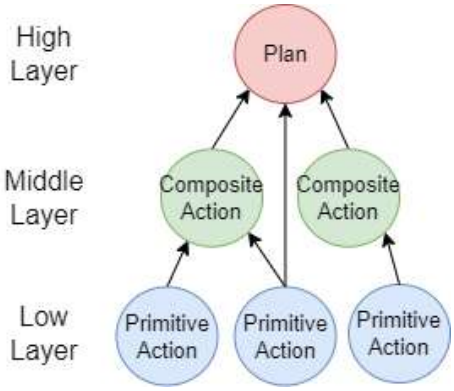


Fig. 3: Hierarchical structure of the Task Planner: the primitive actions are in the lower layer, the composite actions are in the middle layer and the final plan is in the highest layer.

node triggers the related recovery behaviour in order to return to a valid state. A recovery behaviour is a specific and simple action, strictly related to the primitive itself, that is responsible for returning the system to a valid state, thus allowing the process to continue while limiting external intervention to a minimum. For example, if a robot's gripper fails to grasp an object, recovery could be deactivating the gripper and retrying the grasping operation, i.e., reverting the state, performing the object detection again and re-evaluating the precondition primitives for the grasp action. Another more complex example could be the *Piece-Detection* primitive, which involves localizing the object on the pick-up table and providing a suitable grasping point. If the algorithm does not find the desired object, two recovery behaviours can be activated: the first starts a second scan of the table by moving the camera in a slightly different position; the second, if the process allows (i.e. without violating precedence constraints) searches for the next object to pick-up. The action associated to recovery behaviour is defined as a primitive, with its precondition (if needed) and effects. After the intervention of a recovery behaviour, it is verified whether the current plan is still valid and whether the preconditions of that action are now valid. If this happens, the primitive is re-executed, otherwise a re-planning is required. A set of specific configuration files defines the primitives and their structure.

In the middle layer, we have the *Composite actions* defined as a logical sequence of primitive actions. The composite has both preconditions and effects, corresponding to the first and last primitives, respectively. Similar to the primitives, the composite has a set of recovery behaviours triggered when a transition between one primitive and the next fails. The sequence of the primitives is defined in an external configuration by an expert operator who has to collaborate with the robot in the workcell. The definition of primitives and composite actions are provided in input to the Task Planner as shown in Fig. 1. The symbolic language used to model the primitives is the PDDL [35] because its action is precisely defined by a set of parameters, preconditions, and effects required by the Task Planner. Fig. 4 depicts an example of the *Transport* composite action which includes the primitives

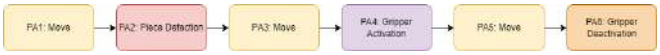


Fig. 4: Example of the Transport Composite Action (CA) as the sequence of Primitive Actions (PA).

Move, *Piece-Detection*, *Activation* and *Deactivation* of the gripper.

Finally, in the highest layer, we have the *Plan* for the entire process. By construction, this is the most abstract layer and where the definitions of composites and primitives are used together with information from the assembly process, environment and human operator to create the process plan. Assembly applications contain precedence constraints to manage and define the order in which the main components should be mounted. For these reasons, it was decided to use a Direct Acyclic Graph (DAG) approach with weighted arcs to represent all the possible alternative plans. In particular, each node of the graph represents an action (primitive or composite), while each arc represents the dependencies that must be fulfilled. For example, some objects must be placed before others in assembly tasks. Therefore, during the building of the graph, the task planner has to take into account that aspect. For example, as shown in Fig. 5, the *Action 5* must be performed only after *Action 3* and *Action 4*. In addition, each arc has an associated cost representing the effort of the robot and the user respectively in performing the action associated with the transition between the two nodes. The goal is to create a plan that minimises the user's effort and maximises the robot's effort by exploiting the possibility of having the robot perform some actions while the user performs others in the same collaborative workcell. Therefore, using the DAG (Alg1 - line 1) it is possible to find a topological ordering which describes the sequence of actions to complete the process (Alg1 - line 2). However, a DAG may contain more than one valid topological ordering. For this reason, the Depth-first search (DFS) algorithm was used for the topological search and optimised the cost function. The cost function used to calculate effort is the same for user and robot and it is the sum of the weights on the arcs in the DAG in Fig. 5. The mathematical formulation of the cost function follows:

$$C(w) = \sum_{u} w_{i,u} + \sum_{ru} w_{i,ru} - \sum_{u} w_{i,r} \qquad (1)$$

where $w_{i,j}$ represents the weight of i^{th} arc and j represents to which agent the weight is associated, whether to the user (u), the robot (r) or both (ru) when that action is to be performed by the two agents together. In order to optimize the cost function and obtain the plan that minimizes the user's effort, the $\arg\min_w C(w)$ is taken (Alg1 - line 3). In this way, a plan can be found to complete the process and described by the sequence of actions to be performed by the robot and user.

B. Human Action Recognition

The Human Action Recognition module monitors human activities during the collaboration, such as phases of the process (e.g., draping) or particular gestures to provide commands to the robot (e.g., request a new ply). Such information allows the task planner to be constantly updated on the current activities of the human operator: the task planner can periodically check whether the human is still engaged in particular tasks (e.g., draping) or whether through the use of gestures it is requesting specific actions from the robot that require the generation of a new task plan.

The human action recognition module is based on a previous work [36], where a graph convolutional neural network was proposed to recognize common human actions and gestures which arise in a collaborative manufacturing scenario. Such network takes as input a sequence of human 3D poses (i.e., skeletons) and tries to classify human movements according to a set of actions of interest by analyzing both spatial and temporal information.

In this work, human poses are provided by the state-of-

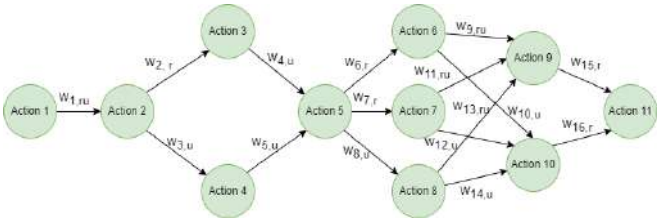


Fig. 5: Plan described by the Direct Acyclic Graph (DAG) where the action is represented by the node and the effort associated by the arc's weight.

Algorithm 1 Task Planner

Input: $\boldsymbol{PA}$ Primitive Action set, $\boldsymbol{CA}$ Composite Action set, $\boldsymbol{P}_d$ Process description, $state$ current state, a action failed, h human command
Output: $\mathcal{P}$ Plan
1: $\mathcal{G} \leftarrow buildDAG(\boldsymbol{PA}, \boldsymbol{CA}, \boldsymbol{P}_d, state, a, h)$
2: $< \mathcal{TP}, C(w) > \leftarrow findAllTopologicalOrdering(\mathcal{G})$
3: $\mathcal{P} \leftarrow \arg\min_w C(w) \in < \mathcal{TP}, C(w) >$
4: return $\mathcal{P}$

Algorithm 2 Central Node

Input: $\boldsymbol{PA}$ Primitive Action set, $\boldsymbol{CA}$ Composite Action set, $\boldsymbol{P}_d$ Process description, $\boldsymbol{H}$ human command set
1: $\mathcal{P} \leftarrow TaskPlanner(\boldsymbol{PA}, \boldsymbol{CA}, \boldsymbol{P}_d)$
2: **for all** $a \in \mathcal{P}$ **do**
3: $valid \leftarrow evaluatePrecondition(a)$
4: **if** $valid$ **then**
5: $state \leftarrow Execute(a)$
6: **else**
7: $(state, valid) \leftarrow RecoveryBehaviour(a)$
8: **if** $valid$ **then**
9: ***go to*** 3
10: **else**
11: $\mathcal{P} \leftarrow TaskPlanner(\boldsymbol{PA}, \boldsymbol{CA}, \boldsymbol{P}_d, state, a, null)$
12: **end if**
13: **end if**
14: $h \leftarrow HumanActionRecognition(), \quad h \in \boldsymbol{H}$
15: **if** h **then**
16: $\mathcal{P} \leftarrow TaskPlanner(\boldsymbol{PA}, \boldsymbol{CA}, \boldsymbol{P}_d, state, null, h)$
17: **end if**
18: **end for**

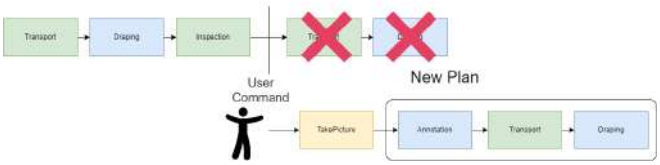


Fig. 6: Example of replanning due to user command.

the-art monocular 3D pose estimator MeTRAbs [37]. Such estimator outputs human poses composed of 19 keypoints describing the main body joints (e.g., torso, arms, legs).

Differently from [36], where actions were recognized using an ensemble of various models specialized for different body parts (e.g., body and hands), in this work we focused only on the body, as during a real manufacturing application the hands are thickly occluded and difficult to estimate accurately.

C. Central Node

The Central Node executes the plan by activating the correct primitives and supervising their execution. In addition, this module is always aware of the state of the workcell through the sensors present in the scene, e.g. a camera network positioned around the robot workcell, laser scanners, etc. A second purpose of the central node is to handle invalid state situations one may find oneself in during the execution of the process plan. When a primitive precondition is not satisfied (Alg2 - line 3), it is notified that the state is invalid and the central node will trigger the corresponding recovery behaviour in order to return to a valid state (Alg2 - line 7). When this happens, the primitive is asked to re-verify whether the preconditions are satisfied (Alg2 - line 9) in order to verify whether the current plan is still valid or if a re-planning is necessary (Alg2 - line 11). In the last case, the central node notified the task planner module that the current plan was unfeasible and a new plan was required. A similar situation occurs when the user wants to make the robot perform a task not foreseen in the plan (Alg2 - line 14). In this case, through the Human Action Recognition system, the user executes a specific command associated with a specific action to be done, be it a primitive or a composite. For example, as depicted in Fig. 6, when the *Inspection* action was finished, the user noticed some defects and decided to take a picture of the area where the defects were present. In order to perform this action, which was not included in the original plan, he performed the specific gesture associated with the composite *TakePicture*. The central node receives this information and sends it to the Task Planner module which is in charge of creating a new plan where the requested action is the first action to be performed. The requested action is treated as a precedence constraint to add to the DAG.

IV. EXPERIMENTS

The Dynamic Human-Aware Task Planner developed in this work was tested in a specific assembly scenario which is the draping of fibre carbon plies. First, we described the draping process and the actions involved, then we evaluated the performance of the Task Planner analyzing the computational time spent to create a suitable plan, considering also the replanning phase, and the performance of the Human Action Recognition system. Finally, a qualitative analysis in a real scenario is provided.

A. Case Study

Draping is a complex industrial operation that requires an advanced skilled user who is not only responsible for draping onto the mould but also for a series of activities such as inspecting the part, noting by text and/or photos of certain areas if they have slight defects, and checking that the orientation of the fibres is correct. The transport of a ply could be executed by the operator or robot alone, or it could be a collaborative transport where human and robot are involved. Thus, the coordination of the activities, like the robot's motion, activation/deactivation of the gripper to perform the pick and place and the detection of the piece in the picking table is crucial.

In addition, the operator could use a gesture to trigger an action which it was not in the original plan (Fig. 6) or to notify the central node that the current action has been completed and to move on to the next one. An example of this situation is when the operator has finished draping the ply and wants to notify the central node so that it can perform the next action. This is a simple and intuitive way for the user to interact with the robots and provide his/her experience into the system. Therefore, analyzing the process and the activities to be done, a set of gestures of interest has been defined based on the possible human interactions which can arise during the process. In particular, a set of 6 human actions has been considered: *Detection*, *Inspection*, *Transport*, *Draping*, *Drape Next* and *Take Picture*.

As described in the previous paragraph, a configuration file is used to represent the precedence constraints and define the order in which the plies are draped. Also, the sets of primitives, composite and gesture are defined in order to perform the entire draping process. Table I provides an overview of the actions used to evaluate the Task Planner, while a detailed list of the human actions of interest and their description is provided in Table II.

Action name	Description
Move	Primitive that represents the motion of the robot and/or the operator
Gripper Activation	Primitive that represents the activation of the gripper to pick up the ply
Gripper Deactivation	Primitive that represents the deactivation of the gripper to place the ply
Piece Detection	Primitive that represents the detection of the plies on the table
Draping	Primitive that represents the draping of the ply onto the mould by the operator
Transport	Composite that represents the transport of the ply from the picking table to the mould
TakePicture	Composite that represents the saving of an image of a certain mould area
Annotation	Composite that represents the saving of information by the user operator
Inspection	Composite that represents the inspection of the draping plies onto the mould

TABLE I: Overview of actions used to evaluate the Task Planner.

B. Task Planner Evaluation

The Task Planner was evaluated through the entire draping process using the actions shown in Table I and 20 independent process trials were performed. For each trial, the plan considered the draping of 5 plies. As mentioned above,

Action name	Meaning	Description
Detection	Trigger the Object Detection	Clap with stretched arms above head
Inspection	Stop current robot operation	Raise one hand with a stretched arm
Transport	Collaborative transport	Move while holding one side of the ply
Draping	Manual draping	Drape the ply on the mould
Drape Next	Require next ply	Move the right arm bend 90°
Take Picture	Take a photo of the ply status	Point at the desired location to be framed

TABLE II: Set of actions of interest considered for evaluating the human action recognition module in the proposed case study.

Trial	First Plan Time [ms]	Replanning Time [ms]
1	3.36	3.16
2	3.58	4.08
3	3.79	5.9
4	3.81	4.1
5	2.95	6.79
6	2.14	5.72
7	2.5	5.95
8	2.4	3.43
9	3.52	3.13
10	3.04	3.5
11	2.42	3.49
12	3.86	5.09
13	3.15	3.81
14	3.62	4.18
15	3.35	5.86
16	2.32	3.67
17	2.42	4.37
18	2.61	6.12
19	3.76	6.52
20	2.70	5.56
Average	3.065	4.7215
Validity	19/20	19/20

TABLE III: Time for the first plan (left) and average time for replanning (right) in ms.

the computational time is used to evaluate the performance and the interaction with the user was done by the Human Action Recognition system. Therefore, when the Central Node received the gesture it would either perform the next action in the plan or send a message to the Task Planner that a re-planning was necessary. Task Planner and Central Node were running in a Lenovo ThinkPad with 11^{th} Intel Core i7 processor and 16GB of RAM. The results obtained are summarized in Table III where the time is expressed in milliseconds (ms).

As shown in Table III, the planner demonstrated high efficiency in generating the first plan, with an average computational time of 3.065 ms. However, when the planner had to replan in response to a user's command, it was slightly slower, with an average computational time of 4.72 ms. The minimum and maximum computational times observed were 2.14 ms and 3.86 ms for the first plan, and 3.13 ms and 6.79 ms for the replanning phase, respectively. All the values in the replanning column are the average of all replanning that happened during the trial. In fact, in this way, it is possible to consider when the rescheduling has taken place. If you replan at the beginning of the process, there are a lot of tasks

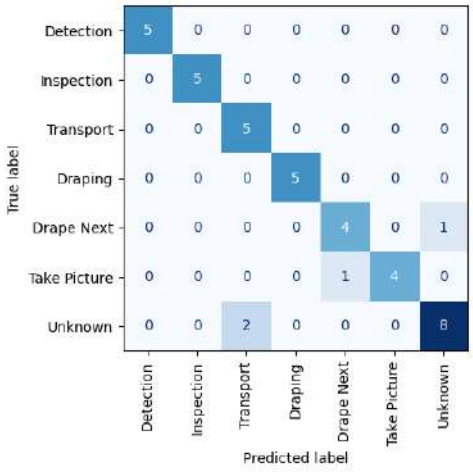


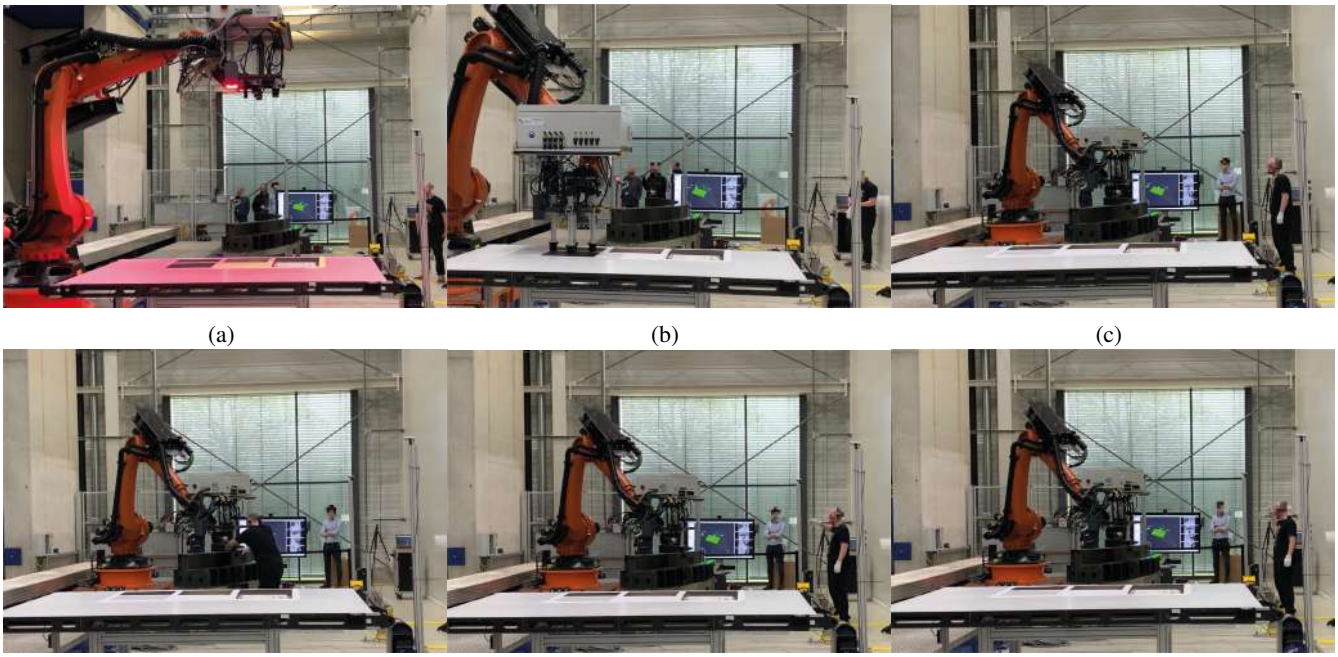
Fig. 7: Evaluation of the action and gesture recognition performance on the test set as confusion matrix.

afterwards, so it takes a lot of time; vice versa if you replan close to the end of the process, it will be faster because there are fewer tasks left to complete the process. Out of all the trials conducted, there was only one instance (Trial 13) where the Task Planner provided an invalid plan both as the first plan and during the replanning phase. The computational time was recorded in this case, and the trial was aborted. These findings highlight the efficiency of the planner in generating initial plans, with most plans being valid and feasible. The slight increase in computational time during the replanning phase suggests that the process of revising and generating a new plan takes slightly more time than the initial planning stage. Additionally, the occurrence of an invalid plan during replanning emphasizes the importance of thorough testing and verification to ensure the reliability and safety of the system.

C. Human Action Recognition Evaluation

For evaluating the Human Action Recognition module, a dataset has been collected for 5 different subjects, each one performing five times all the human actions considered in Table II. In order to improve the reliability of the action classifier, also a general "unknown" class has been considered in the dataset acquisition to learn better to distinguish the movements associated with the gesture from movements related to general movements of the worker within the workcell not related to the overall process as walking and standing. We considered 10 sequences for each subject for evaluating performance on the "unknown" gesture since it includes a larger variability of possible movements.

The action recognition module is trained on the collected dataset, using the sequence relative to 4 subjects. The sequences related to the fifth subject are reserved as a test set, on which the action recognition classifier is evaluated in terms of accuracy. This allows to evaluate the action classifier on a set of data not used to train the model, assessing the classifier's ability to generalize on novel data. The total number of test sequences is 40: five sequences for each of



(a) (b) (c)

(d) (e) (f)

Fig. 8: Evaluation during a draping task: the robot detects the required ply (a) and moves to the "pick" position (b); the operator waits until the robot reaches the final position on the mould (c) and then starts the manual draping phase (d); when draping is finished, the worker requests a new ply using a *Drape Next* gesture (e); while the robot starts moving towards a new ply, the human raise the right arm to trigger a manual "inspection" causing a replanning (f).

the six human actions defined in Table II, and ten sequences involving common movements annotated as "unknown".

The action recognition module has been evaluated in terms of Top1 and Top3 accuracy. The former represents the percentage of correctly predicted gestures in the test set. At the same time, the Top3 accuracy is the percentage of actions whose correct prediction falls in the three highest softmax scores estimated by the network. The performance achieved are Top1=90.00% and Top3=97.50%. As shown in the confusion matrix, Fig. 7, the classifier performs well in terms of accuracy on the considered test set since most of the actions of the fifth subject are correctly recognized.

D. Experimental Validation

The proposed framework has been validated in a real industrial scenario, targeting a collaborative draping task shown in Figure 8. In particular, the human operator and the robot work together to drape a series of plies on a mould: the robot provides the material transport and accurate placement on the mould (Figure 8c), while the human operator performs the actions that require high manual dexterity, such as manually draping the material over the mould (Figure 8d). At any time, the user can request particular actions from the robot by means of gestures, such as a request for a new ply (Figure 8e) or a request to stop the current operation to allow the operator to manually inspect the draping quality (Figure 8f). The proposed framework is able to monitor the worker's activity and handle sudden requests from the operator. For example, in the experimental validation shown in Figure 8, at the end of the manual draping, the operator makes a *Drape Next* gesture, thus triggering the task planner to generate a plan to move the robot towards a new ply; immediately afterwards, when the robot starts to move, the user makes a new *Inspection* gesture forcing the task planner to delete the previous plan and generate a new one.

V. CONCLUSIONS

In this paper, we proposed a Human-Aware Task Planner for Human-Robot Collaborative industrial applications. The advantages of this approach are the ability to create a plan starting from the description of the process and the actions in order to share that activities both from humans and robots. In addition, it is able to handle user interaction through the dynamic rescheduling of the plan following user interruptions or to handle unexpected events. The user commands are handled by an intelligent Human Action Recognition module based on Deep Learning technique. Another main contribution is the ability of the planner to create actions starting from primitives. The framework has been validated in an industrial collaborative scenario derived from the DrapeBot European research project. The results obtained demonstrate the applicability and effectiveness of the proposed approach. In future works, we plan to integrate the Task Planner with an ergonomic Motion Planner in order to evaluate the complete Task and Motion Planner (TAMP) application in a dynamic industrial scenario where one of the collaborative activities between robot and human is the collaborative transport of plies.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 101006732, "DrapeBot – COLLABORATIVE DRAPING OF CARBON FIBER PARTS".

REFERENCES

[1] G. Chryssolouris, N. Papakostas, and D. Mavrikios, "A perspective on manufacturing strategy: Produce more with less," *CIRP Journal of Manufacturing Science and Technology*, vol. 1, no. 1, pp. 45–52, 2008.

[2] S. Pellegrinelli, A. Orlandini, N. Pedrocchi, A. Umbrico, and T. Tolio, "Motion planning and scheduling for human and industrial-robot collaboration," *CIRP Annals*, vol. 66, no. 1, pp. 1–4, 2017.

[3] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, and M. Carreras, "Rosplan: Planning in the robot operating system," in *Proceedings of the international conference on automated planning and scheduling*, vol. 25, 2015, pp. 333–341.

[4] B. Lacerda, D. Parker, and N. Hawes, "Optimal and dynamic planning for markov decision processes with co-safe ltl specifications," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1511–1516.

[5] V. A. Ziparo, L. Iocchi, P. U. Lima, D. Nardi, and P. F. Palamara, "Petri net plans: A framework for collaboration and coordination in multi-robot systems," *Autonomous Agents and Multi-Agent Systems*, vol. 23, pp. 344–383, 2011.

[6] A. Umbrico, A. Cesta, M. Cialdea Mayer, and A. Orlandini, "Platinu m: A new framework for planning and acting," in *AI* IA 2017 Advances in Artificial Intelligence: XVIth International Conference of the Italian Association for Artificial Intelligence, Bari, Italy, November 14-17, 2017, Proceedings 16*. Springer, 2017, pp. 498–512.

[7] M. S. Malvankar-Mehta and S. S. Mehta, "Optimal task allocation in multi-human multi-robot interaction," *Optimization Letters*, vol. 9, pp. 1787–1803, 2015.

[8] F. Chen, K. Sekiyama, F. Cannella, and T. Fukuda, "Optimal subtask allocation for human and robot collaboration within hybrid assembly system," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 4, pp. 1065–1075, 2013.

[9] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[10] T. Yu, J. Huang, and Q. Chang, "Optimizing task scheduling in human-robot collaboration with deep multi-agent reinforcement learning," *Journal of Manufacturing Systems*, vol. 60, pp. 487–499, 2021.

[11] L. Tagliapietra and E. Tosello, "Curami: human-robot collaboration for intelligent assembly tasks."

[12] M. Terreran, S. Ghidoni, E. Menegatti, V. Enrico, P. Nicola, N. Castaman, A. Gottardi, E. Christian, V. Luca, S. Giuseppe, *et al.*, "A smart workcell for cooperative assembly of carbon fiber parts guided by human actions," in *Atti della 4ª Conferenza Italiana di Robotica e Macchine Intelligenti*, 2022.

[13] M. H. Arbo, Y. Pane, E. Aertbeliën, and W. Decré, "A system architecture for constraint-based robotic assembly with cad information," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2018, pp. 690–696.

[14] G. Michalos, J. Spiliotopoulos, S. Makris, and G. Chryssolouris, "A method for planning human robot shared tasks," *CIRP journal of manufacturing science and technology*, vol. 22, pp. 76–90, 2018.

[15] A. Casalino, A. M. Zanchettin, L. Piroddi, and P. Rocco, "Optimal scheduling of human–robot collaborative assembly operations with time petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 1, pp. 70–84, 2019.

[16] Y. Cheng, L. Sun, and M. Tomizuka, "Human-aware robot task planning based on a hierarchical task model," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1136–1143, 2021.

[17] A. Ham and M.-J. Park, "Human–robot task allocation and scheduling: Boeing 777 case study," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1256–1263, 2021.

[18] N. Nikolakis, N. Kousi, G. Michalos, and S. Makris, "Dynamic scheduling of shared human-robot manufacturing operations," *Procedia CIRP*, vol. 72, pp. 9–14, 2018.

[19] P. Tsarouchi, G. Michalos, S. Makris, T. Athanasatos, K. Dimoulas, and G. Chryssolouris, "On a human–robot workplace design and task allocation system," *International Journal of Computer Integrated Manufacturing*, vol. 30, no. 12, pp. 1272–1279, 2017.

[20] P. Tsarouchi, S. Makris, and G. Chryssolouris, "On a human and dual-arm robot task planning method," *Procedia CIRP*, vol. 57, pp. 551–555, 2016.

[21] L. Johannsmeier and S. Haddadin, "A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 41–48, 2016.

[22] K. Darvish, E. Simetti, F. Mastrogiovanni, and G. Casalino, "A hierarchical architecture for human–robot cooperation processes," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 567–586, 2020.

[23] F. Rovida, B. Grossmann, and V. Krüger, "Extended behavior trees for quick definition of flexible robotic tasks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6793–6800.

[24] E. Lamon, F. Fusaro, E. De Momi, and A. Ajoudani, "A comprehensive architecture for dynamic role allocation and collaborative task planning in mixed human-robot teams," *arXiv preprint arXiv:2301.08038*, 2023.

[25] M. Cramer, K. Kellens, and E. Demeester, "Probabilistic decision model for adaptive task planning in human-robot collaborative assembly based on designer and operator intents," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7325–7332, 2021.

[26] R. Caccavale and A. Finzi, "Flexible task execution and attentional regulations in human-robot interaction," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 9, no. 1, pp. 68–79, 2016.

[27] J. Cacace, R. Caccavale, A. Finzi, and R. Grieco, "Combining human guidance and structured task execution during physical human–robot collaboration," *Journal of Intelligent Manufacturing*, pp. 1–15, 2022.

[28] M. Rizwan, V. Patoglu, and E. Erdem, "Human robot collaborative assembly planning: An answer set programming approach," *Theory and Practice of Logic Programming*, vol. 20, no. 6, pp. 1006–1020, 2020.

[29] R. Lallement, L. de Silva, and R. Alami, "Hatp: hierarchical agent-based task planner," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018)*, 2018.

[30] S. Stock, M. Mansouri, F. Pecora, and J. Hertzberg, "Online task merging with a hierarchical hybrid task planner for mobile service robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 6459–6464.

[31] E. Foderaro, A. Cesta, A. Umbrico, and A. Orlandini, "Simplifying the ai planning modeling for human-robot collaboration," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 1011–1016.

[32] H. Oliff, Y. Liu, M. Kumar, M. Williams, and M. Ryan, "Reinforcement learning for facilitating human-robot-interaction in manufacturing," *Journal of Manufacturing Systems*, vol. 56, pp. 326–340, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612520301084

[33] L. Hu, Z. Liu, W. Hu, Y. Wang, J. Tan, and F. Wu, "Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network," *Journal of Manufacturing Systems*, vol. 55, pp. 1–14, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612520300145

[34] Y. G. Kim, S. Lee, J. Son, H. Bae, and B. D. Chung, "Multi-agent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system," *Journal of Manufacturing Systems*, vol. 57, pp. 440–450, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612520301916

[35] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, D. Christianson, *et al.*, "Pddl— the planning domain definition language," *Technical Report, Tech. Rep.*, 1998.

[36] M. Terreran, M. Lazzaretto, and S. Ghidoni, "Skeleton-based action and gesture recognition for human-robot collaboration," in *Intelligent Autonomous Systems 17: Proceedings of the 17th International Conference IAS-17*. Springer, 2023, pp. 29–45.

[37] I. Sárándi, T. Linder, K. O. Arras, and B. Leibe, "Metrabs: metric-scale truncation-robust heatmaps for absolute 3d human pose estimation," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 3, no. 1, pp. 16–30, 2020.

Decentralized Market-Based Task Allocation Algorithm for a Fleet of Industrial Mobile Robots*

João Tavares[1], Alberto Vale[2] and Rodrigo Ventura[3]

***Abstract*— In this paper, we present an efficient, resilient, and flexible market-based task allocation algorithm with a distributed architecture for a dynamic factory environment. The proposed algorithm provides efficient and intelligent task allocation mechanisms that reduce the time and total distance traveled by the agents.**

This algorithm is implemented in a simulation environment that is similar to a real-world environment with various robots and tasks to allocate to test its efficiency, resilience, and flexibility. It is compared quantitatively with other baseline solutions such as auction only with available robots and a queue system.

The results show that the algorithm is more efficient than the other methods tested. It is also reliable since it can handle unpredictable behaviors such as corrupted messages, loss of connection for an extended period, failures to complete tasks, and obstacles blocking the robot's path and forcing them to take a different trajectory. Finally, it is flexible since it can be used for several different purposes and is robust to communications failures. Also, this algorithm possesses the drawback of being ill-equipped to manage a substantial influx of task requests, given that solely a single task is auctioned and assigned at any given time.

I. Introduction

The problem of coordinating robots efficiently and reliably to work together has a wide application in robotics and multi-agent systems. It is one of the main developments needed to update factories to industry 4.0. Efficiency, in this case, means taking the shortest time and distance to execute a task. And reliably means that the system will not stop working if either one or more robots have technical issues or there is a problem with the communication infrastructure. Several task allocation algorithms have been created in recent decades to solve this problem for different environments, such as manufacturing, warehouse management, hospital management, and rescue missions.

This work is part of a research and development project for Industry 4.0 called AGiLE, jointly with Imeguisa, Instituto Superior Tecnico (IST), and Volkswagen Autoeuropa (AE), and is focused on developing an intelligent system of autonomous mobile robots (AMRs) for logistics on the AE factory floor.

*This work was supported by: Aero.Next Portugal

[1]João Tavares is with the Department of Electrical and Computer Engineering, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal joao.n.c.a.tavares@tecnico.ulisboa.pt

[2]Alberto Vale, Instituto de Plasmas e Fusão Nuclear, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal avale@ipfn.tecnico.ulisboa.pt

[3]Rodrigo Ventura, Institute for Systems and Robotics, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal rodrigo.ventura@isr.tecnico.ulisboa.pt

979-8-3503-0704-7/23/$31.00

This project is necessary for AE since the last few years mobile robots called Automated Guided Vehicles (AGVs) have transported mobile storage containers of items throughout the factory. These robots use a simple, yet commonly used guidance system that can only follow a magnetic line on the floor, like trains on rail tracks. In this system, if robots find an obstacle in their path, they will not be able to continue their task until the object is removed, usually by a human. Meaning that if production stops, robots will pile up in front of the delivery area with a full rack, causing an unnecessary traffic jam and possibly blocking other AGVs from executing their task. Often, the system that controls the robots has a centralized architecture that is neither efficient nor resilient because the whole system stops working if there is a problem with either the central computer or the communication network. In short, this system is cost-effective and simple but has inherent disadvantages that potentially make it less reliable.

The system developed in this R&D project must be able to assign transportation requests (tasks) provided by the containers to each robot, compute the optimal trajectory, detect obstacles, recompute the trajectory of each robot if necessary without any human intervention, and work in a non-centralized fashion. The containers used in this project will be smarter than the previous ones since they have a computer that monitors how many items are left and informs the network when it needs to be transported to a different location in the factory. This system is more complex than the one currently operating in the factory, but it should be more efficient and resilient, as explained before, and flexible. Flexibility means that the system can be easily adapted to handle different types of allocation methods, can handle heterogeneous tasks and robots, and a variable number of active robots, and does not require a powerful communication infrastructure.

This paper places its main emphasis on the task allocation mechanism, specifically the utilization of a sequential decentralized market-based approach. The algorithm implemented is sequential, allowing for the allocation of one task at a time. Furthermore, it is decentralized in nature, as all computational resources and decision-making capabilities are equally distributed amongst the robots within the network. Lastly, this mechanism employs a market-based approach to task allocation, thereby facilitating the even distribution of computational power throughout the network of robots.

II. Related Work

To the best of the authors knowledge, the first decentralized auction algorithm was developed in [1] and [2] in 1979 and 1989, respectively. Since then, several decentralized and distributed auction algorithms were developed. A Market-based Multi-Robot Task Allocation via Strategic Pricing algorithm is presented in [3] with a decentralized architecture and the possibility to auction many tasks at once. Murdoch, presented in [4], has a central auctioneer and several heterogeneous agents, which means that only those capable of executing the task in auction will bid on it. This algorithm can only allocate one task at a time. Also, the communication infrastructure requirements are reduced since the number of messages and the size of each message is almost negligible. Finally, this algorithm monitors the task progress and auctions the task again if there is a problem. Consensus-based Auction Algorithm (CBAA), presented in [5], [6], and [7], has a distributed architecture and uses a consensus algorithm to ensure only one robot will execute each auctioned task. Many tasks can be auctioned at once but only robots not executing any task can participate in the auction. The alliance task allocation algorithm was developed in [8] and [9]. It has a behavior-based fully distributed architecture. It can only allocate one task at a time with the option of re-allocation when other agents' motivation, more specifically their impatience and acquiescence, reach a certain level. This makes this algorithm resilient. However, it was only designed for small to medium size fleets. The Consensus-based Parallel Auction and Execution (CBPAE) was introduced in [10] and is similar to the CBAA since it has an auction and a consensus phase and has a distributed architecture. Also, it has the ability to work with heterogeneous agents and tasks, bid and execute tasks in parallel and prioritize tasks. This algorithm, presented in [11], works in a distributed fashion with a network of heterogeneous agents. M+ algorithm presented in [12] has a distributed architecture suited for cooperative missions where task reallocation is possible and agents have reasoning, decision, and reactive capabilities. The Sequential Single-Item Auctions algorithm was proposed in [13]. This algorithm is a robust solution for obtaining the shortest total distance traveled by agents.

III. Task Allocation Algorithm

A. Formulation

Formally there is: τ to represent time; a set of N Autonomous Mobile Robots (AMRs), denoted $R = \{r_1, ..., r_N\}$; a set of Q containers, denoted $P = \{p_1, ..., p_Q\}$, where a container is, as explained in Section I, a movable storage unit with a computer that monitors how many items are left and informs the network when it needs to be transported to a different location in the factory; a set of M tasks, denoted $T = \{t_1, ..., t_M\}$, where each task t_i is defined by 5 attributes (Task ID, Container ID, Start and Finish coordinates (location), Deadline) and consists of a request by the containers to be transported to a predetermined location; $b_{r,t}(\tau)$ to represent the bid robot r did on task t at instant τ; $c_r(\tau)$ to represent the time needed for robot r to finish the current task at instant τ; $n_{r,t}$ to represent the time needed for robot r to execute task t.

We consider the following assumptions: 1) robots are identified by their unique ID; 2) each robot r can only be in one of three states, $s_r(\tau) = \{\text{Available, Occupied, Offline}\}$; 3) a robot is: a) available if it is online and not executing any task; b) occupied if it is online and executing a task; c) offline if it is not connected to the network or has a technical problem; 4) only robots available and occupied participate in auctions; 5) robots have a distributed repository of information with the current task in auction, every robot's current state, bids, and the task being performed; 6) robots are homogeneous, meaning that every tasks can be executed by any robot as long as they are not offline; 7) each robot can only execute one task at a time and can not have any task allocated while performing another; 8) each task can only be executed by one robot.

The bid computation formula is contingent upon the environmental factors under which it is implemented. Hence, for the purposes of the experiments described in this paper, the bid for each task shall be determined by considering the duration required to arrive at the task's starting location during the auction, as well as the task currently being performed. The equation used to calculate is the following:

$$b_{r,t}(\tau) = \begin{cases} n_{r,t}, & \text{if } s_r(\tau) = \text{Available} \\ c_r(\tau) + n_{r,t}, & \text{if } s_r(\tau) = \text{Occupied} \end{cases} \quad (1)$$

The main objective is to find an optimal allocation of robots to tasks. The allocation is done sequentially, one task at a time. An allocation is a set of robot-task pairs (r_i, t_j). The optimal robot i for task j is the one that has the smallest bid:

$$r_i(\tau) = \arg\min_v (b_{v,j}(\tau)) \quad (2)$$

B. Methodology

The algorithm will use a distributed architecture and will be inspired by three algorithms, already explained in Section II, CBAA, Murdoch, and CBPAE. This algorithm provides efficient and intelligent task allocation mechanisms that reduce the time and resources needed in environments with a high number of unpredictable obstacles that might force the robots to take longer to execute their task. Also, it is resilient since it does not fail if a robot has an unexpected problem or there is a communication issue. Finally, this algorithm is flexible since it can be easily adapted to different environments. In Figure 1 a flowchart of the task allocation algorithm is presented.

When a container requests a new task, all robots add the task to a local priority queue, where the highest priority task is the one with the closest deadline. Then, if there is not an auction procedure happening, the robot with the lowest ID starts the auction procedure for the highest priority task. To establish a synchronized and orderly auction process, robots engage in periodic message transmission among themselves. These messages serve two crucial purposes: firstly, to signal

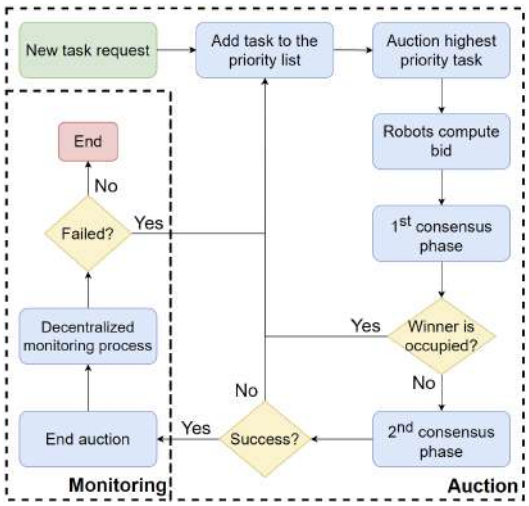


Fig. 1. Flowchart of the auction and monitoring algorithms.

the online status of each robot, and secondly, to enforce the condition that only one agent initiates the auction process at any given time, thereby ensuring the occurrence of a singular auction event at a time. By implementing this communication mechanism, the system effectively prevents simultaneous or overlapping auctions, promoting a streamlined and coherent execution of the auction protocol.

After the auction procedure starts, every available and occupied robot compute their bids using equation (1) and sends them to all other robots participating in the auction. After the robots receive all the bids or a specific time interval has passed, the first consensus phase starts. In this phase, each robot compares its bid with all the others and chooses a winner, just like in CBAA [6]. This consensus phase does the same as equation (2), where the robot with the smallest bid value gets chosen as the winner. It is important to note that if the robot with the lowest bid is already executing a task, the task in auction will not be allocated. This practice is undertaken for two primary purposes. The initial rationale is to enforce the restriction that robots cannot be assigned multiple tasks concurrently. In the event that a robot, engaged in a particular task, experiences an unforeseen delay, any subsequent task(s) awaiting execution by this robot would likewise suffer an unnecessary delay. Secondly, this practice aims to prevent scenarios wherein a robot, nearing completion of its ongoing task, abstains from participating in the auction process. Should this robot abstain, the auctioned task could potentially be allocated to a different robot that, theoretically, would require a longer duration to execute said task.

If the winner already has a task allocated to it, the second highest priority task is auctioned to avoid having the algorithm always trying to allocate the same task and failing consecutively. In the event that the allocation of a task fails due to the fact that the winning candidate has already been allocated, the algorithm proceeds to retry the allocation of the task with the highest priority, and repeats this process iteratively until the task has been successfully allocated. Then, to ensure that every agent reaches the same conclusion and that only one robot will execute the task, a second consensus phase is performed where robots share their winner decision with all the others participating in the auction and compare them. If the consensus is successful, the task gets assigned, and the auction finishes. However, if the consensus is unsuccessful, the task in auction does not get assigned, goes back to the priority queue, and the process repeats until either this task gets allocated or a higher priority task is requested, making it the new task in auction, just like in CBPAE [10]. This second consensus phase might seem redundant, but it guarantees that only one robot will execute the task currently in auction and only requires a reduced amount of computational and communication resources to execute it.

Upon the successful completion of an auction process, all robots within the network engage in the monitoring of the progress made by the robot that has been allocated the task. Like Murdoch [4], robots periodically try to communicate with the assigned robot. In cases where the allocated robot does not respond, the robot is informed that he is no longer executing this task and the task returns to the priority queue. The container will request to be transported from its current location to the final location of the task.

Regarding the communication infrastructure prerequisites, the auction mechanism necessitates the exchange of information among robots. This communication is facilitated through ROS (Robot Operating System) Topics, wherein robots transmit messages containing essential details such as bids, decisions made by winners, and task specifications. These messages primarily consist of integers and a concise assortment of integers and floats to represent the specifications of each task under auction. Furthermore, during the monitoring phase, robots engage in periodic message exchanges akin to ”ping” signals, ensuring continuous connectivity and communication across the entire network of robots.

IV. Simulation Setup

In order to test the proposed approach, a simulation environment was developed using *ROS Noetic* and *Python3*. A *Gazebo* world was built to generate a map with walls and corridors for robots to circulate. Figure 2 illustrates the map used in all simulations. Robots are represented with orange rectangles, and their idle positions, i.e., where robots go when they do not have any task assigned, being located in the red squared room. The tasks paths are shown in yellow. Each task starts at the green circle and finishes in the red circle. So, when a container requests a task, it means that this container is located at the start location of that task. After a task is allocated, the robot travels to the start location of the task, transports the container to the task destination, and [30; 60] seconds after the task was successfully executed, the container requests to be transported back to its initial position, and the process repeats. Each container requests a total of 20 tasks per simulation and each simulation is repeated 10 times with a different order of tasks being

requested at the start, to ensure consistency in the results. Blue rectangles with a cross are dynamic obstacles that are introduced to block corridors, at random moments for a period between [10; 20] seconds, and force robots to recompute their trajectories and take a different route. Only one zone can be blocked at a time to ensure that robots can always get to their destination.

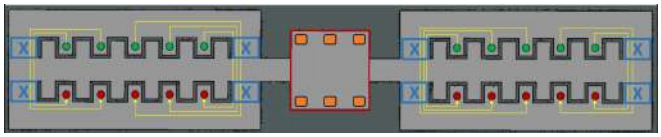


Fig. 2. Map used in simulations. Robots are represented with orange rectangles, and initially located at their idle positions, in the red squared room. Tasks are drawn in yellow. Each task starts at the green circle and finishes in the red circle. Blue rectangles with a cross are zones that are temporarily blocked.

The robots used in simulations have similar kinodynamics to the robots used in real-world applications.

V. Experiments Performed and Results

Since the objective of the Proposed Algorithm (PA) is to be resilient, efficient, flexible, and decentralized, several simulations were performed with a different number of robots and containers. The tests performed will now be described:

A. Resilience

1) Setup: To assess the resilience of the PA, different simulations were performed where randomly chosen robots were temporarily turned off, had their communications obstructed during the auction period and while executing tasks, or were forced to fail a task execution to ensure that the algorithm handles all types of failures. These failures were supposed to prompt the algorithm to fail in several areas such as bid computation, sending or receiving a message with the bid or a winner decision, reaching a consensus on either phase, having more than one robot execute a task, and failing on reallocating a task after the assigned robot failed. All of the failures mentioned were predicted based on the typical behaviors of networks and computers.

2) Results: The algorithm consistently assigned tasks to only one robot and never experienced prolonged stalls in the auction process, even when messages were deliberately blocked or robots changed online/offline status. This is due to the built-in mechanisms within the algorithm that prevent these types of failures. One of the mechanisms used to avoid allocating more than one robot to a task is the interruption of the auction procedure if the robots do not reach a consensus on the winner. Another mechanism for the same purpose is used by robots during an auction where they are constantly checking if the current auction is still on. If it is not, they leave the auction and wait for another task to be auctioned to ensure that they do not send their bids or results to a different auction and allocate the wrong robot. Situations like this will happen when either the bid computation fails or takes too long or when a robot loses connection for an extended period. To prevent the algorithm from getting stuck due to unresponsive robots, in case a robot goes offline or takes an excessive amount of time to answer during an auction, other robots will detect the lack of response and disregard it until the auction concludes. Finally, in situations where robots have unexpected issues while executing a task, other robots will notice and the robot with the lowest ID will add the task back to the priority list. Robots are constantly checking for the robots online so, they always know which robot has the lowest ID. In every environment, the time needed to auction a task was between 1.5 and 15 seconds, depending if there is a communication problem. Because, after each auction phase, a robot only goes to the next phase when all the others reach its phase or, in cases a robot fails, a wait time of 5 seconds has expired to avoid the auction getting stuck or taking too long unnecessarily. In the simulations performed, it is observed that the bid computation time takes around 0.1 seconds. The PA is resilient in environments with a variable number of active robots, bad communication infrastructure, and failures in the execution of the task.

B. Efficiency

1) Setup: In order to analyze the efficiency of the PA, several simulations in different environments were performed and compared with other baseline solutions that compute their bids in the same way as the proposed model but have different criteria to choose the winner of an auction namely:

Auction only with available robots (AOA): Only robots that are not executing any task participate in the auction, and the robot with the lowest bid wins, meaning that the auction will always be successful but the most efficient robot allocation might not happen;

Queue system (QS): All robots participate in the auctions, and the winner is the robot with the lowest bid. However, if this winner is occupied, the task gets added to its queue, which means that the robot will execute it right after finishing the current task.

These algorithms were simulated in environments with 4 robots and 10 containers with and without dynamic obstacles and in environments with 6 robots and 10 containers with and without dynamic obstacles. For every simulation, several statistics were obtained namely the time needed to execute a task since it was requested by the container and the time needed to execute the task since it was allocated. These statistics will be used as metrics to evaluate the algorithm's efficiency and capability to handle unforeseen path changes due to, for example, obstructed corridors.

2) Results: In Figures 3, 4, 5 and 6 the simulation results in environments with and without dynamic obstacles and with different numbers of robots are presented. In each figure, three distributions are shown for each container in the horizontal axis—one for each allocation method. The vertical axis represents the time taken to execute the set of tasks since they were triggered. For each distribution, the median, minimum, maximum, and quartiles time are highlighted. As explained before, each container requests a total of 20 tasks per simulation, and each simulation is repeated 10 times

to ensure consistency in the results. This means that each distribution has a total of 200 values. In all distributions, for each container, the minimal time taken to execute each task is very similar. This has to do with how the simulation starts since the tasks are all triggered simultaneously and auctioned in random order. The first tasks to be auctioned will always have a low execution time.

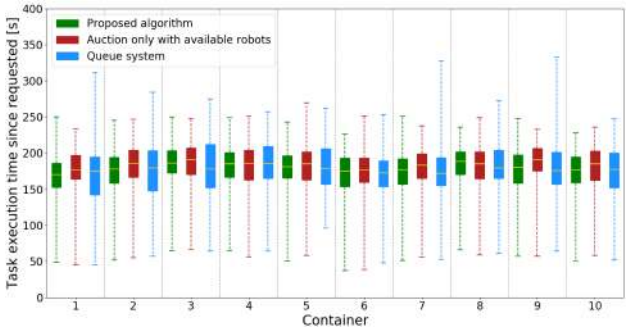


Fig. 3. Results with 4 robots and 10 containers without dynamic obstacles.

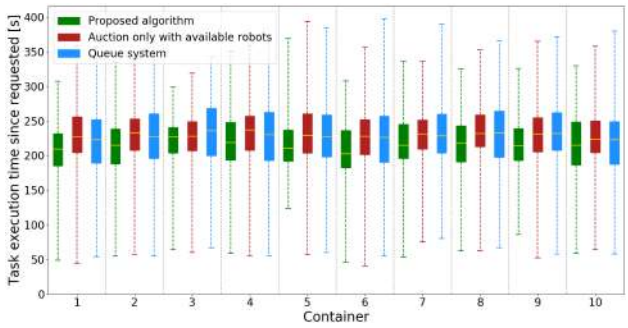


Fig. 4. Results with 4 robots and 10 containers with dynamic obstacles.

In an environment with 4 robots and 10 containers and without dynamic obstacles, Figure 3, the PA takes almost the same time to execute the set of tasks as the QS and slightly less time, around 2%, than the AOA method. With dynamic obstacles, Figure 4, the proposed model is significantly faster than the QS and the AOA model, around 6% for both.

In an environment with 6 robots and 10 containers and without dynamic obstacles, Figure 5, the PA takes the same time to execute the set of tasks as the QS and slightly less time than the AOA method (1%). With dynamic obstacles, Figure 6, the proposed model is slightly faster than the QS and the AOA model, around 3% and 4%, respectively.

3) Results Analysis: The AOA takes more time to execute the tasks than the proposed model in all tested environments. Because, in many cases, robots that were closer to the start of the task but were still finishing another one would not be able to participate in the auction. A typical example of this situation is shown in Figure 7. In this case, there are 3 robots executing task 2 (T2), task 3 (T3), and task 4 (T4). Task 1 (T1) was requested but is not yet allocated and there is one robot in its idle position. When T1 is auctioned, only the robot in the idle area (red square) will participate in the

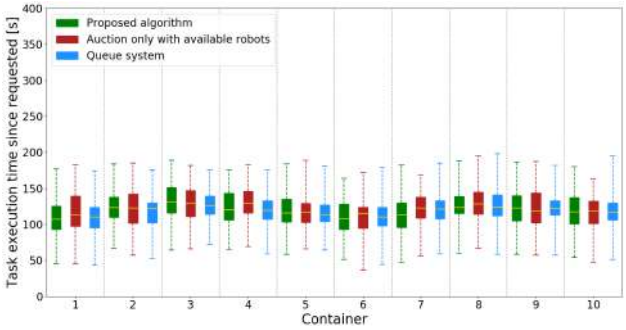


Fig. 5. Results with 6 robots and 10 containers without dynamic obstacles.

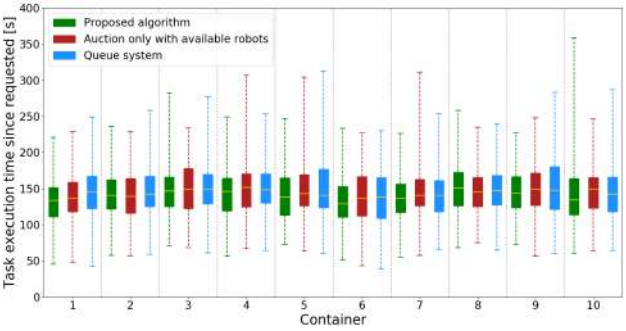


Fig. 6. Results with 6 robots and 10 containers with dynamic obstacles.

auction and, therefore, get the task allocated to him. The other robots did not have the possibility to participate in the auction, even those that were almost finished executing their task and that would probably execute the task in auction faster. The PA solves this problem by always having all the robots in the network participate in the auction and if the winner is occupied, the task gets re-auctioned as it was explained in III-B.

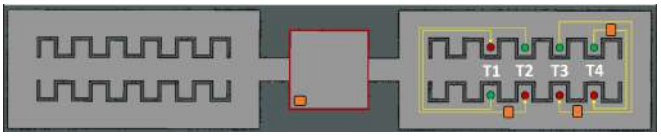


Fig. 7. Example of a state where AOA is inefficient.

The QS presents similar execution time results in environments without dynamic obstacles to the proposed model. However, in environments with dynamic obstacles, the results are worse than the proposed model. In fact, when the robot has to change its path due to unforeseen obstacles blocking the corridors, the execution time of its current task and, therefore, all the others in its queue will increase unnecessarily. An example of this situation is shown in Figures 8 and 9.

In this example, two possible paths exist to execute T3, but the robots always choose the shortest path. In the first state, Figure 8, the robot executing T3 has not noticed that there is an obstacle blocking the corridor. In this state, an auction is started for T4, where all robots bid. Since the robot executing T3 thinks it is the closest to being able to execute T4, this

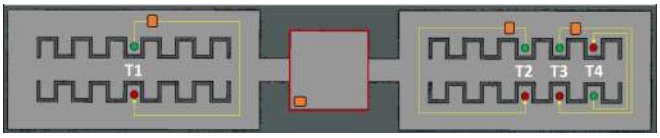


Fig. 8. Before the robot executing T3 detects the blocked corridor.

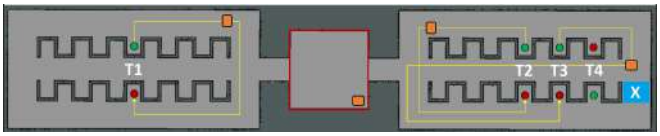


Fig. 9. After the robot executing T3 detects the blocked corridor.

task gets added to its queue. After the robot identifies the obstacle, a new path is calculated, as it is shown in Figure 9. This means it will take more time than expected to finish T3 and start T4. If the PA had been used, the auction for T4 in the first state would have failed and restarted as many times as needed until the auction winner was available. This means that when the robot executing T3 noticed the obstacle, T4 would have been attributed to the robot in the idle area. So, in a dynamic environment, where obstructions occur, the PA is better suited than the QS.

C. Flexibility

The PA is also flexible and can be used in different types of scenarios and robots models since the bid computation algorithm and the task allocation restrictions, used in the efficiency experiments performed are easy to implement with very few changes needed in the code.

Furthermore, this algorithm is suitable for environments where robots have different capabilities and can't perform all tasks. This is because the messaging system employed by the containers to request tasks consists of several attributes that are accessible to all robots, enabling them to determine their eligibility to participate in an auction and execute a task. These attributes can also be used to organize the order in which tasks are auctioned and therefore handle tasks with different levels of priority.

Finally, since the task allocation algorithm proposed does not require a powerful communication infrastructure, as explained before, it can easily be implemented in most infrastructures.

VI. Conclusion

This paper proposes a task allocation algorithm for a fleet of heterogeneous robots operating in factory floor environments, where obstacles may affect the path execution and communication. This algorithm is inspired by state-of-the-art task allocation algorithms developed in the past like CBAA, Murdoch, and CBPAE. It has a distributed architecture, meaning that the computation and decision-making are distributed between all the agents involved.

The proposed task allocation algorithm is implemented and tested in simulation environments with a variable number of robots, containers, and tasks to evaluate its efficiency and resilience. Different task allocation methods are used as baselines, such as AOA, and a QS to compare the efficiency results with the proposed method.

To evaluate the efficiency of the model, two metrics are taken into account: the time taken to execute each task after it was triggered, and its ability to handle unforeseen obstacles that can force the robot to change its current path.

The results show that the proposed algorithm is more efficient than the other methods tested in environments with and without dynamic obstacles. Also, in terms of resilience, the algorithm can handle unpredictable behaviors such as corrupted messages, loss of connection for an extended period, failures to complete tasks, and obstacles blocking the robot's path and forcing them to take a different trajectory. Finally, the algorithm is also flexible since it can be used for several different purposes and is robust to communications failures.

One limitation of the proposed algorithm is being ill-equipped to manage a substantial influx of task requests, given that solely a single task is auctioned and assigned at any given time.

As future work, efforts will be directed towards the refinement of the algorithm to address its limitations in managing high rates of task input. Additionally, plans are in place to deploy the algorithm in an authentic factory setting.

References

[1] D. Bertsekas. A distributed algorithm for the assignment problem. Massachusetts Institute of Technology (MIT), 1979.
[2] D. Bertsekas. The auction algorithm for assignment and other network flow problems. MIT, 1989.
[3] L. Liu and D. Shell. Optimal Market-based Multi-Robot Task Allocation via Strategic Pricing. Texas University, College Station, TX, USA, 2013.
[4] B. Gerkey and M. Mataric. Sold!: auction methods for multirobot coordination. IEEE Trans. on Robotics and Automation, 2002.
[5] A. Gómez. Comparison of multi-robot task allocation algorithms. Technical Report, Hochschule Bonn-Rhein-Sieg - Univ. of Applied Sciences, Dept. of Computer Science, 2019.
[6] H. Choi, L. Brunet, and J. How. Consensus-Based Decentralized Auctions for Robust Task Allocation. IEEE Trans. on Robotics (TRO), 2009.
[7] H. Choi, L. Brunet, and J. How. Consensus-Based Auction Approaches for Decentralized Task Assignment. MIT, Cambridge, 2008.
[8] L. Parker. L-ALLIANCE: Task-oriented multi-robot learning in behavior-based systems. Advanced Robotics, 1996.
[9] L. Parker. ALLIANCE: An architecture for fault tolerant multirobot cooperation. IEEE Trans. on Robotics and Automation, 1998.
[10] G. Das, T. McGinnity, S. Coleman, and L. Behera. A Distributed Task Allocation Algorithm for a Multi-Robot System in Healthcare Facilities. Journal of Intelligent and Robotic Systems, 2015.
[11] L. Parker. Autonomous Fault Tolerant Multi-Robot Cooperation Using Artificial Immune System. Proc. of the IEEE Int. Conf. on Automation and Logistics, 2008.
[12] S. Botelho and R. Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. Proc. of the IEEE Int. Conf. on Robotics & Automation, 1999.
[13] SS. Koenig, C. Tovey, M. Lagoudakis, V. Markakis, D. Kempe1 P. Keskinocak, A. Kleywegt, A. Meyerson and S. Jain. The Power of Sequential Single-Item Auctions for Agent Coordination. Proc. of the Conf. on Advancements of Artificial Intelligence, 2006.

Distributed 3D-Map Matching and Merging on Resource-Limited Platforms using Tomographic Features

Halil Utku Unlu[1], Anthony Tzes[2,3], Prashanth Krishnamurthy[1,3], and Farshad Khorrami[1,3]

***Abstract*— A fast, robust, resource-efficient, and distributed 3D map matching and merging algorithm utilizing extracted tomographic features is studied. Instead of depending on 3D features and descriptors, 2D features are extracted from 2D projections of horizontal sections of gravity-aligned local maps and matched with slices from the other map at different height differences, enabling the estimation of four degrees of freedom. The proposed algorithm is observed to provide order-of-magnitude improvements in memory and time efficiency over state-of-the-art feature extraction and registration pipelines, rendering it useful for near real-time map merging tasks in resource-limited platforms (e.g. UAVs).**

I. Introduction

Implementation of collaborative robotics in real systems [1] suffers from communication limitations, including network delays, available bandwidth, and intermittent connectivity. Furthermore, mobile robotic devices usually have limited computational capabilities, restricting the computational budget of the robotic platform significantly.

Simultaneous localization and mapping (SLAM) is a requirement for robots operating in an unknown environment. Single robot SLAM for both 2D- and 3D-motion is a mature field with many advanced platforms and frameworks for a variety of sensor setups [2]–[4]. Multi-robot collaborative SLAM (C-SLAM) remains an active research area [5]–[7]. C-SLAM algorithms focus on performing state estimations for multi-robot teams to improve resiliency and accuracy. Sharing map data between agents is not of primary concern.

In map matching and merging, agents share and update their understanding of the collective map upon establishing a communication link with one another. Multiple approaches have been proposed for 2D-map matching and merging scenarios [8], [9], but the solutions do not scale to 3D-map representations. The lack of a standard that establishes a common representation for 3D maps exacerbates the problem further, leaving the problem of multi-robot map knowledge sharing partially unaddressed.

Map matching is a subset of the point cloud registration problem in which the relative rotation and translation between two sets of point clouds with some overlap are sought. Many of the existing datasets and algorithms focus on scan matching [10]–[12]. However, map matching differs from scan matching due to the higher density of the input point clouds, providing a bigger computational challenge for existing feature extraction and matching algorithms, motivating this paper.

In this work, we address 3D map matching problem by providing a feature extraction framework that enables the use of 2D-image features in a pair of gravity-aligned 3D maps. The proposed framework is akin to the use of tomography-where a) the algorithm extracts a binary occupancy representation for horizontal cross-sections of the maps, b) extracts 2D features over these slices, and c) restricts matching space for the features across maps to its corresponding section only. The proposed approach is significantly efficient (in both time and memory) and accurate compared to state-of-the-art point cloud registration methods.

The contributions of this article include

- a simple and efficient approach to extracting 3D features via tomographic extraction of horizontal sections,
- study of viable uses of the aforementioned features in addressing large-scale 3D-map matching and merging scenarios,
- studies on real data to compare effectiveness against alternative methods.

The remainder of the paper is structured as follows: Section II provides the relevant work. Section III formulates the studied problem. Section IV defines the tomographic feature extraction, and Section V details two frameworks that utilize tomographic features for map matching. Studies to verify the algorithm's performance are provided in Section VI followed by concluding remarks.

II. Related Work

A. Point Feature Extraction & Description

Local 3D feature extraction algorithms require translation and orientation invariance for effectiveness and can be analyzed in two main categories: hand-crafted and learning-based. Many hand-crafted feature extractors define a local reference frame around individual points. Most notably in Fast Point Feature Histograms (FPFH) [13] the descriptors are comprised of the histograms of angular variations for a point of interest in its k-nearest neighborhood. Scale-persistent and statistically unique features are returned as features. A comprehensive review of hand-crafted features in [14] found FPFH to be effective in low-noise scan matching scenarios.

[1]H.U. Unlu, P. Krishnamurthy, and F. Khorrami are with Electrical & Computer Engineering Department, New York University, Brooklyn, NY 11201, USA. {utku, prashanth.krishnamurthy, khorrami}@nyu.edu

[2]A. Tzes is with the Electrical Engineering Program, New York University Abu Dhabi, Abu Dhabi, P.O. Box 129188, UAE. anthony.tzes@nyu.edu

[3]A. Tzes, P. Krishnamurthy and F. Khorrami are with the Center for Artificial Intelligence and Robotics, New York University Abu Dhabi, Abu Dhabi, P.O. Box 129188, UAE.

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

Learning-based feature extraction and description schemes utilize deep neural networks (DNNs) to find points of interest from an unordered set of points. Various DNN architectures have been proposed but fully-convolutional [15], [16] and transformer-based [17] DNNs are gaining popularity due to their efficiency and effectiveness. The main problem with 3D point feature extractors and descriptors is the speed. Algorithms either require GPUs to operate or take too long to compute for a near-real-time operation.

B. Point Cloud Registration

3D feature matches across different maps are converted into a pose transformation through registration algorithms. For noiseless data and perfect correspondences, an ideal form of the orthogonal Procrustes problem provides a closed-form solution. However, 3D correspondences are commonly observed to have as high as 95% outlier ratios [18], creating a need for robust solutions.

Learning-based registration frameworks cast the problem into differentiable sets of modules, treating the initial set of correspondences as putative and assigning weights to each correspondence. Deep Global Registration (DGR) [19] minimizes a robust energy metric with a convolutional network for confidence assignment, while PREDATOR [20] model enables the network to focus only on the overlapping regions via an attention-based mechanism.

Many other solutions for robust registration without learning-based methods exist. Sample consensus-based algorithms [21], [22] find the most consistent registration candidate by evaluating the solution generated from a minimal set of correspondences. TEASER [23] relies on semi-definite relaxation in its optimization, allowing it to handle extreme ($> 95\%$) outlier ratios with a certificate of optimality.

C. Map Matching & Merging

A review of map matching and merging algorithms for 2D maps can be found in [24]. Other notable algorithms utilize 'suppositional boxes' as features on 2D occupancy maps for better performance over feature-based methods [25] and centralized genetic algorithms operating on 2D raster representations of maps [26].

Some examples of 3D map merging algorithms are provided in [9]. A probabilistic map matching and merging scheme for 3D occupancy grids on heterogeneous sensor modalities is proposed in [27]. An algorithm on 3D maps with a pose-graph backend is proposed in [28], which allows for non-rigid deformations of the map structure and yields a tighter merging, albeit at a computational cost that prohibits online execution.

D. Collaborative SLAM (C-SLAM)

Many centralized [29], [30] and fully-distributed [5], [6] solutions have been proposed for the C-SLAM problem. Unlike map matching and merging, the primary focus for C-SLAM algorithms has been the pose optimization of the local trajectories. Even though the map matching and merging problem can be solved using C-SLAM solutions, a framework for sharing map data between agents is generally overlooked. The high bandwidth requirement for dense 3D map data remains a significant challenge.

III. Problem Formulation

This paper is tackling the problem of pairwise merging of 3D maps that can be canonically represented as a point cloud ${}^i\mathbf{m} = \{\dots, {}^i\mathbf{p}_j, \dots\}, |{}^i\mathbf{m}| = N_i$ expressed as a set of 3D coordinates ${}^i\mathbf{p}_j \in \mathbb{R}^3$ and generated by agent i.

The points in map i can be transformed into a common world frame w, $\mathbf{T}_i^w \in SE(3)$, via a rigid 3D 4×4 homogeneous transformation. The global world frame w can be arbitrarily defined by anchoring one agent's frame as the global origin. Therefore, the goal is to find the 3D rigid transformation between the global frame of reference and the agent i's local frame using the maps ${}^i\mathbf{m}, i \in \{c, d\}$. No prior knowledge of the absolute pose information for any agent is assumed, and the agents are not required to observe each other via a rendezvous (indirect map merging).

In this work, we assume that the local maps are gravity-aligned (i.e., the same z-axis). If the first agent's (# c) coordinate is selected as the world frame, the problem reduces to estimating the transformation $\mathbf{T}_d^c$ that is restricted to four degrees of freedom (DoF): three translations in x, y, and z and one rotation θ around z-axis. Notice that the given setup does not restrict the motion of the agents. The gravity vector can be reliably measured using an inertial measurement unit (IMU) while the agent is executing any motion in 6 DoF.

Optimization-based registration algorithms utilize putative correspondences to calculate the transformation between the representations of the maps, which can be all the points (dense), or a smaller subset of points of interest (sparse), matched across different point clouds via their descriptors. Part of this paper addresses the problem of efficient feature extraction using tomographic features.

Let the set of correspondences be defined as $\mathcal{C} = \{({}^c\mathbf{p}_m, {}^d\mathbf{p}_n) \;:\; m, n \in \mathbb{N}, m \leq N_c, n \leq N_d\}$.The registration task can be cast as an optimization as

$$\mathbf{T}_d^c = \underset{\mathbf{T} \in SE(3)}{\arg\min} \sum_{({}^c\mathbf{p}_m, {}^d\mathbf{p}_n) \in \mathcal{C}} \rho({}^c\mathbf{p}_m, \mathbf{T}\ {}^d\mathbf{p}_n) \qquad (1)$$

where $\rho(\mathbf{a}, \mathbf{b})$ is a non-negative metric to determine the error for a given correspondence. In the case of point-to-point ICP, the cost metric assumes the form of Euclidean distance.

Despite the non-convexity of Euclidean distance in the optimization cost, it is possible to find a closed-form solution, assuming all of the correspondences are correct. In practice, the correspondences should be treated as putative due to high outlier rates, and an inlier set $\mathcal{C}_{\text{in}} \subseteq \mathcal{C}$ needs to be selected from correspondences that agree with the unknown, correct transformation, leading to a modification of (1) as:

$$\mathbf{T}_d^c = \underset{\mathbf{T} \in SE(3)}{\arg\min} \sum_{({}^c\mathbf{p}_m, {}^d\mathbf{p}_n) \in \mathcal{C}_{\text{in}}} \rho({}^c\mathbf{p}_m, \mathbf{T}\ {}^d\mathbf{p}_n)\ . \qquad (2)$$

IV. Tomographic Feature Extraction

A tomographic section of the map, termed 'slice' throughout the paper, is defined as the 2D binary occupancy representation of a horizontal cross-section of a map $\mathbf{m}_i$, ${}^i\hat{\mathbf{s}}_h = \{{}^i\mathbf{p}_j \ : \ h - t < |{}^i\mathbf{p}_j|_z < h + t\}$ with h as the height at which the slice is extracted and t as the distance parameter that determines the thickness of the cross-section.

Points in ${}^i\hat{\mathbf{s}}_h$ are projected onto the xy-plane and discretized on a 2D grid to obtain a 2D binary occupancy image, ${}^i\mathbf{s}_h$. The extrema xy coordinates of points in ${}^i\hat{\mathbf{s}}_h$ and the grid size g dictate the width and height of the binary image. Intuitively, each pixel represents a real area of size $g \times g$.

In practice, the 3D point cloud is pre-processed with a voxel grid filter to reduce the number of points to a given resolution and to eliminate uneven point density. The leaf size of the voxel grid filter is a natural choice for the grid size g during the slicing of the entire map. Furthermore, the points are separated approximately by the voxel grid leaf size along the z-dimension. For this reason, we select the heights h at which the slices are extracted to be at least g apart, encompassing a height of g (i.e., $t = g/2$) in order to utilize all of the available information.

ORB features and descriptors [31] are extracted for the 2D binary images. There are many other potential feature extraction and description pipelines [32]. However, the processed binary images do not possess real photo-like intensity changes. ORB feature pipeline describes oriented FAST corners with BRIEF descriptor on orientation-aligned patches, providing an ideal option in this scenario. Its reliability and efficiency are validated in the literature against its alternatives [33] for other use cases.

The reason that these slices are extracted perpendicular to the z-axis is due to the ease of the observation of the gravity vector. Many mobile robots utilize an onboard IMU to estimate part of their state (e.g. attitude, velocity, acceleration). The gravity vector is common among all agents, regardless of their motion or mobility. As such, the algorithm is agnostic to agents' motion capabilities.

V. Registration with Tomographic Features

Extracting features on slices removes a dimension that prevents exactly identifying the correct match for a repetitive feature in the 2D projection of a 3D structure. Finding the closest match for one feature in the source map among all the features in all slices of a target map will yield erroneous correspondences. We analyze two different algorithms that address this issue: Consensus-based and Direct. A visual summary of the general framework is provided in Figure 1.

A. Consensus-based Registration

In this algorithm, we divide the computation of the 4 DoF into two distinct components: joint estimation of x, y, and θ, and the estimation of z.

1) Per-slice Estimation: For now, let us assume that the relative height (i.e. parameter z_d^c) is known. We will outline the exact method to estimate relative height in the next section. Given z_d^c, the problem reduces to finding a 2D rigid transformation, which has the form

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & x_d^c \\ \sin(\theta) & \cos(\theta) & y_d^c \end{bmatrix} \begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{3}$$

for corresponding points $\mathbf{p}_{\{c,d\}} = \begin{bmatrix} x_{\{c,d\}} & y_{\{c,d\}} \end{bmatrix}^\top$. Solution to the linear system below yields the parameters θ_d^c, x_d^c, and y_d^c:

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ x_d & -y_d & 1 & 0 \\ y_d & x_d & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ x_d^c \\ y_d^c \end{bmatrix} = \begin{bmatrix} \vdots \\ x_c \\ y_c \\ \vdots \end{bmatrix} \tag{4}$$

where $\alpha = s\cos(\theta_d^c)$ and $\beta = s\sin(\theta_d^c)$ with s as the scale parameter. To recover the angle θ_d^c, we simply use $\text{atan2}(\beta, \alpha)$. Since the maps are known to have the same scale, no estimation of s is needed. For simplicity, we use RANSAC to obtain a robust solution, followed by Levenberg-Marquardt refinement steps over the inlier set.

In this manner, each individual slice provides a 2D rigid transformation estimation $\mathbf{T}(x_d^c, y_d^c, z_d^c, \theta_d^c) \triangleq \mathbf{T}_d^c$, at a known height of z_d^c. The collection of the 2D rigid estimations at a particular z_d^c forms the set $\mathcal{T}(z_d^c)$.

However, we cannot expect all slices to contain sufficient occupancy information to provide a meaningful estimate, resulting in erroneous measurements that need to be eliminated. To that end, we find the consensus between different 2D rigid estimations by finding the largest set of hypotheses with the shortest distance to an anchor hypothesis:

$$\mathcal{T}(z_d^c) = \arg\max \left| \ [\mathbf{T}_d^c]_i : \mathbf{d}([\mathbf{T}_d^c]_i - [\mathbf{T}_d^c]_j) \prec \mathbf{t}, [\mathbf{T}_d^c]_{\{i,j\}} \in \mathcal{T}(z_d^c) \ \right| \tag{5}$$

where the vector-valued function $\mathbf{d}$ encodes the Euclidean distance between the estimates of x_d^c, y_d^c and the angular distance between the angles θ_d^c, and $\mathbf{t}$ is a threshold to specify maximum allowed deviations. We then take the parameter-wise average of the hypotheses in the inlier set $\mathcal{T}(z_d^c)$ to compute the resultant pose, $\bar{\mathbf{T}}_d^c$.

2) Relative Height Estimation: Up until now, we assumed that the height z_d^c is known. However, matching z-axis height between different agents is a strong assumption that restricts the applicability of the proposed system significantly.

We estimate z_d^c based on consensus again. Due to the grid-based nature of the 3D map, there is a finite number of relative height differences we can establish between different maps, all of which are g units apart. We calculate the "cross-correlation" of the map slices from different maps, estimating the rigid 2D transformation as outlined above and using the number of inliers as the correlation value. The height difference z_d^c with the largest inlier set cardinality is selected as the height estimate:

$$z_d^c = \arg\max_{z_d^c} \left| \mathcal{T}(z_d^c) \right| \tag{6}$$

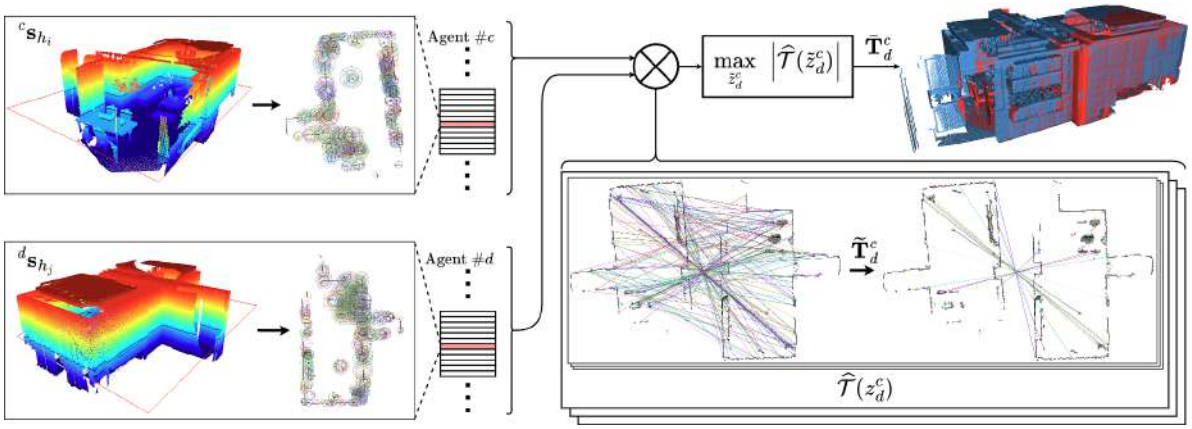


Fig. 1: Visual summary of the proposed distributed 3D map matching framework. Each agent $\{c, d\}$ is responsible for extracting slices ${}^{\{c,d\}}\mathbf{s}_{h_i}$ at a predefined grid size. One of the agents cross-correlates the slices by estimating a 3DoF rigid transformation, $\mathbf{T}_d^c$, between slices. Consensus of different height hypotheses, z_d^c, yields the 4th DoF.

Our empirical tests indicate that the number of inlier correspondences is maximized for all slices when they are matched with the correct slice from the other map, for indoor/outdoor environments with distinctive geometric features. The cross-correlation scheme is expected to fail for environments that are corridor-like with no distinct 3D objects that change the uniformity between adjacent slices. However, such idealized scenarios are not usual to encounter in practice. As such, we opted for the cross-correlation scheme for the relative height estimate.

Even though the above algorithm is expensive due to the slice correlations, each individual step (feature calculation and 3DoF estimation for a pair of slices) is independent of each other, enabling parallelization opportunities that can provide further speed-ups with hardware acceleration. However, we will demonstrate that it is not fully necessary for intermittent map matching and merging.

B. Direct Registration

Instead of breaking down the estimation into two separate steps, we can estimate the full 4 DoF transformation if we can convert the correspondences to 3D-3D. As discussed before, we lose some information during the slicing operation, yielding many incorrect feature matches. However, the slicing operation is structured and it is not necessary to find matches for one feature across all maps. For a given height z_d^c, we expect the matching features to be contained within the slice pair from the opposite map that is z_d^c apart in z-axis.

To that end, the relative height z_d^c can be used to augment 2D correspondences across different slices into their 3D counterparts. We aggregate features from different slices to yield two 3D point clouds with putative matches, which we register using the TEASER algorithm to obtain the 4 DoF rigid transformation estimate. We refer to this method as "Tomographic TEASER++" throughout the paper.

Similar to the consensus-based algorithm, the relative height z_d^c is not known. We estimate it by performing registration at every unique z_d^c hypothesis that is valid and select the value that provides the largest inlier set. Again, each registration problem is independent of each other, providing opportunities for parallelization.

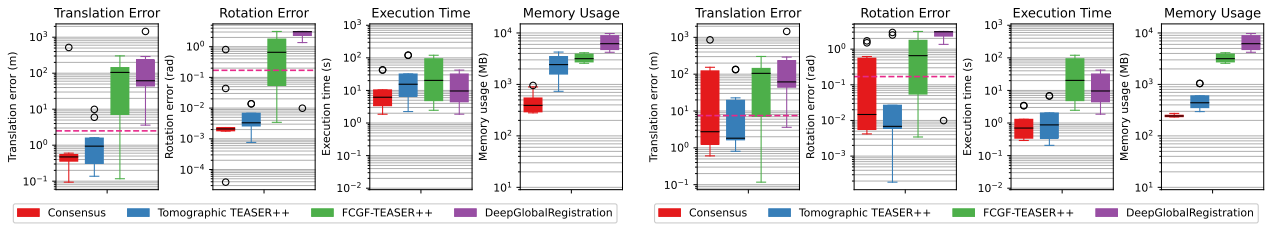
VI. Performance Evaluation

We evaluated the performance of the proposed algorithms in real-life datasets, in terms of their translational accuracy, rotational accuracy, memory footprint, and execution time. We provide comparisons against robust registration pipelines using learning-based features (FCGF [15] TEASER++) and completely learning-based registration frameworks (DeepGlobalRegistration [19]). We note that newer state-of-the-art learning-based registration pipelines have been proposed (e.g. PREDATOR [20], GeoTransformer [17]). However, due to the scale of the map matching problem, none of the pipelines with existing implementation could be made to fit into the GPU memory of a laptop-grade (or sometimes even a workstation-grade) GPU.

Unless specified otherwise, the tests are performed on an Intel Phantom Canyon NUC11PHKi7C (Intel i7-1165G74 CPU, NVIDIA RTX 2060 6GB GPU, 64 GB RAM).

To evaluate the performance on a real-life dataset, KITTI [34] odometry sequences are used. There are 11 sequences for which the GPS ground truth position information is provided. However, only 5 of the 11 sequences (00, 02, 05, 06, 07) revisit the previously explored locations, providing a map matching scenario when the sequence is divided into two. In total, there are 10 different instances of map merging tasks that are generated with the KITTI dataset, with an average overlap of 43.90% (minimum 11.17%, maximum 83.53%).

KITTI sequences provide significantly more challenges for learning-based systems due to the sheer size of the maps. The original grid size of 0.3 m as used by the original authors of the works cannot fit into the memory of a desktop-grade GPU. Therefore for the KITTI study, learning-based algorithms are run using the KITTI weights as trained by the original authors on a 0.5 m grid size, but downsizing the maps to a grid size of 1.5 m, and setting the grid parameters



(a) (b)

Fig. 2: Aggregated errors, execution times, and memory usage of the tested algorithms on select KITTI sequences. Inputs maps have been pre-filtered with a voxel grid filtering with a grid size of 0.5 m. In Figure 2a, tomographic algorithms use a grid size of 0.5 m, while learning-based algorithms use a grid size of 1.5 m. All algorithms use 1.5 m grid size in 2b. Error thresholds ($5 \times$ grid size for translation, 0.17 rad ($\approx 9.7°$) for rotation) are marked in dashed magenta line.

accordingly. Even at 1.5 m, the learning-based algorithms cannot be run on the NUC Enthusiast. As such, the results reported for learning-based algorithms on KITTI data are from execution on a workstation with 12 GB GPU memory. The proposed algorithms still use the device specified before.

Tomography-based methods can handle as low as 0.5 m grid sizes without long execution times. Results on KITTI sequences for a grid size of 0.5 m for tomographic methods, and 1.5 m for learning-based methods are provided in Figure 2b. To compare the performance under the same grid size, Figure 2a provides the results when the grid size is set to 1.5 m for all algorithms. Note that the learning-based algorithms are run on a desktop machine. Also, note that no parameter tuning is performed for the proposed tomographic algorithms in the KITTI studies, except for the grid size adjustment.

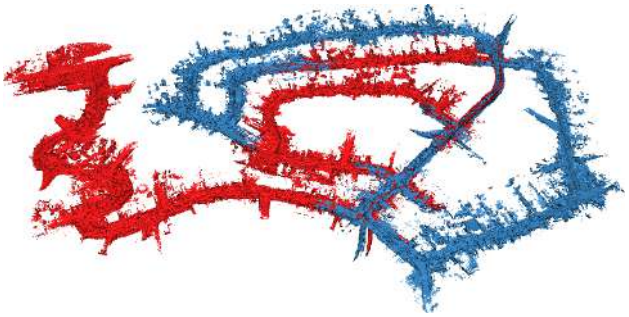
At the grid size of 0.5 m, the Consensus algorithm provides the lowest translation and rotation errors with the shortest execution time and smallest memory footprint, rivaled only by Tomographic TEASER++. Both learning-based methods manage to accurately register only one instance out of 10 possible pairings. Even though the execution time of DeepGlobalRegistration is comparable to tomographic methods, memory usage is the largest out of all tested systems.

Increasing the grid size to 1.5 m degrades the performance of tomographic methods, but they still perform better than learning-based methods. Execution times of Consensus and Tomographic TEASER++ methods are roughly equal at a grid size of 1.5 m, but Consensus has the lowest memory footprint of approximately 250 MB.

An example merging on maps generated from the KITTI 02 sequence using Consensus is provided in Figure 3. This pair provides the smallest overlap of all sequences in KITTI. Each individual map spans a height of approximately 50 m, which is color-coded in the figures. Resultant alignment correctly estimates the 30 m displacement in the z-axis required to find the correct alignment.

(a) Map from the first half of KITTI sequence 02.

(b) Map from the second half of KITTI sequence 02.



(c) Matched & merged map with Consensus.

Fig. 3: A sample matching & merging operation using Consensus algorithm. Figures 3a and 3b are color coded based on their z-coordinate.

VII. Conclusions

We proposed a computationally lightweight approach to generate effective features for gravity-aligned maps and

demonstrated the performance against possible alternative algorithms. A consensus-based and a more holistic registration paradigm are demonstrated to be both more accurate and efficient compared to 3D feature generation and matching algorithms. State-of-the-art learning-based approaches are not suitable for the map matching task due to a lack of standard training data and the memory requirements that surpass that of scan matching. Furthermore, the proposed tomographic approach to extracting features is observed to be resilient to noise and does not require any additional parameter tuning for maps of different scales. The findings are corroborated on real datasets that map volumes of different scales, underscoring the algorithmic efficiency and accuracy.

ACKNOWLEDGMENT

This work was partially supported by the NYUAD Center for Artificial Intelligence and Robotics (CAIR), funded by Tamkeen under the NYUAD Research Institute Award CG010.

REFERENCES

[1] S. Papatheodorou, A. Tzes, K. Giannousakis, and Y. Stergiopoulos, "Distributed area coverage control with imprecise robot localization: Simulation and experimental studies," *International Journal of Advanced Robotic Systems*, vol. 15, no. 5, pp. 1–15, 2018.

[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[3] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "maplab: An open framework for research in visual-inertial mapping and localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1418–1425, 2018.

[4] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[5] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, "DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1656–1663, 2020.

[6] Y. Tian, Y. Chang, F. H. Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-multi: Robust, distributed, dense metric-semantic slam for multi-robot systems," *IEEE Transactions on Robotics*, vol. 38, no. 4, 2022.

[7] P.-Y. Lajoie, B. Ramtoula, F. Wu, and G. Beltrame, "Towards collaborative simultaneous localization and mapping: a survey of the current research landscape," *Field Robotics*, vol. 2, no. 1, pp. 971–1000, 2022.

[8] A. Birk and S. Carpin, "Merging occupancy grid maps from multiple robots," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1384–1397, 2006.

[9] I. Andersone, "Heterogeneous map merging: State of the art," *Robotics*, vol. 8, no. 3, p. 74, 2019.

[10] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. A. Funkhouser, "3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 199–208, 2016.

[11] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger, "Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset," in *British Machine Vision Conference (BMVC)*, 2018.

[12] F. Pomerleau, M. Liu, F. Colas, and R. Y. Siegwart, "Challenging data sets for point cloud registration algorithms," *The International Journal of Robotics Research*, vol. 31, pp. 1705 – 1711, 2012.

[13] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3212–3217.

[14] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3D local feature descriptors," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 66–89, 2016.

[15] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8958–8966.

[16] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3D point cloud matching with smoothed densities," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5545–5554.

[17] Z. Qin, H. Yu, C. Wang, Y. Guo, Y. Peng, and K. Xu, "Geometric transformer for fast and robust point cloud registration," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11 133–11 142, 2022.

[18] A. P. Bustos and T.-J. Chin, "Guaranteed outlier removal for point cloud registration with correspondences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2868–2882, 2017.

[19] C. Choy, W. Dong, and V. Koltun, "Deep global registration," in *CVPR*, 2020.

[20] S. Huang, Z. Gojcic, M. Usvyatsov, A. Wieser, and K. Schindler, "Predator: Registration of 3d point clouds with low overlap," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[21] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[22] D. Baráth, J. Noskova, and J. Matas, "Marginalizing sample consensus," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 8420–8432, 2021.

[23] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and certifiable point cloud registration," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 314–333, 2020.

[24] S. Yu, C. Fu, A. K. Gostar, and M. Hu, "A review on map-merging methods for typical map types in multiple-ground-robot SLAM solutions," *Sensors*, vol. 20, no. 23, p. 6988, 2020.

[25] B. Chen, S. Li, H. Zhao, and L. Liu, "Map merging with suppositional box for multi-robot indoor mapping," *Electronics*, 2021.

[26] Q. Sun, T. Liao, H. Du, Y. Zhao, and C.-C. Chen, "A Method of Merging Maps for MUAVs Based on an Improved Genetic Algorithm," *Sensors (Basel, Switzerland)*, vol. 23, 2023.

[27] Y. Yue, P. N. Senarathne, C. Yang, J. Zhang, M. Wen, and D. Wang, "Hierarchical probabilistic fusion framework for matching and merging of 3-D occupancy maps," *IEEE Sensors Journal*, vol. 18, no. 21, pp. 8933–8949, 2018.

[28] T. M. Bonanni, B. Della Corte, and G. Grisetti, "3-D map merging on pose graphs," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1031–1038, 2017.

[29] P. Schmuck and M. Chli, "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *Journal of Field Robotics*, vol. 36, no. 4, pp. 763–781, 2019.

[30] Y. Chang, K. Ebadi, C. Denniston, M. F. Ginting, A. Rosinol, A. Reinke, M. Palieri, J. Shi, A. Chatterjee, B. Morrell, A. akbar Agha-mohammadi, and L. Carlone, "Lamp 2.0: A robust multi-robot slam system for operation in challenging large-scale underground environments," *IEEE Robotics and Automation Letters*, vol. 7, pp. 9175–9182, 2022.

[31] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2564–2571.

[32] J. Ma, X. Jiang, A. Fan, J. Jiang, and J. Yan, "Image matching from handcrafted to deep features: A survey," *International Journal of Computer Vision*, vol. 129, no. 1, pp. 23–79, 2021.

[33] S. A. K. Tareen and Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," in *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE, 2018, pp. 1–10.

[34] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

A Temporal Perspective n-Point Problem with Model Uncertainties for Cooperative Pose Estimation in a Heterogeneous Robot Team

Florian Steidle[1], Simon Boche[2], Wolfgang Stürzl[1], and Rudolph Triebel[1]

***Abstract*— Many solutions exist for estimating the pose of an object with respect to a camera, where perfect knowledge of the object is assumed. In this work we lift the assumption of a perfectly known model and introduce uncertainties for the 3d points, which are retrieved from a dynamically created model. The positions of model points can either be uncorrelated or correlated. The latter is typically the case for mobile robots navigating based on results of visual-inertial pose estimation in unknown and GNSS-denied environments. In our approach, a selection of poses estimated by one robot is used as a dynamical 3d model and combined with 2d points from tracking the robot over time with the camera of another robot. In addition, selection criteria for adding and deleting 3d model points in an optimal way are proposed. Weighted residuals in the tangent space are used in a generalized least-squares problem to calculate the transformation between the tracking camera and an object. Measurement errors are projected into tangential planes of the unit sphere.**

The proposed method allows to estimate the relative pose of members of a robotic team with high accuracy. The benefits of our approach are shown in simulation and also during real-world experiments using visual odometry measurements from a multicopter that is tracked by the camera of a rover.

I. INTRODUCTION

Determination of the pose of an object with respect to a camera is a broad field and many different applications and approaches exist to solve the problem. If distinct feature points on the object can be detected by a calibrated camera, the task is called Perspective n-Point Problem (PnP). It needs a model of the object and a calibrated camera. Usually, the model is assumed to be perfectly known. Therefore, no uncertainty in the location of 3d model points is considered and also most approaches do not consider uncertainties of 2d image points. Instead of using 3d points from a model that is known in advance, e.g. from CAD, also the model can be created at runtime. One possibility is the integration of consecutive Visual Odometry (VO) measurements. Thereby, the model is spanned over time and the transformation cannot be determined at a single time instant, but needs several observations over time (Fig. 1). As in usual PnP approaches, observations are the projections of 3d points into the camera used for tracking. But instead of using a predefined 3d model, we create the 3d model at runtime by integrating VO observations from the tracked object. Hence our approach combines the classical PnP with a dynamical creation of the model and addresses the main challenges in this scenario.

[1]Authors are with DLR German Aerospace Center, Institute of Robotics and Mechatronics florian.steidle@dlr.de

[2]Author is with Smart Robotics Lab, Technical University of Munich, School of Computation, Information and Technology

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

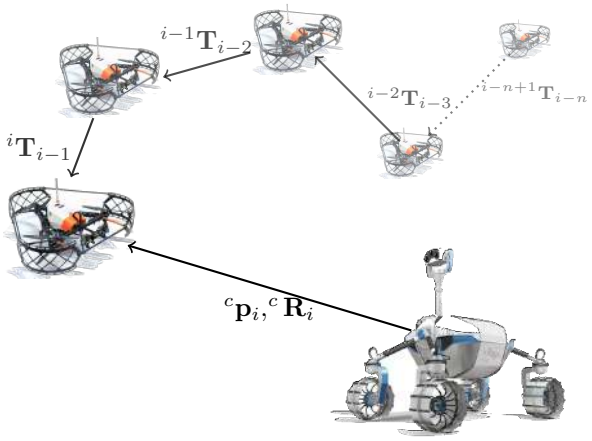


Fig. 1: The pose $^{c}\mathbf{p}_i, ^{c}\mathbf{R}_i$ of the multicopter ARDEA (ARDEA) at time i is estimated with respect to the Light Weight Rover Unit (LRU). Estimation is based on 2d measurements of ARDEA in the camera of LRU and integration of a number of past visual odometry measurements $^{i}\mathbf{T}_{i-1}, ..., ^{i-n+1}\mathbf{T}_{i-n}$.

An envisioned application of the proposed method, e.g. in a planetary exploration setting, is the accurate pose estimate of a scouting drone with respect to another team member in order to make optimal use of the information provided by the drone. This, for instance, will allow a rover to directly reach a point of interest detected by the drone or to evaluate the terrain for path planning with respect to the rover's reference frame based on images sent by the drone and could be a relevant component in space-analogue demonstrations like ROBEX [13] or ARCHES [9] and future planetary missions.

There exist many approaches to solve the classical PnP problem. In [4], the problem is reduced to estimating 4 virtual control points. The result is obtained by a weighted sum of these points and called Efficient PnP (EPnP). An other approach is introduced in [1], which takes into account the uncertainty of 2d image features and therefore lift the assumption that the location of all 2d features is known with the same accuracy. More recently, in [11] a statistically optimal solution taking feature point uncertainty into consideration was introduced. The approach is called Maximum Likelihood Perspective n-Point Problem (MLPnP). It projects the 3d model points and the corresponding 2d camera observations to a tangent plane on the unit sphere. Also uncertainties associated to 2d camera observations are projected to the tangent space and a weighted non-linear optimization is performed to get the result.

Besides uncertainty of the 2d features, [12] introduced uncertainty of the sparse feature points of the underlying 3d model. The approach, which uses point and line features, is based on EPnP [4] and Direct Least Squares Method (DLS) [3]. It assumes knowledge of the average scene depth or needs a rough initial guess of the transformation. With the average scene depth the 3d point depths are approximated and based on that an isotropic approximation of the 3d point covariance is calculated. A rough initial guess of the transformation is used to approximate the covariances of the 3d points. The second work, that incorporates uncertainty of 3d feature points is Extended Kalman Filter for Camera Pose Estimation in a Sequence of Images (EKFPnP) [6]. During the update step of an Extended Kalman Filter (EKF) an a priori estimate is received based on the camera motion model. In the correction step, the reprojection error is minimized. In contrast to [12] and [6], our approach does not use a model known in advance and allows correlation of model points.

Our main contributions are:

- instead of using a beforehand known 3d model, the model is spawned dynamically, online and over time
- 3d uncertainties of and correlation between model points are considered in addition to 2d measurement uncertainties
- different criteria for selecting new points to be added to the model are proposed and evaluated
- different criteria for deciding which point to delete are proposed and evaluated. One criterion is directly based on propagated uncertainty.

We show the benefit of our approach in simulation and real-world experiments.

II. SYSTEM MODELING AND POSE ESTIMATION

The overall system consists of two robots, see Fig. 1. One robot, LRU [9] is equipped with several calibrated cameras, which can be used for tracking of objects. The second robot is ARDEA [5], [7], which uses VO to estimate its egomotion. The primary goal is to calculate the position ${}^{c}\mathbf{p}_i$ and orientation ${}^{c}\mathbf{R}_i$ of ARDEA with respect to LRU.

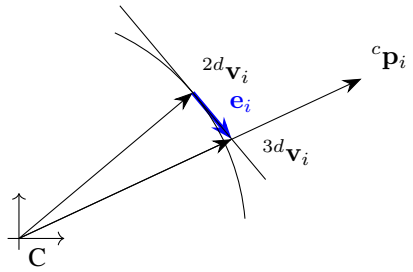


Fig. 2: The error $\mathbf{e}_i$ is the difference between ${}^{2d}\mathbf{v}_i$ and ${}^{3d}\mathbf{v}_i$, which is the projection of ${}^{c}\mathbf{p}_i$ onto the tangent plane defined by ${}^{2d}\mathbf{v}_i$. ${}^{2d}\mathbf{v}_i$ is the unprojected and normalized 2d observation in the camera frame and ${}^{c}\mathbf{p}_i$ is the corresponding 3d model point after transformation into the camera frame.

At each time i the calibrated camera provides 2d measurements $\mathbf{x}'_i \in \mathbb{R}^2$ with associated covariance matrices $\mathbf{\Sigma}_{\mathbf{x}'}$ representing measurement uncertainties. For better readability the subscript i is omitted in the remainder of this document, whenever possible and if it is clear that the variables refer to a specific point in time. The unprojection of a point $\mathbf{x}' \in \mathbb{R}^2$ in the camera frame to a direction vector $\mathbf{x} \in \mathbb{R}^3$ follows the equation

$$\mathbf{x} = \pi^{-1}(\mathbf{x}') \qquad \mathbf{\Sigma}_{\mathbf{x}} = \mathbf{J}_{\pi^{-1}} \mathbf{\Sigma}_{\mathbf{x}'} \mathbf{J}_{\pi^{-1}}^T \tag{1}$$

with $\pi^{-1} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ being the unprojection function of the camera. In the perspective case with focal length f and principal point (c_x, c_y), we simply have $\pi^{-1}(\mathbf{x}') = (x' - c_x, y' - c_y, f)^T$. Following the ideas of [11], the subsequent spherical normalization

$${}^{2d}\mathbf{v} = \frac{\mathbf{x}}{\|\mathbf{x}\|} \qquad {}^{2d}\mathbf{\Sigma}_{\mathbf{v}} = \mathbf{J}_{{}^{2d}\mathbf{v}} \mathbf{\Sigma}_{\mathbf{x}} \mathbf{J}_{{}^{2d}\mathbf{v}}^T$$

leads to the final observations on the unit sphere and

$$\mathbf{J}_{{}^{2d}\mathbf{v}} = \frac{1}{\|\mathbf{x}\|} \left(\mathbf{I}_{3\times 3} - {}^{2d}\mathbf{v} {}^{2d}\mathbf{v}^T \right).$$

The superscript '$2d$' indicates that the unit vector $\mathbf{v}$ corresponds to a 2d image point. The formulation on the unit sphere allows to use non-standard cameras, e.g., a fisheye camera with field of view beyond 180°. According to [2] and [11], a homogeneous vector $\mathbf{v}$ can be projected to its reduced equivalent ${}^{2d}\mathbf{v}_r \in \mathbb{R}^2$ with

$${}^{2d}\mathbf{v}_r = \mathbf{J}_{\mathbf{v}_r}^T {}^{2d}\mathbf{v} = 0, \tag{2}$$

where the column vectors of $\mathbf{J}_{\mathbf{v}_r} \in \mathbb{R}^{3\times 2}$ are a basis for the the nullspace of ${}^{2d}\mathbf{v}$ and can be used to obtain residuals in the tangent plane defined by ${}^{2d}\mathbf{v}$, see Fig. 2. For a vector in the tangent plane the residual is $\mathbf{e} = \mathbf{J}_{\mathbf{v}_r}^T({}^{3d}\mathbf{v} - {}^{2d}\mathbf{v}) = \mathbf{J}_{\mathbf{v}_r}^T {}^{3d}\mathbf{v}$. In addition, by projecting ${}^{2d}\mathbf{v}$ to its reduced subspace, the associated covariance matrix $\mathbf{\Sigma}_{{}^{2d}\mathbf{v}_r}$ is no longer singular.

The second source of measurements originates from the tracked system and consists of a sequence of pose changes

$${}^{\mathrm{i-j+1}}\boldsymbol{T}_{\mathrm{i-j}} = \begin{bmatrix} {}^{i-j+1}\mathbf{R}_{i-j} & {}^{i-j+1}\mathbf{t}_{i-j} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$$

with the corresponding initial $t^s_{i,j}$ and final timestamps $t^e_{i,j}$. Such pose changes could be estimated by a VO or Visual-Inertial Navigation System (VINS). If no camera measurement $\mathbf{x}'$ corresponding to the end time $t^e_{i,j}$ of a VO measurement is available, several VO measurements are integrated until a camera observation with matching timestamp is available. With n previous readings from the VO ${}^{\mathrm{i}}\boldsymbol{T}_{\mathrm{i-1}}, {}^{\mathrm{i-1}}\boldsymbol{T}_{\mathrm{i-2}}, ..., {}^{\mathrm{i-n+1}}\boldsymbol{T}_{\mathrm{i-n}}$ and the current transformation ${}^{c}\boldsymbol{T}_{\mathrm{i}} = ({}^{c}\mathbf{R}_i, {}^{c}\mathbf{t}_i)$ between ARDEA and LRU, the previous positions of ARDEA can be calculated according to the following scheme

$$\begin{aligned} {}^{c}\mathbf{p}_i &= {}^{c}\mathbf{t}_i \\ {}^{c}\mathbf{p}_{i-1} &= {}^{c}\mathbf{t}_i + {}^{c}\mathbf{R}_i {}^{i}\mathbf{t}_{i-1} \\ {}^{c}\mathbf{p}_{i-2} &= {}^{c}\mathbf{t}_i + {}^{c}\mathbf{R}_i {}^{i}\mathbf{t}_{i-1} + {}^{c}\mathbf{R}_i {}^{i}\mathbf{R}_{i-1} {}^{i-1}\mathbf{t}_{i-2} \\ &\vdots \\ {}^{c}\mathbf{p}_{i-n} &= {}^{c}\mathbf{t}_i + \dots \end{aligned} \tag{3}$$

By introducing the term

$$ {}^{i}\bar{\mathbf{p}}_j = \sum_{m=1}^{j} \Big(\prod_{n=1}^{m-1} ({}^{i-m+n+1}\mathbf{R}_{i-m+n})^{i-m+1}\mathbf{t}_{i-m} \Big) $$

the pose of ARDEA at a former time t_{i-j} can be expressed with

$$ {}^{c}\mathbf{p}_{i-j} = {}^{c}\mathbf{R}_i \, {}^{i}\bar{\mathbf{p}}_j + {}^{c}\mathbf{t}_i \tag{4} $$

In case of 3d measurements, the points ${}^{c}\mathbf{p}_{i-j}$ with $j \in [0, 1, ..., n]$ have to be projected to the tangential plane defined by ${}^{2d}\mathbf{v}_{i-j}$. By calculating

$$ {}^{3d}\mathbf{v}_{i-j} = \frac{{}^{c}\mathbf{p}_{i-j}}{{}^{c}\mathbf{p}_{i-j}^{T} \, {}^{2d}\mathbf{v}_{i-j}} \tag{5} $$

the observations ${}^{2d}\mathbf{v}_{i-j}$ and ${}^{c}\mathbf{p}_{i-j}$ are projected onto the tangential plane defined by ${}^{2d}\mathbf{v}_{i-j}$.

Also, the covariance information has to be propagated to the tangential plane

$$ \mathbf{\Sigma_p} = \mathbf{B}\mathbf{\Sigma_u}\mathbf{B}^T. \tag{6} $$

Due to the structure of the underlying problem, $\mathbf{B} \in \mathbb{R}^{3(n+1)\times 6n}$ is a lower triangular matrix with $\mathbf{0}_{3\times 3}$ blocks on the diagonal. See the appendix for further details on its derivation. $\mathbf{\Sigma_u} \in \mathbb{R}^{6n\times 6n}$ is a block-diagonal matrix with elements $\mathbf{\Sigma}_{\mathrm{i-j+1}\boldsymbol{T}_{\mathrm{i-j}}}$ representing the uncertainty of the VO observations ${}^{\mathrm{i-j+1}}\boldsymbol{T}_{\mathrm{i-j}}$. The covariance of the points ${}^{c}\mathbf{p}_i$ to the tangential plane defined by ${}^{2d}\mathbf{v}$ is calculated by

$$ {}^{3d}\mathbf{\Sigma_v} = \mathbf{J_v}\mathbf{\Sigma_p}\mathbf{J_v}^T $$

with the block-diagonal matrix $\mathbf{J_v} \in \mathbb{R}^{3(n+1)\times 6n}$. As can be derived from (5), each block $\mathbf{J}_{\mathbf{v},kk} \in \mathbb{R}^{3\times 3}$ is given by

$$ \mathbf{J}_{\mathbf{v},kk} = \frac{1}{(\mathbf{p}^T\mathbf{v})^2} \left((\mathbf{p}^T\mathbf{v})\, \mathbf{I}_{3\times 3} - \mathbf{v}\mathbf{p}^T \right) $$

with the abbreviations $\mathbf{p} = {}^{c}\mathbf{p}_i$ and $\mathbf{v} = {}^{2d}\mathbf{v}$ introduced for readability.

The final covariance matrix

$$ \mathbf{\Sigma_v} = {}^{3d}\mathbf{\Sigma_v} + {}^{2d}\mathbf{\Sigma_v} $$

is used in the optimization of the transformation from Eq. (7) after projecting it to its reduced counterpart $\mathbf{\Sigma}_{\mathbf{v}_r}$ using $\mathbf{J}_{\mathbf{v}_r}$ from Eq. (2).

The main goal of the algorithm is to estimate the pose of the tracked object/robot with respect to the camera. This can be formulated as a nonlinear minimization problem. The transformation ${}^{\mathrm{c}}\boldsymbol{T}_{\mathrm{i}}$ consists of a translation ${}^{c}\mathbf{t}_i \in \mathbb{R}^3$ and a rotation part ${}^{c}\mathbf{R}_i$. During optimization a minimal representation of rotation is used and transformed to ${}^{c}\mathbf{R}_i$ using Rodrigues' Formula [8]. By stacking the reduced observations from Eq. (2) for different times, a nonlinear optimization problem in the form

$$ F(\mathbf{u}) = \mathbf{\Pi}_{\mathrm{null}}(\mathbf{p}, \mathbf{u})^T \mathbf{\Sigma}_{\mathbf{v}_r}^{-1} \mathbf{\Pi}_{\mathrm{null}}(\mathbf{p}, \mathbf{u}) \tag{7} $$

can be formulated and solved by e.g. a trust-region minimization. The vector $\mathbf{u} \in \mathbb{R}^{6\times 1}$ contains the six parameters defining ${}^{\mathrm{c}}\boldsymbol{T}_{\mathrm{i}}$ and $\mathbf{\Pi}_{\mathrm{null}}$ consists of the stacked residuals $\mathbf{J}_{\mathbf{v}_r}^T \, {}^{3d}\mathbf{v}$. Each camera measurement $\mathbf{x}'$ from Eq. (1) defines the projection matrices $\mathbf{J}_{\mathbf{v}_r}$ and each integrated VO measurement from Eq. (4) defines the reduced observations ${}^{3d}\mathbf{v}$.

To start the tracking process, an inital solution has to be calculated. Therefore, the non-linear system from Eq. (7) is reformulated as a linear system, following the approach from [11]. Thereby, the state of the linear system consists of three elements representing the translation ${}^{c}\mathbf{t}_i$ and nine elements representing the rotation matrix ${}^{c}\mathbf{R}_i$.

When the algorithm is executed, new measurements arrive continuously and it is assumed that the camera and VO streams are synchronized. Whenever a measurement from VO arrives, it is looked for a camera observation with equal timestamp. If no corresponding camera observation is available, the new VO measurement is appended to the measurement before. If a corresponding camera measurement is available, the decision has to be made, if the transformation ${}^{\mathrm{c}}\boldsymbol{T}_{\mathrm{i}}$ should be optimized with the new information or, if the benefit of executing the optimization is marginal in comparison to appending the latest VO measurement to the estimated transformation ${}^{\mathrm{c}}\boldsymbol{T}_{\mathrm{i}}$ in the time step before. If measurements are added each time the optimization is executed, the buffer containing all measurements grows unbounded. To keep processing time constant, each time a new measurement is added to the buffer another measurement has to be removed. Therefore, two aspects are remaining. The first one concerns the decision, when to add a new measurement to buffer and use it to execute the optimization. The second aspect relates to the decision, which measurement to remove from the buffer.

In our approach, the decision which measurement to replace is based on its influence on pose estimation, which we determine by means of the tangent plane leverage matrix $\mathbf{H} \in \mathbb{R}^{2n\times 2n}$ [10]. It is defined according to

$$ \mathbf{H} = \mathbf{J}_{\mathrm{opt}} (\mathbf{J}_{\mathrm{opt}}^T \mathbf{J}_{\mathrm{opt}})^{-1} \mathbf{J}_{\mathrm{opt}}^T $$

where the diagonal elements $h_{k,k}$ represent the sensitivity of a measurement with respect to the result and $\mathbf{J}_{\mathrm{opt}} \in \mathbb{R}^{2n\times 6}$ is the Jacobian of the weighted residual $\mathbf{\Sigma}_{\mathbf{v}_r}^{-\frac{1}{2}} \mathbf{\Pi}_{\mathrm{null}}$ at the solution. In [10], the largest values $h_{k,k}$ are used to determine leverage points. These points are assumed to have a high influence on the result. On the other hand, a value $h_{k,k}$ close to zero, indicates that the measurement has very little influence on the result. Therefore, each time the optimization is run, the diagonal elements $h_{k,k}$ are stored. Whenever a new measurement should be added to the optimization, the measurement corresponding to the smallest diagonal element is deleted and replaced with the new measurement.

The second aspect concerns the addition of a new observation. We employed three different criteria:

- ARDEA traveled more than a defined distance ("position-based")
- the last addition of an observation is too far back in time ("time-based")
- the covariance of the accumulated delta poses is above a threshold ("uncertainty-based")

All three criteria specify a rule for the addition of new measurements.

In the course of this paper, the camera used for tracking is assumed to be static. But it would be straightforward to extend the approach to a moving camera, e. g. by attaching it to a pan-tilt unit. Additionally, both system are assumed to be temporally aligned. That means, measurements from both systems refer to a common time basis and therefore, the correspondence between 3d model points and 2d camera observations is known. In the special case of uncorrelated model points, the covariance ${}^{3d}\mathbf{\Sigma_v}$ reduces to a block-diagonal matrix. Finally, due to the underlying, iterative structure of the problem, mainly caused by Eq. (3), updates of most matrices can be done iteratively, which can be exploited during implementation.

III. SIMULATIONS

Extensive simulations were done to evaluate different aspects of the approach. For the evaluation of each aspect 100 s segments from 50 different trajectories were simulated 5 times each. Simulation parameters were carefully chosen to represent the characteristics of true experiments. Two different errors are introduced. On the one hand, the translation error is defined by

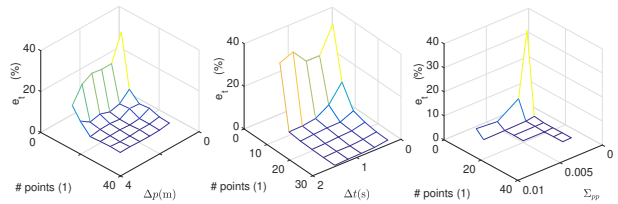
$$e_t = \frac{\|{}^c\hat{\mathbf{t}}_i - {}^c\mathbf{t}_i\|}{\|{}^c\mathbf{t}_i\|}$$

with ${}^c\hat{\mathbf{t}}_i$ being the estimated translation and ${}^c\mathbf{t}_i$ the true translation. On the other hand the rotation error is defined by

$$e_r = f_\alpha\left({}^c\mathbf{R}_i^T\ {}^c\hat{\mathbf{R}}_i\right)$$

with ${}^c\hat{\mathbf{R}}_i$ being the estimated rotation, ${}^c\mathbf{R}_i$ the true rotation and the function $f_\alpha()$ extracting the angle of a rotation matrix.

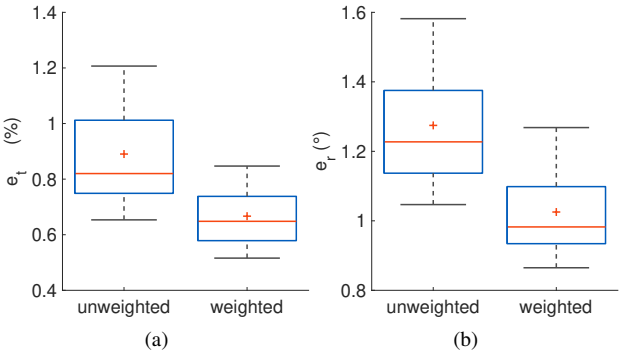
The main points that we evaluate with simulation are: (i) Influence of different noise levels on results, (ii) comparison of different strategies for adding points, (iii) comparison of different strategies for deleting points and (iv) comparison of weighted and non-weighted solution.



(a) position-based (b) time-based (c) uncertainty-based

Fig. 3: The three figures show errors for different strategies to add points to the optimization. In all three cases leverage information is used for the decision, which point to delete and the error is below 1% for a suitable choice of parameters.

Fig. 3 shows the errors for the three different strategies to add points to the optimization. For all three strategies and an appropriate choice of parameters the error is below 1%. In the remainder of the paper, a combination of position and time-change-based strategies is used. The threshold for position change is set to 1.5 m and for the time based strategy to 0.9 s. In addition, the number of points used for optimization is limited to $n = 15$.



(a) (b)

Fig. 4: Influence of using weights on translation and rotation errors. The + sign indicates the median and the horizontal bar — indicates the mean of each box. In (a) the translation error e_t is depicted and in (b) the rotation error e_r in degree.

Fig. 4 shows the effect of considering VO uncertainty information on the error of the estimated trajectory. The weighted solution uses uncertainties associated with camera points ${}^{2d}\Sigma_\mathbf{v}$ and with integrated odometry readings ${}^{3d}\Sigma_\mathbf{v}$, while in the unweighted case only ${}^{2d}\Sigma_\mathbf{v}$ is used and ${}^{3d}\Sigma_\mathbf{v} = 0$. For evaluation, 50 segments of trajectories were simulated 10 times. Each segment had a length of 100 s. A new integrated odometry measurement was added every 1.5 m distance traveled. The buffer of measurements was limited to 15 and the oldest measurement was always deleted to keep the buffer size constant. The intrinsic parameters of the simulated camera were the same as those of the tele camera on LRU and the distance of ARDEA was approximately 50 m. Odometry readings were created at a frequency of 10 Hz and zero-mean Gaussian noise was added that would on average result in a Relative Pose Error (RPE) [14] of $0.03\frac{\text{m}}{10\,\text{m}}$ for translation and $0.01\frac{\text{rad}}{10\,\text{m}}$ for rotation. By using weights the mean translation error could be reduced by 24% and the mean rotation error by 19%. The standard deviation is reduced by 37% and 34%, respectively.

In Fig. 5, the mean errors in dependency of noise levels are displayed. Noise levels from a to d were chosen in a way, that an integration of odometry readings would on average result in RPE [14] from $0.05\frac{\text{m}}{10\,\text{m}}$ for translation and $0.01\frac{\text{rad}}{10\,\text{m}}$ for rotation to $0.56\frac{\text{m}}{10\,\text{m}}$ and $0.15\frac{\text{rad}}{10\,\text{m}}$. For all evaluated noise levels, the errors e_t and e_r were reduced by the weighted approach. Based on the results from Fig. 4 and Fig. 5 a distinct advantage of using 3d uncertainties is visible. But the additional workload has an impact on processing times. Compared to the algorithm proposed by [11], the runtime is ≈10 times slower. Nevertheless, our algorithm still runs in real time with measurements from the sensors used on ARDEA and LRU.

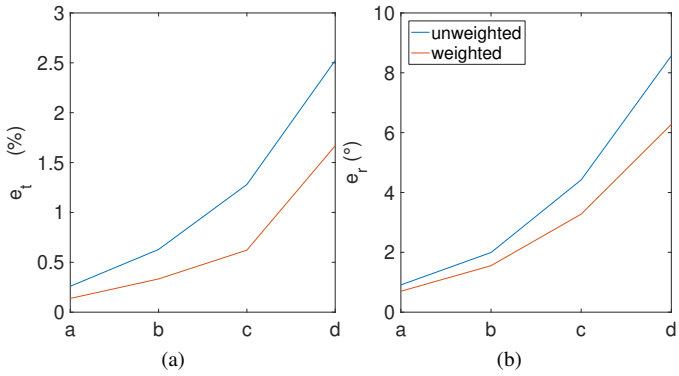


Fig. 5: Influence of noise level on mean translation and rotation error. The noise level increases from a to d.

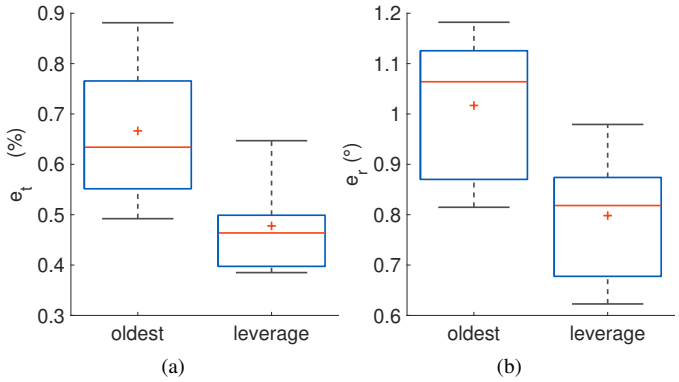


Fig. 6: Influence of different point deletion criteria on translation and rotation errors of the estimated transformation. In (a) the translation error e_t is depicted and in (b) the rotation error e_r in degree.

In Fig. 6, results are shown for the two different strategies for deleting points from the buffer of measurements used for estimating the transformation. By switching from simply deleting the oldest point in the buffer to the leverage criterion, the translation and rotation errors are reduced by 27% and 21% and standard deviations by 32% and 13%. This shows the clear benefit of using leverage information when deciding which point to remove.

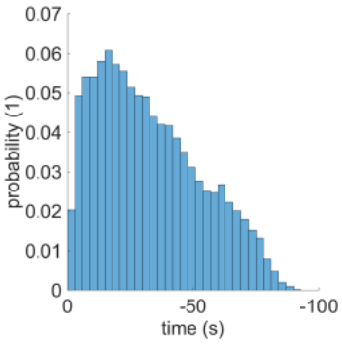


Fig. 7: Distribution of delayed measurements being used in the optimization of the transformation

When using the leverage-based criterion, the buffer of measurements covers different time spans. The time span represented in the buffer is influenced by different parameters, e. g. the course of the trajectory followed and the distribution of the uncertainty of measurements acquired along the trajectory. In Fig. 7 the probability of a measurement being part of the buffer over the amount of time passed since the measurement was taken is shown. When the time passed since the measurement was taken is around 15s, the probability has a peak. Afterwards it declines. Most recent measurements have a higher probability of being removed from the buffer, which indicates that measurements could be added less frequently to the buffer. The latest measurement is not displayed in the figure as it is always used. In phases of good VO measurement quality, indicated by low uncertainties, the observations in the buffer cover a bigger time span than during phases of high VO uncertainty.

IV. EXPERIMENTS

During an indoor experiment ARDEA was flying several times. Each time for approximately 50s. The main benefit of the lab environment is the possibility to record ground truth data along with the robot data.

In Fig. 8 the tracked and the reference trajectory are depicted as an overlay to the initial image of the tracking camera on LRU. The estimated poses are shown in Fig. 9. Determining the positions of ARDEA and their uncertainties in the image was done manually. Shortly after take off, when the buffer of measurements is filled, the pose between LRU and ARDEA can be determined. The times at which the optimization was carried out are indicated by ✳ symbols. Between two consecutive optimizations, the transformation between both systems is updated with integrated VO readings. There are no optimizations carried out for a time span of 2.5 s beginning after approximately 35 s. This corresponds to the time, when ARDEA was not visible in the camera and can also be seen in Fig. 8, where both trajectories leave the field of view of the camera. In general, the errors of the transformation are relatively small and highest perpendicular to the image plane of the tracking camera.

V. CONCLUSIONS

In this work, we estimated the transformation between a static camera and a moving object by minimizing errors between 3d model points and 2d camera observations by projecting them onto respective tangent planes on the unit sphere. Instead of assuming a predefined model of the object, we spawn the model over time and not only consider 2d uncertainties of camera measurements, but also 3d uncertainties of the model points. The 3d model points are calculated based on integration of VO observations. Therefore, they are highly correlated and potentially change with each observation. To keep computation times limited, the number of measurements used for the optimization is limited. Different criteria for the addition of new points to the optimization and also for the deletion of points from the buffer are proposed and evalulated.

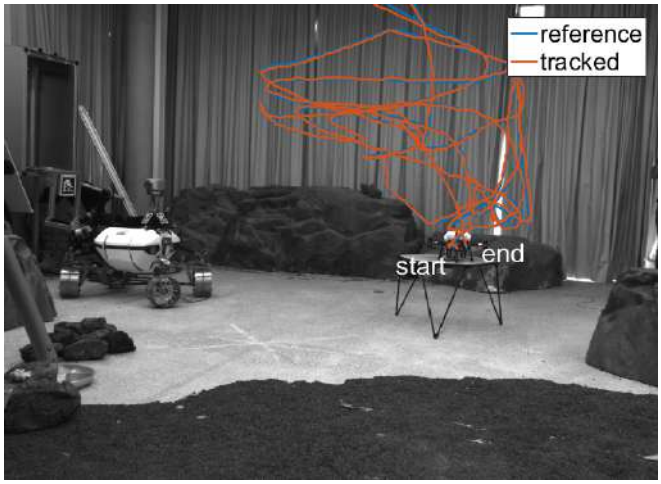


Fig. 8: Reference and tracked trajectories of ARDEA overlayed on the initial image of the dataset. The reference trajectory was measured by an external tracking system and projected into the camera reference frame of LRU. In addition to ARDEA, a second LRU is visible in the image. It is not used during the experiment.

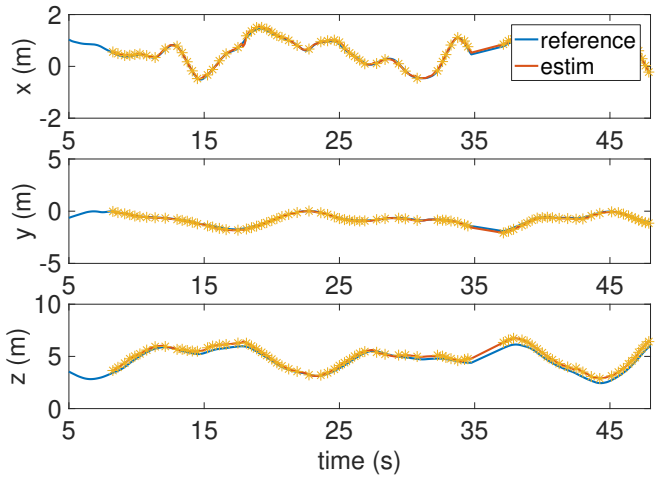


Fig. 9: Reference and estimated trajectories of ardea. The $*$ symbol indicates that the optimization was executed.

The benefits of the proposed approach were shown in simulation and in experiments conducted in a lab environment.

Investigating the implications of a moving tracking camera could be an extension to the presented approach.

References

[1] L. Ferraz Colomina, X. Binefa, and F. Moreno-Noguer. Leveraging feature uncertainty in the PnP problem. In *British Machine Vision Conference*, pages 1–13, 2014.

[2] W. Förstner. Minimal representations for uncertainty and estimation in projective spaces. In *Asian Conference on Computer Vision*, pages 619–632. Springer, 2010.

[3] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (DLS) method for pnp. In *International Conference on Computer Vision*, pages 383–390. IEEE, 2011.

[4] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[5] P. Lutz, M. G. Müller, M. Maier, S. Stoneman, T. Tomić, I. von Bargen, M. J. Schuster, F. Steidle, A. Wedler, W. Stürzl, et al. ARDEA—an MAV with skills for future planetary missions. *Journal of Field Robotics*, 37(4):515–551, 2020.

[6] M. A. Mehralian and M. Soryani. EKFPnP: Extended kalman filter for camera pose estimation in a sequence of images. *IET Image Processing*, 14(15):3774–3780, 2020.

[7] M. Müller, F. Steidle, M. J. Schuster, P. Lutz, M. Maier, S. Stoneman, T. Tomic, and W. Stürzl. Robust visual-inertial state estimation with multiple odometries and efficient mapping on an MAV with ultra-wide FOV stereo vision. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3701–3708. IEEE, 2018.

[8] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC press, 1994.

[9] M. J. Schuster, M. G. Müller, S. G. Brunner, H. Lehner, P. Lehner, R. Sakagami, A. Dömel, L. Meyer, B. Vodermayer, R. Giubilato, et al. The ARCHES space-analogue demonstration mission: Towards heterogeneous teams of autonomous robots for collaborative scientific sampling in planetary exploration. *IEEE Robotics and Automation Letters*, 5(4):5315–5322, 2020.

[10] R. T. St Laurent and R. D. Cook. Leverage and superleverage in nonlinear regression. *Journal of the American Statistical Association*, 87(420):985–990, 1992.

[11] S. Urban, J. Leitloff, and S. Hinz. MLPnP–a real-time maximum likelihood solution to the perspective-n-point problem. *arXiv preprint arXiv:1607.08112*, 2016.

[12] A. Vakhitov, L. Ferraz, A. Agudo, and F. Moreno-Noguer. Uncertainty-aware camera pose estimation from points and lines. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4659–4668, 2021.

[13] A. Wedler, M. Vayugundla, H. Lehner, P. Lehner, M. J. Schuster, S. G. Brunner, W. Stürzl, A. Dömel, H. Gmeiner, B. Vodermayer, et al. First results of the ROBEX analogue mission campaign: Robotic deployment of seismic networks for future lunar missions. In *International Astronautical Congress*, volume 68. International Astronautical Federation, 2017.

[14] Z. Zhang and D. Scaramuzza. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251, 2018.

Appendix

In Eq. (6) the matrix $\mathbf{B} \in \mathbb{R}^{3(n+1)\times 6n}$ to project uncertainties of single VO measurements to uncertainties of 3d points is introduced. It can be derived with the error model

$$\begin{aligned} {}^{c}\mathbf{R}_{i} &= {}^{c}\hat{\mathbf{R}}_{i}\,{}^{c}\delta\mathbf{R}_{i} \\ {}^{c}\mathbf{t}_{i} &= {}^{c}\hat{\mathbf{t}}_{i} - {}^{c}\delta\mathbf{t}_{i}, \end{aligned} \tag{8}$$

where ${}^{c}\hat{\mathbf{R}}_{i}$, ${}^{c}\hat{\mathbf{t}}_{i}$ are estimated quantities, ${}^{c}\mathbf{R}_{i}$, ${}^{c}\mathbf{t}_{i}$ are true quantities and ${}^{c}\delta\mathbf{R}_{i}$, ${}^{c}\delta\mathbf{t}_{i}$ are rotation and translation errors. Rotation errors $\delta\phi \in \mathbb{R}^{3\times 1}$ are locally defined and a linear approximation of ${}^{c}\delta\mathbf{R}_{i} = I + \lfloor\, {}^{c}\delta\phi_{i} \,\rfloor_{\times}$ is used for covariance propagation. The operator $\lfloor\ \rfloor_{\times}$ for a vector $\mathbf{t} \in \mathbb{R}^{3\times 1}$ is given by

$$\lfloor\, \mathbf{t} \,\rfloor_{\times} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

By plugging the error model from Eq. (8) into Eq. (3), the matrix $\mathbf{B}$ can be derived

$$\mathbf{B} = \mathrm{blkdiag}\left({}^{c}\hat{\mathbf{R}}_{i}\right) \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_{2n} \end{bmatrix} \tag{9}$$

with $\mathbf{b}_h \in \mathbb{R}^{3(n+1)\times 3}$ and $h \in [1, 2, ..., 2n]$. The function $\mathrm{blkdiag}\,(\mathbf{X})$ creates a block-diagonal matrix of appropriate size from $\mathbf{X} \in \mathbb{R}^{3\times 3}$. In Eq. (9) the size of $\mathrm{blkdiag}\left({}^{c}\hat{\mathbf{R}}_{i}\right)$ is therefore $3(n+1) \times 3(n+1)$.

The first submatrix $\mathbf{b}_1 = \begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{I}_{3\times3n}\end{bmatrix}^T$. For the remaining odd indices $h \in [3, 5, ..., 2n-1]$ the sub matrices $\mathbf{b}_h$ are given by

$$\mathbf{b}_h = f_d \left(\prod_{k=1}^{\frac{h+1}{2}} {}^{i-k+1}\hat{\mathbf{R}}_{i-k} \right) \underbrace{\begin{bmatrix} \mathbf{0}_{\frac{3}{2}(h+1)\times3} \\ \mathbf{I}_{3\times3} \\ \vdots \\ \mathbf{I}_{3\times3} \end{bmatrix}}_{\mathbf{b}_{h,r}}. \tag{10}$$

In $\mathbf{b}_{h,r}$ from Eq. (10), the identity matrix $\mathbf{I}_{3\times3}$ is repeated $n+1-\frac{h+1}{2}$ times. For even indices $h \in [2, 4, ..., 2n]$ the sub matrices $\mathbf{b}_h$ are given by

$$\mathbf{b}_h = f_d \left(\prod_{k=1}^{\frac{h}{2}} {}^{i-k+1}\hat{\mathbf{R}}_{i-k} \right) \begin{bmatrix} \mathbf{0}_{(3+\frac{3}{2}h)\times3} \\ \mathbf{C}_h \end{bmatrix}.$$

The matrix $\mathbf{C}_h \in \mathbb{R}^{3(n-\frac{h}{2})\times3}$ can be developed iteratively according to the following scheme

$\mathbf{C}_{h,1} \leftarrow \lfloor {}^{i-h+1}\mathbf{t}_{i-h} \rfloor_\times$
for $k \leftarrow h+1$ to n **do**
  $\mathbf{C}_{h,k+1} \leftarrow \mathbf{C}_{h,k} + {}^{i-k+2}\hat{\mathbf{R}}_{i-k+1} \lfloor {}^{i-k+1}\mathbf{t}_{i-k} \rfloor_\times$
end for

Each block $\mathbf{C}_{h,k}$ refers to a 3×3 matrix and overall $\mathbf{C}_h$ is created by stacking them according to

$$\mathbf{C}_h = \begin{bmatrix} \mathbf{C}_{h,k} \\ \vdots \\ \mathbf{C}_{h,n} \end{bmatrix}$$

The sub matrix $\mathbf{b}_h$ from Eq. (9) with odd indices h refer to the propagation of translation uncertainty of a VO measurement and for even indices h it refers to propagation of rotation uncertainty.

Human-centered Benchmarking for Socially-compliant Robot Navigation

Iaroslav Okunevich[1], Vincent Hilaire[1], Stephane Galland[1],
Olivier Lamotte[1], Liubov Shilova[2], Yassine Ruichek[1], and Zhi Yan[1*]

***Abstract*—Social compatibility is one of the most important parameters for service robots. It characterizes the quality of interaction between a robot and a human. In this paper, a human-centered benchmarking framework is proposed for socially-compliant robot navigation. In an end-to-end manner, four open-source robot navigation methods are benchmarked, two of which are socially-compliant. All aspects of the benchmarking are clarified to ensure the reproducibility and replicability of the experiments. The social compatibility of robot navigation methods with the Robotic Social Attributes Scale (RoSAS) is measured. After that, the correspondence between RoSAS and the robot-centered metrics is validated. Based on experiments, the extra robot time ratio and the extra distance ratio are the most suitable to judge social compatibility.**

***Index Terms*—Social navigation, human-robot interaction, benchmarking**

I. Introduction

The development of computing and sensing technologies allows us to apply mobile robotic systems in different environments. Robot behavior is especially important in an environment with human presence, such as in the case of mobile robots for emerging logistic [1] or disinfection [2] purposes. In these cases, socially-compliant robot navigation [3] is one of the main requirements that guarantees a high-quality human-robot interaction (HRI).

Although robot systems perform relatively well, people still tend to fear them, which negatively affects mental health and decreases the productivity of workers [4]. The problem behind fear is the lack of understanding of robot behavior [5]. Robotic intelligence is different from that of humans, and human-robot interaction is limited in ways of communication compared to human-human interaction. People feel safer in the presence of other people, thus preferring them to robots as their working partners. The feeling of safety comes from the belief that people's behavior is more predictable. Similarly, one generally feels uneasy when communicating with a drunk person, as alcohol makes their behavior unpredictable.

To make the behavior of the robot more understandable, one could apply different engineering solutions. In addition to the sensors necessary to perceive the world, the robot can be equipped with mechanical elements to show its behavior or intention, such as the light [6] and sound [7] signaling system or an additional screen [8]. Another way to reduce robot fear is to improve the quality of navigation algorithms. This implies that the robot tries to follow the unspoken social rules that people have in their regular life. For instance, the left- and right-hand rules to avoid collisions [9], social zones around people [10], and navigation through pedestrian flow [11].

Fig. 1. The experiment to examine the social compatibility of robot navigation. The person works in a room, while the mobile robot moves nearby. The logistic operations at a warehouse are an example of a real scenario, where human workers and autonomous mobile robots need to collaborate with each other and navigate in a shared space.

However, evaluating the social effectiveness of socially-compliant navigation methods can be challenging. Many studies [9], [10] apply robot-centered metrics (RCM) to assess the quality of the social part of navigation methods. These metrics are numerical and usually measure robot functionality as a reference. For example, the speed of the robot or the length of the traveled path. As fear is not a numerical parameter, scientists also need to use psychological metrics to assess the acceptance of the robot by people. The latter can be regarded as compatibility from a robot perspective. We suggest therefore to use "social compatibility" (SC), rather than "social acceptance" used in the literature, which characterizes the

This work was supported by the Bourgogne-Franche-Comté regional research project LOST-CoRoNa.
[1]UTBM, CIAD UMR 7533, F-90010 Belfort, France. firstname.lastname@utbm.fr
[2]Center for Bioinformatics, Saarland Informatics Campus, Saarbrücken, Germany. lish00001@stud.uni-saarland.de
*Corresponding Author.

979-8-3503-0704-7/23/$31.00

social effectiveness of robotic navigation methods. The high SC value of a navigation method implies that, in a social environment, a robot moves in an efficient, safe and socially acceptable manner [3].

One of the most popular approaches to measure SC is to invite people to participate in an experiment (such as that shown in Fig. 1) and then conduct a questionnaire for the participants. As questions are used as metrics to evaluate human feelings, they are called human-centered metrics (HCM). However, in different articles various metrics and experimental settings are applied to assess the interaction between robots and humans. Consequently, reproducing these experiments is often not straightforward, making comparisons between different methods tricky.

The contributions of this paper are twofold.

- We propose an end-to-end human-centered benchmarking framework. To confirm our idea, we benchmark four open-source robot navigation methods under the proposed framework. Two of these methods have been developed to be socially-compliant. All experimental settings and parameters are clearly stated to ensure the reproducibility and repeatability of the experiments. The software-hardware integration scheme is publicly available to the community[1].
- We evaluate different methods using both HCM and RCM and report the experimental results. We gain insight that some RCMs are suitable for assessing SC while others are not, if considered for HCM. This provides a basis for clarifying the connection between RCM and HCM.

II. Related Work

Much work has been done on socially aware robot navigation, as well as interaction between humans and autonomous mobile robots. However, the applied experiment conditions (e.g. hardware, software, environment, etc.) and metrics to measure method performance vary from paper to paper significantly.

[9] focused on a multi-agent collision avoidance algorithm that exhibits socially-compliant behavior. The authors trained their algorithm in a reinforcement learning framework and compared it with two algorithms in simulation. They chose three performance metrics: 1) average extra time to reach the goal; 2) minimum separation distance to other agents; 3) relative preference between left-handedness and right-handedness. Although the experiment in real life proved that the method developed was safe, the work did not show the opinion of the people about the behavior of the robot. [12] compared standard and social navigation strategies for efficient robot behavior. For a person and a robot moving in the corridor, the following metrics were recorded: 1) the speed of the robot and the person during the experiment. Higher speed indicates a more efficient HRI. It was shown to be a useful metric to measure the difference in HRI representing the changing human behavior; 2) the signaling distance between the person and the robot. For the human, it was measured when the person started to change their trajectory to react to the robot. For the robot, it was measured when the robot started to avoid the person. This metric was shown to be suitable for a perception system but not for HRI.

Another way to evaluate socially aware robot navigation is to use simulations. This has the advantage of repeatability of the experimental conditions for each evaluated navigation method. In addition, simulated experiments often do not require real participants, which decreases the cost of the study. [13] presented a grounded simulation framework to evaluate social navigation. This simulator included pre-recorded pedestrian trajectory datasets in different scenarios. Despite the effectiveness of the proposed framework, the simulator included only RCM and could not provide any information on HCM.

[14] developed a 29-question HRI measurement questionnaire to assess how humans feel about robots. Questions were asked in five groups: anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety. The answers are ranked from 1 to 5, with 1 being the worst and 5 being the best opinion mark. This questionnaire has been used as a baseline for numerous questionnaires in HRI research [15]. However, [16] criticized the Godspeed questionnaire [14]. Through the exploratory factor analysis (EFA), it was shown that the Godspeed questionnaire has been loaded onto three unique factors, while originally this questionnaire was designed for the five factors/groups. Therefore, based on the Godspeed questionnaire, Carpinella *et al.* developed RoSAS. It consisted of 18 questions, which were chosen from the psychological literature on social cognition. Despite these questionnaires being one of the ways to represent HRI, their application leads to limited autonomy, since a robot itself cannot assess it.

[15] presented the design of the user study for the experimental evaluation of mobile robot navigation strategies in human environments. The authors applied different RCM to define the most suitable navigation strategies for HRI, such as average acceleration and energy, minimum distance between robots and humans, irregularity of the path, efficiency of the path, time spent per unit of length of the path, and topological complexity. After the experiment, the participants evaluated HRI during the experiment through a questionnaire. The combination of the results of two different types of metrics allowed HRI measurement by RCM and confirmed the results by comparing the responses to the questionnaire (i.e. HCM). The work provides immensely valuable input regarding the evaluation of mobile robot navigation strategies in a controlled lab environment. However, it could also be noted that the questionnaire used was later criticized by [16]. [17] studied how different robot navigation strategies are perceived by users in terms of comfort, safety, and awareness. Their results demonstrated some correlation between safety and comfort and the distance between the robot and the pedestrian when the robot passed the intersection.

From the survey, it became clear that there is still much to be done about the benchmarking methods and standardizable

[1] https://github.com/Nedzhaken/human_aware_navigation

metrics for socially-compliant navigation. The existing evaluation mainly uses RCM. However, it is not completely clear how these metrics reflect the SC. Furthermore, the lack of necessary experimental information makes benchmarking of the community difficult. The status quo drives us to develop reproducible experiments based on standardizable processes to accelerate the development and comparison of relevant methods in our community. In the current work, our aim is to develop such an experiment and explore the correlation between RCM and HCM for the SC parameter.

III. Benchmarking Framework

HRI benchmarking is often very challenging. This is due to, on the one hand, the increasing complexity of the robotic system (both hardware and software) and, on the other hand, the unforeseeable and unpredictable behavior of different participants with different understandings of the experimental procedures, which makes benchmarks difficult to reproduce. To this end, we propose an end-to-end benchmarking framework (similar to black-box testing in software engineering), focusing on human-centricity that allows rapid and efficient evaluation and comparison of the performance of different socially-compliant navigation methods, by clearly defining experimental scenarios and evaluation metrics. Applying HCM ensures that human opinion is one of the criteria of evaluation, which makes our framework human-centered. Moreover, we propose to divide the experiment into explicit and as small steps as possible, ideally consisting of simple motion or action primitives, which make it easier to reproduce and avoid any ambiguity. For example, the instruction to a person could be "go straight forward for three meters at normal speed to point B" rather than "go to point B". Based on this principle, we propose the following experimental design.

A. Experiment Design

Unlike the non-object experiments commonly seen in the literature [9], [17]–[22], our experiment required humans to move cartons. This task was inspired by the industrial example in which workers carry boxes in factories, warehouses, or supermarkets. We tried to reproduce the situation in which a person should complete a working task in the presence of the robot. This setting helps us to avoid bias in HCM results. According to research in the field of sociology, people are less likely to pay attention to robots when they concentrate on their tasks [23]. Therefore, experiments can provide an objective and impartial assessment of SC performance, which is beneficial for comparing different methods. Specifically, in a 2.5×4 m room, trial participants were asked to carry three cartons from one side of the room to the other (see Fig. 2). During this period, the robot moved in the shared space. The robot's acceleration and maximum velocity of the robot were set to 0.3 m/s^2 and 0.3 m/s, respectively. Humans were told to move at normal speed. The evaluation of the socially-compliant navigation methods included two parts: the robot path being coinciding or perpendicular to the pedestrian. We wanted to test navigation methods in three general ways of

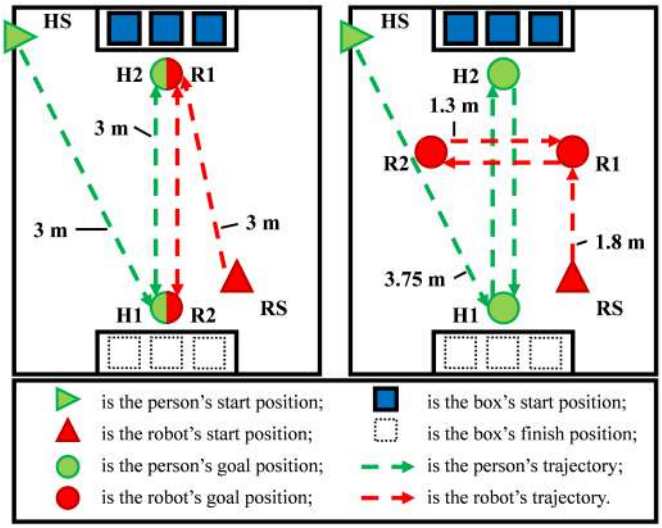


Fig. 2. Our reproducible experiment design. Shown on the left is the case where the robot's trajectory is coinciding with the human's. The coordinates of positions H2-R1 and H1-R2 are equal to ensure the crossing of the robot's and human's trajectories. Shown on the right is the case where the robot moves perpendicular to the pedestrian.

social interaction of a mobile robot with a human: *passing*, *crossing*, and *overtaking* [9]. The robot movement along the human path was used to simulate the passing and overtaking scenario, and the perpendicular robot movement was used for the crossing scenario. The passing and crossing movements can be performed by both the robot and the human. The overtaking movement was performed only by a human, as the robot's speed was chosen to be low to decrease the influence of the velocity on SC. To make our experiments reproducible and to facilitate the comparison of results between different methods, we next describe the full implementation details.

The initial position of the person was at the entrance of the room, denoted as HS. The person was asked first to reach H1 and then H2, walking in a straight line. When reaching H2, people were asked to pick up a box and take it to H1 to drop it off. This process was repeated until all cartons were transported to H1 and the experiment ended. On the other hand, the starting position of the robot was in the opposite corner of the entrance to the room, marked RS. Similarly, the robot first moved to R1 and then went back and forth between R1 and R2 four times to ensure that the person completed the task within its moving time. As shown in Fig. 2, the robot moved between R1 and R2 following the same trajectory as H1-H2 or perpendicular to H1-H2. When the robot finally reached R1, we collected the experimental RCM.

The moderately sized workspace ensures actual HRI and reliable robot navigation and allows experiments to be easily reproduced at other places. The distances between human positions were chosen to ensure the naturalness of human behavior. In the first case, the robot and human waypoints were in the same location (R1-H2 and H1-R2) to ensure that a participant interacted with the robot and did not ignore it. During the perpendicular movements of the robot, the

human could solve the task without interaction with the robot. The close positioning of R1-R2 in this case increased the probability of the intersection of trajectories.

B. Human-centered Metrics

In principle, RCM alone cannot fully describe SC, as they do not reflect people's subjective feelings about the robot's behavior. To assess human opinions, we adopted the aforementioned RoSAS questionnaire. It includes 18 questions[2], each of which is answered on a scale of 1 to 9. The questions are divided into three underlying factors: warmth, competence, and discomfort. The questionnaire provides a psychometrically validated and standardized measure of HRI. The RoSAS was applied to measure social perceptions of human, robot and blended human-robot faces [16] or human-to-robot handovers [24]. We innovatively applied the RoSAS to measure the SC of the robot navigation methods. Moreover, we wanted to demonstrate that the scale was applicable in mobile robotics to assess SC as a form of HRI.

C. Robot-centered Metrics

Five RCM metrics commonly used from the literature are selected, as well as one additional metric.

1) *The robot extra time ratio* evaluates how efficiently a robot can complete a task in an environment shared with humans [9], [25], [26], and is defined as:

$$R^r_{extra} = T^r / T^r_h, \tag{1}$$

where T^r and T^r_h are the time it takes the robot to complete the task without and in the presence of humans, respectively.

2) *The human extra time ratio* is a human analog of the previous one. It is first proposed in this paper to assess changes in human performance when working with robots. It could improve our understanding of the connection between human performance and SC. It is defined as:

$$R^h_{extra} = T^h / T^h_r, \tag{2}$$

where T^h and T^h_r are the time it takes a human to complete the task without and in the presence of robots, respectively.

3) *The extra distance ratio* evaluates system performance in terms of the distance a robot would have to travel additionally when a human is present [15], [27], and is defined as:

$$R_{dist} = D^r / D^r_h, \tag{3}$$

where D^r and D^r_h represent the distance that the robot travels to complete a task without and in the presence of a human, respectively.

4) *The success ratio* assesses the ability of a robot to complete a task without colliding with a human [9], [25], [26], and is defined as:

$$R_{succ} = N_{succ} / N, \tag{4}$$

where N_{succ} represents the number of successful trials during which the robot does not hit a human and N indicates the total number of trials.

5) *The hazard ratio* assesses the time that a robot gets too close to a human [26], which is defined as:

$$R_{haza} = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{T_i^{hazard}}{T_i^{social}}, \tag{5}$$

where n is the number of people, T_i^{hazard} is the duration of time when the distance between the robot and the i-th person is less than the safe distance (denoted as D_{safe}), and T_i^{social} is the duration of time when the distance between the robot and the i-th person is less than the social distance (denoted as D_{social}). In our experiments, $D_{safe} = 0.2\ m$ and $D_{social} = 0.4\ m$.

6) *The deceleration ratio* evaluates a robot's ability to slow down when approaching a human [12], which is defined as:

$$R_{dec} = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{V_i}{V^{max}}, \tag{6}$$

where n represents the number of speed measurements when the robot is less than D_{social} from the human. V_i represents the instantaneous speed of the robot at i-th measurement, and V^{max} is the maximum speed of the robot ($0.3\ m/s$). The maximal velocity was kept the same for all methods. Although the different methods can work with different maximal velocities, variations in this parameter would complicate the analysis. It would be difficult to understand whether the maximal velocity or the algorithm itself affects the SC.

IV. Experiments

Our experiments aimed to benchmark four open-source robot navigation methods. Two of them were developed as socially-compliant. This, on the one hand, showed the effectiveness of the proposed benchmarking framework and, on the other hand, revealed the connection between RCM and HCM.

A. Experimental Platform

For the experiment, we used a mobile robotic platform. The robot chassis is a Clearpath Jackal UGV. The perception system includes four RGB-D cameras and a 3D lidar. The RGB-D cameras are placed toward all sides of the robot for a panoramic view. The 3D lidar allows people detection and tracking under different lighting conditions. The robot is equipped with a 2D lidar that has higher measurement frequency, accuracy, and resolution compared to the 3D lidar. It is beneficial for robot localization and collision-free navigation. The software system has been fully implemented in ROS [28] with high modularity and is publicly available to the community.

[2]https://github.com/Nedzhaken/human_aware_navigation

B. Evaluated Methods

We deployed several open-source methods and reported results on four of them. The choice was based on two factors: 1) the method must be deployable on real robots, and 2) the effectiveness of the method must have been confirmed in its corresponding paper. Two of these methods are socially-compliant robot navigation methods and two are traditional navigation approaches that include only collision avoidance mechanisms.

- *Social Navigation Layers (SNL)*[3] [29]: This method implements a Gaussian mixture model around the detected person on the navigation cost map. The extra cost area around the person makes the robot consider avoiding it when planning its path. This allows the robot to demonstrate better social attributes during navigation. Also, if the person moves, the social area grows in the direction of the movement (i.e., from a circle to an ellipse). In our experiments, according to the characteristics of the working environment, the social radius was set to be 0.4 *m* centered on the person.
- *Time Dependent Planning (TDP)*[4] [30]: This method is similar to SNL, except that the social area is no longer limited to a person's current location, but also includes their predicted location several time steps in the future, based on a constant velocity model.
- *Collision Avoidance with Deep Reinforcement Learning (CADRL)*[5]: This method is the underlying implementation of the well-known SA-CADRL (socially aware CADRL) [25], while the latter has not been ROSified. However, it is still considered a baseline, as collision avoidance is one of the most fundamental elements in the social properties of robot navigation.
- *move_base (MB)*[6]: This is a basic component provided by the ROS navigation stack and does not contain any socially-compliant modules.

Additionally, we added the human-human interaction to understand the difference between a robot and a human interaction in the terms of HCM.

- *Human-human interaction (HH)*: In this case, the robot is replaced by a human who performs the task assigned to the robot, that is, moving from one point to another.

The results of the RCM were recorded during the execution of the above methods by the robot, and the participants were asked to complete the questionnaire after each method to assess the HCM.

C. Participants

The recruitment was carried out within the University of Technology of Belfort-Montbéliard (UTBM) in France. Twenty volunteers (14 men, 6 women), aged 18 to 39 years [*M* = 27.10, *SD* = 5.30] participated in the experiment. Participants were not rewarded in this research. To avoid carry-over effects, the methods of the experiment were counterbalanced among participants by applying a Latin square design [24].

[3]https://github.com/DLu/navigation_layers

[4]https://github.com/marinaKollmitz/human_aware_navigation

[5]https://github.com/mit-acl/cadrl_ros

[6]https://github.com/ros-planning/navigation

D. Experimental Results

As RoSAS had not been used before to measure SC of a mobile robot, we performed an internal consistency (IC) test, which allows us to confirm the results of the EFA performed in the original investigation [16]. Specifically, the IC measures how closely the RoSAS questions match three factors (warmth, competence, and discomfort) by applying the data from our experiment. For the test, Cronbach's alpha should be more than 0.90 to represent high IC [31]. Cronbach's alphas of warmth ($\alpha_{Cronbach} = 0.94$), competence ($\alpha_{Cronbach} = 0.94$), and discomfort ($\alpha_{Cronbach} = 0.92$) satisfied this condition. Thus, the factors have relatively high IC with their respective questions. For the analysis of RoSAS, six questions were averaged that comprise the dimensions of warmth, competence, and discomfort. The warmth factor includes the items: happy, feeling, social, organic, compassionate, and emotional. The competence factor includes the following elements: capable, responsive, interactive, reliable, competent, and knowledgeable. The discomfort factor includes items: scary, strange, awkward, dangerous, awful, and aggressive. The one-way ANOVA results (see Table I) show that there is a statistically significant difference between the methods evaluated for each HCM and applied RCM ($p < 0.05$) except for R^h_{extra}.

TABLE I
ANOVA RESULTS OF APPLIED HCM AND RCM

Metric	Sum Sq	F value	p
Warmth	250.134	28.203	<0.001
Competence	173.484	20.495	<0.001
Discomfort	110.194	10.317	<0.001
R^r_{extra}	0.957	21.608	<0.001
R^h_{extra}	0.045	1.501	0.22
R_{dist}	0.041	3.025	0.035
R_{succ}	0.459	3.435	0.021
R_{haza}	0.104	4.052	0.010
R_{dec}	2.626	166.332	<0.001

Fig. 3 summarizes the normalized HCM results. The blue, orange, and green bars represent respectively the average rates of the warmth, competence, and discomfort factor of RoSAS. It can be seen that TDP performs best in experiments involving the robot. This is reasonable, as this method is the only one with pedestrian prediction capability, which also confirms the importance of robot foresight in socially-compliant navigation. The reason for the worst CADRL performance is the freezing movement of the robot during the experiment. The reason for that is the implementation of the open-source version of the algorithm. Therefore, it leads to aggressive motion and freezing of the robot, therefore to low rates of warmth and competence and a high rate of discomfort. Instead, HH scores for warmth and competence are much higher and for discomfort much lower than in other robot-involved methods. This reflects the

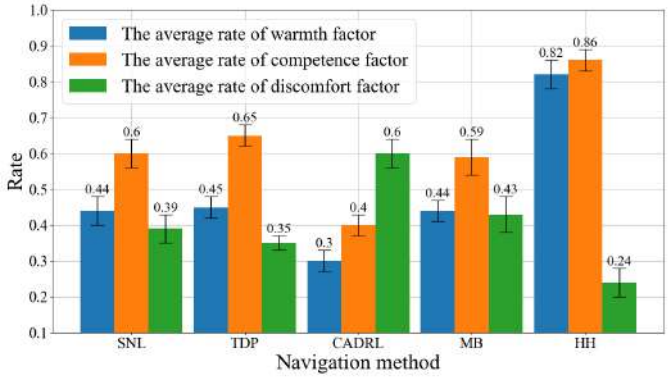


Fig. 3. Experimental results of RoSAS. The bars represent the average values of 3 questionnaire factors (warmth, competence, discomfort), normalized to $[0,1]$ with standard error ($N = 20$).

general understanding that people still find other people more socially acceptable than robots. The difference between SNL and MB is only the implementation of social zones in SNL. The close values of the warmth and competence factors of MB and SNL demonstrate that these social zones influence exclusively the discomfort factor.

The results of the experiment are presented in Table II. The gray row shows that R^h_{extra} provides values that do not vary significantly among the methods. Red and green cells are respectively the worst and best results of a metric in terms of SC. In terms of HCM, the CADRL with its freezing movements can be seen as the worst and the TDP with pedestrian prediction capability as the best method. RCM partially follows this trend. On the one hand, three out of five RCMs were indeed the worst for CADRL. On the other hand, R^r_{extra} and R_{succ} demonstrated the method to be the best. This means that, while the robot did not pose a real danger to people and did not spend extra time with them, it was still perceived as the most uncomfortable to work with. For TDP, only R_{dist} reached the best value. In line with the HCM results, this metric has the highest value in TDP and the lowest value in CADRL.

R^r_{extra} shows the inverse relation to HCM, which allows the application of the inverse value of R^r_{extra} to measure SC. The

TABLE II
EXPERIMENTAL RESULTS OF RCM AND HCM

Metric	SNL	TDP	CADRL	MB
Warmth	0.44	0.45	0.30	0.44
Competence	0.60	0.65	0.40	0.59
Discomfort	0.39	0.35	0.60	0.43
R_{haza}	0.59	0.57	0.65	0.56
R^h_{extra}	0.9	0.88	0.94	0.87
R_{dist}	0.96	1.00	0.95	0.97
R_{dec}	0.56	0.58	0.17	0.61
R^r_{extra}	0.77	0.74	1.00	0.83
R_{succ}	0.92	0.85	1.00	0.8

TABLE III
CORRELATION COEFFICIENTS OF RCM TO HCM

Metric	R_{haza}	R^h_{extra}	R_{dist}	R_{dec}	R^r_{extra}	R_{succ}
Warmth	-0.700	0.148	0.152	-0.402	0.114	0.086
Compet.	-0.620	0.304	0.156	-0.195	0.029	0.085
Discom.	0.454	-0.456	-0.197	0.059	-0.079	0.022

reason for the lowest value R^r_{extra} of TDP is the pause during movements. The mobile robot with the pedestrian prediction capability prefers to wait while the person liberates the path of the robot than trying to avoid them. In this case, the robot spends more time finishing the task but crosses fewer distances and seems to be better accepted by people. As R^r_{extra} and R_{dist} have low correlation coefficients (see Table III), the relationship between these RCM and HCM is likely non-linear.

The R_{haza}, R_{dec}, and R_{succ} do not seem to match the HCM trends when comparing the methods, although the correlation coefficients for some of them are considerable. The values of R_{haza} are similar for SNL, TDP and MB. This matches the warmth factor of HCM. As expected, the more often the robot is located near the human, the lower is the warmth and competence factors, and the greater is the discomfort. R_{dec} has a trend similar to R_{haza}. The larger decrease in speed in close proximity to the person corresponds to a worse HCM. However, as with R_{haza}, the highest speed of the robot near the participants does not correspond to the best HCM. Interestingly, R_{succ} does not reflect HCM. The reason might be the low speed of the robot in the experiment, which made collisions negligible to the participants.

Therefore, the following conclusions can be made:

- TDP has the best HCM among the robot navigation methods, because of its pedestrian prediction capability.
- HH interaction has higher values of HCM and therefore higher SC.
- When people worked with the robot, they needed more time to complete the tasks (i.e. $R^h_{extra} < 1.00$ for each method).
- While R^r_{extra} and R_{dist} reflect the HCM and can be used to judge SC, other RCM do not give a clear picture of SC. Therefore, in the experiments with mobile robots, especially when assessing human opinion is not possible, it is highly advisable to record R^r_{extra} and R_{dist} to judge the SC of the navigation method.

V. CONCLUSIONS

In this paper, we proposed a human-centered benchmarking framework for socially-compliant robot navigation with RoSAS and benchmarked four open-source approaches. The benchmarking framework is end-to-end and explicitly provides all parameters required for the reproduction of experimental results. This benchmark aims to evaluate the social part of a navigation method. Only the full survey of participants, preferably conducted with a standard questionnaire such as RoSAS, can provide the full picture of SC. However, in

situations where it is not possible, one could record RCM like R^r_{extra} and R_{dist} that reflect the SC of navigation. We suggest to apply these two metrics for the comparison of state-of-the-art and new socially-complaint robot navigation methods in simulators.

Our future work will explore new approaches for socially-compliant navigation and continue to evaluate them under the proposed benchmarking framework. Furthermore, our objective is to develop dependence functions for socially-compliant navigation methods from the most relevant RCM. This task can be done using neural networks, but more training data needs to be collected.

ACKNOWLEDGMENT

We thank RoboSense for sponsoring the 3D lidar, UTBM CRUNCH Lab for support in robotic instrumentation, and all those involved in the experiments.

REFERENCES

[1] R. Fukui, Y. Yamada, K. Mitsudome, K. Sano, and S. Warisawa, "Hangrawler: Large-payload and high-speed ceiling mobile robot using crawler," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1053–1066, 2020.

[2] S. Perminov, N. Mikhailovskiy, A. Sedunin, I. Okunevich, I. Kalinov, M. Kurenkov, and D. Tsetserukou, "Ultrabot: Autonomous mobile robot for indoor uv-c disinfection," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 2147–2152.

[3] Y. Gao and C.-M. Huang, "Evaluation of socially-aware robot navigation," *Frontiers in Robotics and AI*, vol. 8, p. 420, 2022.

[4] M. Sahin and C. Savur, "Evaluation of human perceived safety during hrc task using multiple data collection methods," in *2022 17th Annual System of Systems Engineering Conference (SOSE)*. IEEE, 2022, pp. 465–470.

[5] N. Akalin, A. Kristoffersson, and A. Loutfi, "Do you feel safe with your robot? factors influencing perceived safety in human-robot interaction based on subjective and objective measures," *International journal of human-computer studies*, vol. 158, p. 102744, 2022.

[6] R. Fernandez, N. John, S. Kirmani, J. Hart, J. Sinapov, and P. Stone, "Passive demonstrations of light-based robot signals for improved human interpretability," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2018, pp. 234–239.

[7] M. C. Shrestha, T. Onishi, A. Kobayashi, M. Kamezaki, and S. Sugano, "Communicating directional intent in robot navigation using projection indicators," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2018, pp. 746–751.

[8] J. Hart, R. Mirsky, X. Xiao, S. Tejeda, B. Mahajan, J. Goo, K. Baldauf, S. Owen, and P. Stone, "Using human-inspired signals to disambiguate navigational intentions," in *International Conference on Social Robotics*. Springer, 2020, pp. 320–331.

[9] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[10] X.-T. Truong and T.-D. Ngo, "Dynamic social zone based mobile robot navigation for human comfortable safety in social environments," *International Journal of Social Robotics*, vol. 8, no. 5, pp. 663–684, 2016.

[11] Y. Morales, N. Akai, and H. Murase, "Personal mobility vehicle autonomous navigation through pedestrian flow: A data driven approach for parameter extraction," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3438–3444.

[12] D. V. Lu and W. D. Smart, "Towards more efficient navigation for robots and humans," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1707–1713.

[13] A. Biswas, A. Wang, G. Silvera, A. Steinfeld, and H. Admoni, "Socnavbench: A grounded simulation testing framework for evaluating social navigation," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 11, no. 3, pp. 1–24, 2022.

[14] C. Bartneck, D. Kulić, E. Croft, and S. Zoghbi, "Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots," *International journal of social robotics*, vol. 1, no. 1, pp. 71–81, 2009.

[15] C. Mavrogiannis, A. M. Hutchinson, J. Macdonald, P. Alves-Oliveira, and R. A. Knepper, "Effects of distinct robot navigation strategies on human behavior in a crowded environment," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 421–430.

[16] C. M. Carpinella, A. B. Wyman, M. A. Perez, and S. J. Stroessner, "The robotic social attributes scale (rosas) development and validation," in *Proceedings of the 2017 ACM/IEEE International Conference on human-robot interaction*, 2017, pp. 254–262.

[17] S.-Y. Lo, K. Yamane, and K.-i. Sugiyama, "Perception of pedestrian avoidance strategies of a self-balancing mobile robot," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1243–1250.

[18] C. Mavrogiannis, K. Balasubramanian, S. Poddar, A. Gandra, and S. S. Srinivasa, "Winding through: Crowd navigation via topological invariance," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 121–128, 2022.

[19] S. Pirk, E. Lee, X. Xiao, L. Takayama, A. Francis, and A. Toshev, "A protocol for validating social navigation policies," *arXiv preprint arXiv:2204.05443*, 2022.

[20] Y. Cui, H. Zhang, Y. Wang, and R. Xiong, "Learning world transition model for socially aware robot navigation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9262–9268.

[21] E. Repiso, A. Garrell, and A. Sanfeliu, "People's adaptive side-by-side model evolved to accompany groups of people by social robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2387–2394, 2020.

[22] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

[23] R. Y.-T. Lo, M.-W. Suen *et al.*, "The influence of visual distraction on awareness test," *Sociology Mind*, vol. 4, no. 04, p. 259, 2014.

[24] M. K. Pan, E. A. Croft, and G. Niemeyer, "Evaluating social perception of human-to-robot handovers using the robot social attributes scale (rosas)," in *Proceedings of the 2018 ACM/IEEE international conference on human-robot interaction*, 2018, pp. 443–451.

[25] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[26] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5671–5677.

[27] Y. Gao and C.-M. Huang, "Evaluation of socially-aware robot navigation," *Frontiers in Robotics and AI*, p. 420, 2021.

[28] S. Gatesichapakorn, J. Takamatsu, and M. Ruchanurucks, "Ros based autonomous mobile robot navigation using 2d lidar and rgb-d camera," in *2019 First international symposium on instrumentation, control, artificial intelligence, and robotics (ICA-SYMP)*. IEEE, 2019, pp. 151–154.

[29] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 709–715.

[30] M. Kollmitz, K. Hsiao, J. Gaa, and W. Burgard, "Time dependent planning on a layered social cost map for human-aware robot navigation," in *2015 European Conference on Mobile Robots (ECMR)*. IEEE, 2015, pp. 1–6.

[31] J. C. Nunnally, *Psychometric Theory 2nd ed.* Mcgraw hill book company, 1978.

Improved path planning algorithms for non-holonomic autonomous vehicles in industrial environments with narrow corridors: Roadmap Hybrid A* and Waypoints Hybrid A*

Alessandro Bonetti[1], Simone Guidetti[2] and Lorenzo Sabattini[1]

***Abstract*— This paper proposes two novel path planning algorithms, Roadmap Hybrid A* and Waypoints Hybrid A*, for car-like autonomous vehicles in logistics and industrial contexts with obstacles (e.g., pallets or containers) and narrow corridors. Roadmap Hybrid A* combines Hybrid A* with a graph search algorithm applied to a static roadmap. The former enables obstacle avoidance and flexibility, whereas the latter provides greater robustness, repeatability, and computational speed. Waypoint Hybrid A*, on the other hand, generates waypoints using a topological map of the environment to guide Hybrid A* to the target pose, reducing complexity and search time. Both algorithms enable predetermined control over the shape of desired parts of the path, for example, to obtain precise docking maneuvers to service machines and to eliminate unnecessary steering changes produced by Hybrid A* in corridors, thanks to the roadmap and/or the waypoints. To evaluate the performance of these algorithms, we conducted a simulation study in an industrial plant where a robot must navigate narrow corridors to serve machines in different areas. In terms of computational time, total length, reverse length path, and other metrics, both algorithms outperformed the standard Hybrid A*.**

I. INTRODUCTION

Mobile robotics and Autonomous Mobile Robots (AMRs) have been increasingly used in recent years, with applications ranging from industrial automation to personal assistance and transportation. In particular, wheeled non-holonomic AMRs are becoming crucial for improving productivity and safety in industrial and logistics environments, such as autonomous forklifts or picking robots that operate independently in warehouses and factories. Path planning [18] is one of the main challenges of autonomous non-holonomic vehicle navigation, as it involves finding a path from a starting point to a goal, avoiding obstacles, and respecting kinematic constraints. There are numerous algorithms that can be used to solve this problem, such as Theta*-RRT [13] and Spline-based Rapidly-exploring Random Tree (SRRT) [20] as they combine the principle of RRT [10] in the sampling space with efficient spline parameterization to meet the kinematic and dynamic limitations of the robot and avoid obstacles. One of the most recent studies of this type that uses RRT to solve the path planning problem for non-holonomic vehicles is [3], where an RRT* [7] is combined with B-spline curves to obtain quasi-optimal smooth paths.

The Hybrid A* [12] algorithm is also a popular path planning method for car-like robots. It combines the classical A* algorithm [4] in discrete space with the vehicle's kinematic model and an analytic expansion using Reed-Sheep curves to obtain a non-holonomic and feasible path in continuous space. Several studies have been conducted to improve Hybrid A* performance in a variety of mobile robotics application fields. For example, in [16] a waypoint generation method for Hybrid A* using visibility graph is proposed for car parking applications in order to speed up the computation time and increase the quality of the path. Another research [17] addresses the theme of parking valet, where a multistage Hybrid A* is used to reduce the runtime required to obtain the path. The authors of [11] used a topological roadmap to implement a variable curvature approach applied to Hybrid A* in order to improve the path quality and success rate of an AMR operating in narrow known industrial environments. This study [19] addresses a common problem with Hybrid A* that produces an output that is too close to obstacles and is characterized by unnecessary steering action and oscillations. They solve the problem by applying an artificial potential field to obstacles and using it to optimize and smooth the Hybrid A* path.

Finally, [2] focused on improving the analytical expansion part of the algorithm by generating multiple Reeds-Shepp curves with different curvatures and choosing the best one that makes it safer to pass near obstacles and reduces the number of turning points.

In this paper, we present two new path planners for industrial autonomous car-like vehicles called RoadMap Hybrid A* and Waypoint Hybrid A*. RoadMap Hybrid A* combines the traditional Hybrid A* with a route finding component on fixed segments. The former provides flexibility in areas populated by obstacles, while the latter ensures robust paths in narrow corridors and for maneuvering in and out of machines. Waypoint Hybrid A*, on the other hand, employs fixed segments only for maneuvers and uses waypoints to guide Hybrid A* within corridors. In this way, both algorithms can improve the performance of Hybrid A*, achieve accurate docking maneuvers, and solve the problem of undesirable oscillations and unnecessary steering actions in corridors, where this defect occurs very frequently. The

*This work was supported by the COLLABORATION Project through the Italian Ministry of Foreign Affairs and International Cooperation.

[1]A. Bonetti and L. Sabattini are with Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Via Amendola 2, Pad. Morselli - 42122 Reggio Emilia, Italy {alessandro.bonetti, lorenzo.sabattini}@unimore.it

[2]Simone Guidetti is with Gruppo TecnoFerrari S.p.a. con socio unico, Via Ghiarola Nuova, 105, 41042 Fiorano Modenese (MO), Italy simone_guidetti@tecnoferrari.it

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

paper is structured as follows: Section II introduces the operating environment of the mobile robot and the challenges of Hybrid A* that led to this study. Section III describes the Roadmap Hybrid A* and Waypoint Hybrid A* algorithms and explains the steps taken to implement them. Section IV outlines the method used for developing the simulations and describes the metrics used to compare Hybrid A*, Waypoint Hybrid A*, and Roadmap Hybrid A*. Hence, the results of the simulation are presented and analyzed. Finally, section V presents the conclusions and introduces some ideas for future developments.

II. Statement of the problem

The purpose of this study is to develop an effective path planning solution for an autonomous car-like mobile robot operating within an industrial plant environment.

We consider a plant consisting of four areas where 11 machines are located, which require servicing by the AMR for loading and unloading payload operations. These zones may have pallets or goods inside waiting to be sorted, making it essential to employ a path planning algorithm that can avoid obstacles and efficiently navigate the plant. In addition, the vehicle must move between these areas through narrow corridors with a width of 3 meters that are kept clear for the robot's passage. The plant is inspired by a real case study but has been modified from the original so as to highlight the main challenges addressed in this study and simplify the stages of research development. The plant environment is represented using a grid map with a size of 50 x 50 meters and a resolution of 0.5 meters, as illustrated in Fig.1. The working space of the robot is depicted by white cells on the map, while black cells represent walls and obstacles that cannot be crossed.

The AMR used in the simulation of this research is characterized by a rectangular footprint of 2 meters long by 1 meter wide, a distance between the centers of the wheel axes of one meter, and a maximum steering angle of 45 degrees. To solve the path planning problem, we chose the Hybrid A* algorithm from the PythonRobotics [15] library.

A. Hybrid A*

Hybrid A* [12] is a path planning algorithm used in autonomous vehicle navigation. It is known as one of the most efficient path planners for non-holonomic autonomous vehicles. The goal of the algorithm is to produce an effective and collision-free path that the vehicle can follow from the start to the destination using two different phases: forward search and analytic expansion. In the forward search phase, the algorithm iteratively uses the kinematic model of the robot to generate forward motions in continuous space that depend on vehicle parameters like speed, direction, and steering angle. These continuous motions are then converted approximately to discrete coordinate nodes, i.e., the cells of the grid map, and the cost associated with each node is calculated as the maximum value from two different heuristics. The first one is the shortest distance to the goal among obstacles provided by the classic 2D A*, while the second is the length of the shortest path to the goal, ignoring obstacles but taking into account the non-holonomic nature of the vehicle. The latter includes the cost parameters related to changing direction, reversing, steering, and changing direction actions.

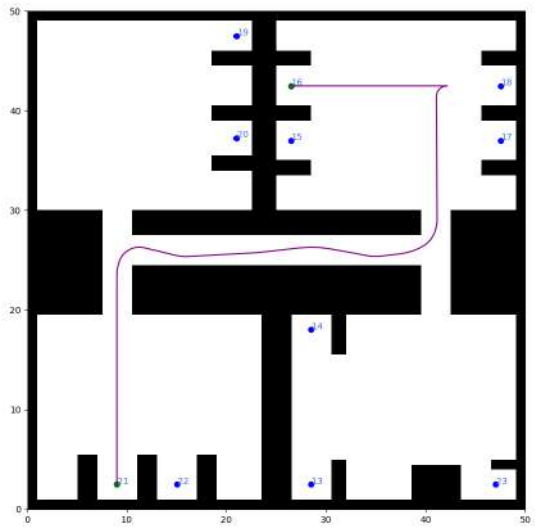


Fig. 1. Grid map of the industrial plant. The blue dots are the poses that the vehicle must reach to serve the machines. The purple path is the output of Hybrid A* that connects the start pose 21 with the end pose 16.

The node with the lowest objective value is then chosen to start the next iteration of the forward search, and this mechanism continues until the algorithm reaches the goal. However, getting to the exact continuous coordinate goal node is difficult due to the grid map's resolution and the smallest motion that the robot can take.

To overcome this issue, the analytic expansion phase of Hybrid A* is used, as it guarantees that the algorithm reaches the exact continuous coordinate of the goal state. This phase consists of generating multiple Reeds-Shepp [14] curves that respect the non-holonomic constraints of the vehicle in both forward and reverse directions. Then, the lowest cost curves based on a heuristic estimate of the remaining distance to the goal are selected in order to connect the actual node calculated by the Hybrid A* forward search phase with the target node. Analytic expansion leads to significant benefits in terms of accuracy and search speed and ensures that the algorithm reaches the exact continuous coordinate of the target state.

B. Hybrid A* issues

After we set the vehicle parameters in Hybrid A* and carefully fine-tuned the various algorithm parameters, several issues emerged. As shown in Fig. 1, the algorithm is producing an undesired oscillating path effect within the corridors, despite a high steer change cost being set. This behavior not only causes the vehicle to approach dangerously close to the walls, but is also inefficient due to increased wheel wear, energy consumption, and travel time resulting from unnecessary turns. This path effect of Hybrid A* is caused by

the costs assigned to the expanded nodes: the two algorithm heuristics with a relatively low resolution of the map, make the path fluctuate around the optimum cost cells.

In order to avoid this problem, it is important to have a well-designed objective function that accurately reflects the goals of the navigation system, but it is really difficult to find one that works efficiently both in large areas and in narrow passages. It is also preferable to have a high enough resolution to accurately represent the environment and produce a smooth and efficient path, but this leads to higher computational complexity. Because the AMR has limited hardware resources, a resolution of less than 0.5 meters is not possible in this case. Smaller cells result in a higher number of iterations of Hybrid A* required to plan a path, which is not feasible for the onboard computational capabilities. In fact, industrial vehicles often operate in dirty and dusty environments, requiring onboard PCs that can passively dissipate heat and are very robust, such as those used in the ceramic industry.

Another issue with Hybrid A* in this environment is the excessive length of the reverse portion of the path generated by the analytic expansion phase of the algorithm. Due to the nature of the Reeds-Shepp curves, the generated portion of the path can sometimes be characterized by a long reverse section. It is usually preferable for industrial vehicles to navigate in reverse as little as possible because this condition is considered less safe. The cause is due to the presence of loads and tools, which make the perception of sensors more complex and computationally demanding. Because of the limited processing power of the vehicle's computer, this situation must be minimized. For these reasons and because of the high cost of the sensors, attempts are made to concentrate the driving mostly in the forward direction.

Lastly, Hybrid A* does not provide a path that includes highly precise and repeatable maneuvers for machine entry and exit and battery charger docking. Due to the optimization process, the algorithm will produce different path shapes in the initial and final parts depending on the position of obstacles within the environment. In the industrial field, there are often non-functional requirements and demands from customers regarding the docking of mobile vehicles. To meet these requirements, it is necessary to define, control, and modify the path in advance. Hence, it is important to introduce the ability to manage the geometric shape of the initial and final parts of the path.

In the following chapter, the Roadmap Hybrid A* and Waypoints Hybrid A* algorithms will be proposed as solutions to improve performance and solve the aforementioned problems.

III. Proposed solutions

In order to overcome the issues that have arisen from the standard version of Hybrid A* in the industrial environment described in Section II, two new global path planners are presented: Roadmap Hybrid A* and Waypoint Hybrid A*. For the development of both algorithms, some preliminary steps were required. To begin, the map was divided manually into rectangular zones. This division aimed to provide a topological representation of the environment, composed of machine servicing areas and corridors. The former are represented by green rectangles, while the corridors are depicted by pink rectangles, as shown in Fig. 2. A topological graph of the plant was then set up using the NetworkX [6] library by imposing the connections between these rectangular zones.

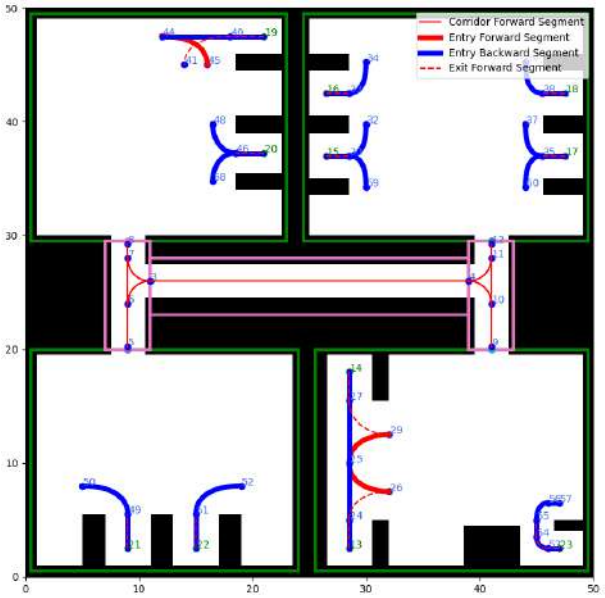


Fig. 2. The image shows the topological map of the plant, highlighting areas with green borders containing machines and narrow corridors marked in pink. The map features Bezier curve segments, with the red curves driven forward by the vehicle and the blue ones in reverse. Furthermore, the legend identifies the different types of segments used in corridors and entry and exit maneuvers.

To complete the preliminary steps required for implementing the algorithms, we manually designed the corridor segments and the machine entrance and exit segments for the AMR using Bezier curves. The corridor curves were drawn in the center to maximize the distance between the AMR and the walls, while machinery entry and exit curves were designed in order to minimize the vehicle footprint while maneuvering and ensuring safety. To accomplish this, we carefully selected the control points of each Bezier curve so as to obtain a collision-free and feasible path. The former condition is achieved by applying a collision checking algorithm that compares the bounding box of the vehicle with the grid map cells on the traversed poses. The latter condition is achieved by checking the maximum curvature of each curve segment and ensuring that consecutive segments have matching tangents to guarantee smoothness. In Fig. 2, the red segments represent the forward fixed sections of the path traveled by the AMR, while the blue segments represent the reverse ones. The blue segments are only used for the entrance to the machines, as the uploading and downloading tool is located at the back of the vehicle.

Then we collected the connectivity relationships between all the curves into a graph, on which a graph search algorithm is applied to extract the fixed path parts needed to form

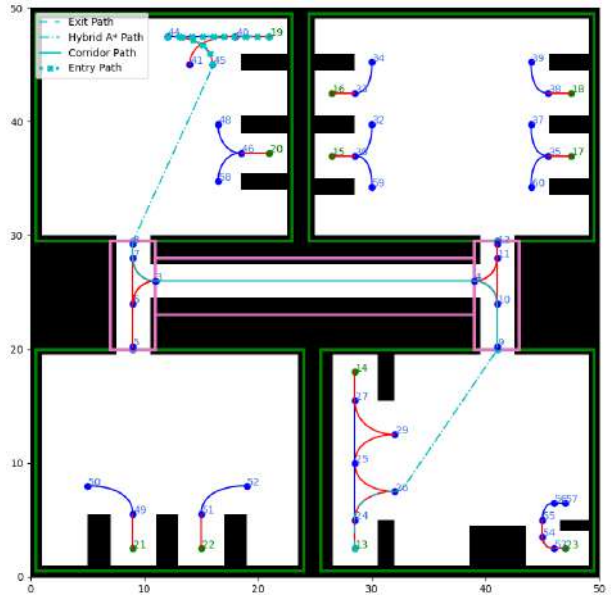


Fig. 3. Roadmap Hybrid A* path planned from node 13 to 19.The legend identifies the exit path, Hybrid A* path, corridor path and entry path parts.

the final path of the proposed algorithms. In conclusion, the topological map graph and the Bezier curve segment graph are obtained in order to implement the Roadmap Hybrid A* and Waypoint Hybrid A* algorithms, as detailed in the following subsections.

A. Roadmap Hybrid A*

In this subsection, we propose Roadmap Hybrid A*, a novel path planning technique for autonomous mobile vehicles subject to non-holonomic constraints. The technique combines two different methods: a graph search algorithm applied to fixed segments and the Hybrid A* algorithm. The former is used in obstacle-free zones and for maneuvering in and out of machines to guarantee a robust and predictable path. The latter provides flexibility and the ability to navigate around obstacles in more dynamic areas.

Roadmap Hybrid A* uses the start and goal nodes of the vehicle, the Bezier segment graph, the topological graph, and the grid map as input to initiate the planning process. The first step is to identify the entry and exit paths as well as the corresponding attachment and detachment nodes. The exit path is the fixed segment curve that allows the vehicle to safely exit the starting machine and whose end points are the start and detachment nodes, as shown in 3. Following the same principle, the entry path is the fixed curve that the AMR must travel to enter the target station and whose end points are the goal and detachment nodes. The attachment and detachment nodes serve as transition points between the free motion of the vehicle provided by Hybrid A* and the fixed path during the exit and entry processes.

The next step involves determining the start and goal areas using the Even-Odd rule [5], applying it to the rectangular areas, and using the positions of the start and goal machine nodes. If the initial area matches the target area, the detachment pose is directly connected with the attachment pose using Hybrid A*. If, on the other hand, the start and goal areas are different, Roadmap Hybrid A* uses Dijkstra's algorithm to determine the sequence of areas that the vehicle must traverse. The sequence of corridors, the Bezier curve paths within them, and the corresponding ordered end point list that the vehicle must traverse are also found at this stage. Subsequently, the fixed-path segment endpoints, as well as the attachment and detachment nodes, are connected using standard Hybrid A*: the detachment pose is linked to the first endpoint, the even-indexed endpoints are linked to the odd-indexed ones, and the last endpoint is connected to the attachment node.

Finally, the exit path, corridor path, Hybrid A* paths, and entry path are concatenated to produce the final output of the algorithm. An example of RoadMap Hybrid A* is shown in Fig. 3. Further information and the pseudocode of the algorithm can be found in [1].

B. Waypoint Hybrid A*

In this subsection, the Waypoint Hybrid A* path planner is presented. The implementation of this algorithm aimed to evaluate the cost-effectiveness of using waypoints in narrow corridors compared to the static roadmap employed in Roadmap Hybrid A*.

Waypoint Hybrid A* takes inspiration from the planner described in [16], where waypoints were generated by applying a visibility graph and then connected by means of Hybrid A*. In [16], it has been found that using waypoints to guide the Hybrid A* to its destination results in a 40% faster run-time. In this research, we propose to adapt the waypoints principle in a slightly different way in order to speed up computational time while also trying to avoid oscillating paths produced by Hybrid A*, as described in Section II.

Waypoint Hybrid A* uses the start and goal nodes of the vehicle, the Bezier segment graph, the topology graph, and the grid map of the environment as input. To begin the planning process, as explained with Roadmap Hybrid A* in Subsection III-A, the algorithm finds the entry and exit paths as well as the attachment and detachment nodes. After that, the initial and target areas are determined by applying the Even-Odd rule to the rectangular shape zones and the start and goal node positions. If the starting and target areas are different, the algorithm employs Dijkstra's algorithm to determine the sequence of areas through which the vehicle must pass. The ordered sequence of waypoints is then defined as the sequence of midpoints of the free space connection width between consecutive zones.

Subsequently, the waypoints, the attachment node, and the detachment node are connected using standard Hybrid A*. The detachment pose is connected to the first waypoint, and then each waypoint is connected with its successor except for the last one, which is connected with the attachment node. If, on the other hand, the initial area matches the target area, the detachment pose is directly connected with the attachment pose using Hybrid A*. It is worth noting that, if this condition is verified, the output of Roadmap Hybrid

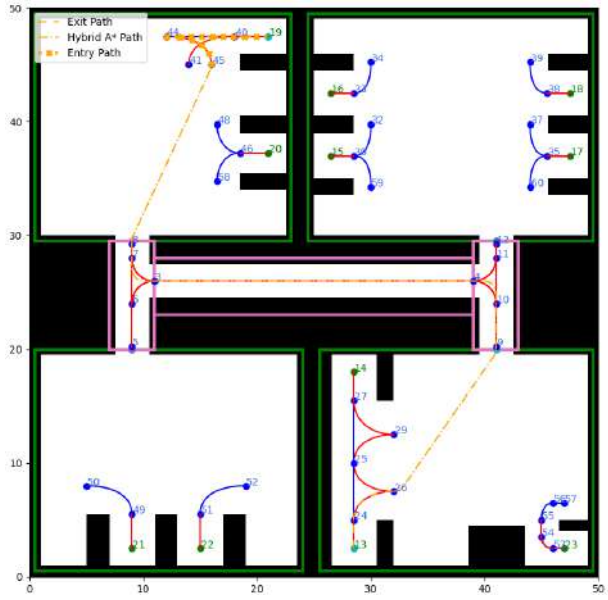


Fig. 4. Waypoint Hybrid A star path planned from node 13 to 19. The legend identifies the exit path, Hybrid A* path and entry path parts.

A* and Waypoint Hybrid A* is exactly the same. Finally, the exit path, Hybrid A* paths, and entry path are concatenated to produce the final path of the algorithm, as shown in Fig. 4. Further information and the pseudocode of the algorithm can be found in [1].

IV. Simulation and Results

This section describes the simulation approach, comparison metrics, and result analysis of Hybrid A*, Roadmap Hybrid A* and Waypoint Hybrid A*, whose operating principles have been described in the previous chapter. The simulation was implemented using Python 3.10 and was run on a laptop with the following hardware specifications: an Intel I7 12700H processor and 16 GB of RAM.

To evaluate all potential routes a vehicle could take on this map, all two-element permutations of machine nodes were simulated as source and goal node pairs (q_{star}, q_{goal}). With 11 machine stations, the two-element permutations result in 110 pairs, and for each pair, the three path planning algorithms were executed, obtaining 330 total different paths.

A. Metrics

This subsection describes the metrics calculated to evaluate the simulation results and compare the three path planners. First, the execution time of the algorithm is crucial since the AMR has limited computational resources on board and must perform multiple tasks in addition to route planning. Therefore, it is desirable to keep this value as low as possible. Total path length is an important metric that should be minimized, as shorter paths generally lead to greater energy efficiency for the robot and reduced mission time. In addition, this metric gives us information about the presence of oscillations as they produce longer paths than necessary. Reverse length was also taken into account because the loads and tools placed at the back of AMR make the perception of sensors more complex and computationally demanding. It needed to be as short as possible in order to increase safety and ensure low costs.

To obtain the last two metrics studied in this research, the Model Predictive Controller (MPC) [8], [9] path tracking algorithm was used. In particular, we choose the model predictive speed and steering control version from the PythonRobotics [15] library, which is based on a linearized vehicle model. The MPC parameters were set to ensure that the vehicle followed the planned routes as closely as possible during the simulation. To achieve this behavior, small values were set for the difference cost and input cost matrices to avoid limiting the system's inputs and make it more responsive in following the trajectory. In addition, the final state cost was also made very small to prevent the vehicle from cutting corners and make trajectory tracking more accurate.

From the MPC controller, the travel time and the maximum acceleration acting on the vehicle during the simulation are obtained. The latter is obtained by summing, for each instant, the tangential acceleration (a_t) provided by the MPC and the centripetal acceleration (a_n) calculated from the curvature (C) of the path, according to the following formula:

$$a = a_t + a_n = a_t + v^2 C. \quad (1)$$

It is desirable to minimize both of these metrics in order to increase productivity and decrease the forces acting on the vehicle and load during travel.

B. Results Analysis

After conducting the simulation, the result data was collected, and the average values of the metrics were calculated, which are shown in Table I. The results showed that, on average, RoadMap Hybrid A* has a lower computational time compared to that of standard Hybrid A* by 80 %. The reason is that the use of fixed segments in narrow passages allows for finding a robust path in a shorter amount of time, while the original algorithm must expand the search through many nodes before correctly finding the passage and building a path that respects the non-holonomic constraints of the AMR. The Waypoint Hybrid A* has an intermediate average value that is still very good for this kind of application. The use of waypoints tends to direct the algorithm towards the optimal path, resulting in a reduction in the number of searched nodes.

The total length of all three path planners is similar, with RoadMap Hybrid A* and WayPoint Hybrid A* having slightly shorter routes. This is because the two modified versions of Hybrid A* eliminate oscillations in narrow corridors, resulting in shorter paths. Both the use of fixed curves in RoadMap Hybrid A* and the use of waypoints in WayPoint Hybrid A* prove to be excellent ways to improve the performance of standard Hybrid A*, which produces longer indirect paths in tight spaces, as shown in Fig. 5.

It can also be observed in Table I that the standard Hybrid A* is characterized by a longer reverse path for

TABLE I

AVERAGE VALUE OF METRICS CALCULATED ON THE PATHS GENERATED BY HYBRID A*, WAYPOINT HYBRID A*, ROADMAP HYBRID A* FROM START AND GOAL POSE PAIRS OBTAINED FROM THE PERMUTATION OF MACHINE NODES IN THE PLANT CONSIDERED IN SECTION II.

	HA*	**Waypoint HA***	**Roadmap HA***
Computation Time $[s]$	0.26	0.11	0.05
Total length $[m]$	61.86	61.69	61.17
Reverse length $[m]$	13.93	6.93	6.93
Path time $[s]$	47.67	47.31	46.99
Max accel. $[m/s^2]$	1.81	1.80	1.79

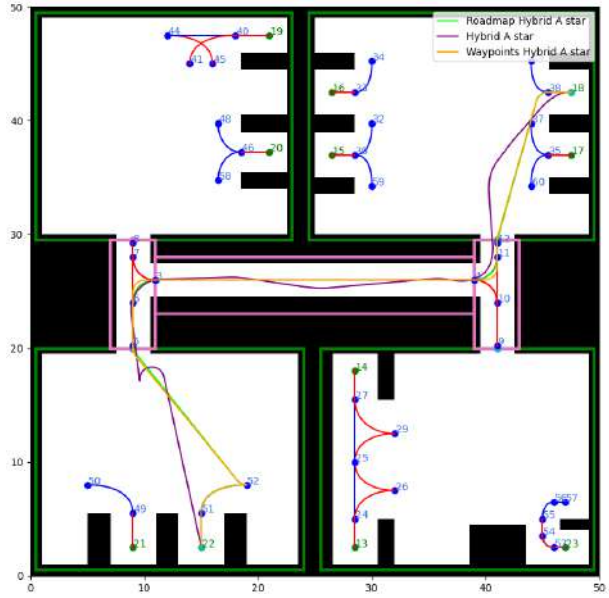


Fig. 5. Comparison between Hybrid A*, Roadmap Hybrid A* and Waypoint Hybrid A* paths.

machine entry. This is evident in the table, where the average value of reverse length for Hybrid A* is slightly more than double that of the other two solutions. As explained earlier in Subsection IV-A, this makes Roadmap Hybrid A* and Waypoint Hybrid A* the preferred solutions. The reverse path for both Roadmap Hybrid A* and Waypoint Hybrid A* is given by the fixed segments, and for this reason, it has the same length value.

The path time obtained by using the MPC follows the same considerations as for total length, with a shorter time obtained from RoadMap Hybrid A*. Finally, the maximum acceleration experienced by the vehicle is very similar in all cases. Although the differences are small, excessive load acceleration can be a safety concern if objects are not properly secured or if fragile materials are being transported.

To test the performance of the algorithms proposed in this study more objectively, a new, larger industrial environment was constructed. This allows for studying how performance scales with increasing map complexity. The new environment is characterized by dimensions of 100 meters in width and 50 meters in height, represented by a grid with a resolution of 0.5 meters. The structure of areas with serving machines and corridors remains the same as in the original facility, as shown in Fig. 6.

TABLE II

AVERAGE VALUE OF METRICS CALCULATED ON THE PATHS GENERATED BY HYBRID A*, WAYPOINT HYBRID A*, ROADMAP HYBRID A* FROM START AND GOAL POSE PAIRS OBTAINED FROM THE PERMUTATION OF MACHINE NODES IN THE LARGER INDUSTRIAL ENVIRONMENT.

	HA*	**Waypoint HA***	**Roadmap HA***
Computation Time $[s]$	0.37	0.29	0.13
Total length $[m]$	85.98	82.96	82.61
Reverse length $[m]$	20.25	6.52	6.52
Path time $[s]$	65.1	62.64	62.45
Max accel. $[m/s^2]$	1.82	1.81	1.81

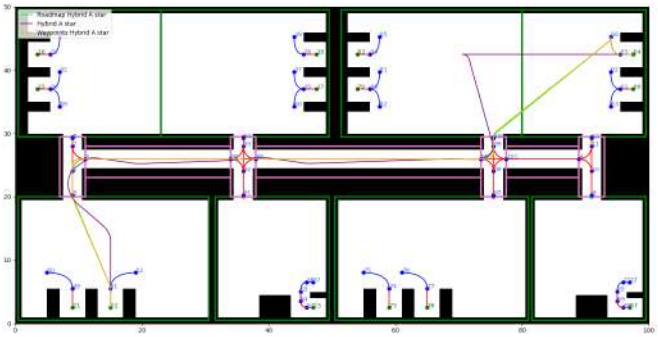


Fig. 6. Comparison between Hybrid A*, Roadmap Hybrid A* and Waypoint Hybrid A* paths in the larger scenario.

In particular, there are 14 serving stations that translate into 182 pairs (q_{star}, q_{goal}), produced by two-element permutations of the station nodes. For each of these pairs, the three algorithms were simulated, and data related to a total of 546 different paths was collected. From the collected data, the averages related to the computation time, total length, reverse length, path time, and max acceleration metrics were calculated. The values are reported in Table II.

Regarding computation time, it can be observed that the results resemble those of the original environment. Roadmap Hybrid A* confirms itself as the best of the three, while Waypoints Hybrid A* is positioned in an intermediate position. However, the differences with respect to the previous results have become smaller, with Roadmap Hybrid A* registering a 65% reduction and Waypoint Hybrid A* a 22% reduction compared to the result of the original version of Hybrid A*.

Regarding total length and reverse length, Roadmap Hybrid A* and Waypoint Hybrid A* are characterized by much lower values than Hybrid A*. The roadmap and the waypoints guided the node search, generating a path without unnecessary steering actions and guiding the reverse phase for the vehicle's entry into the machine. Even in this case, the reverse length value for Roadmap Hybrid A* and Waypoint Hybrid A* is the same because the fixed reverse segments for both algorithms are the same.

In this scenario, the path time also reflects the results obtained with the original facility, with Roadmap and Waypoint Hybrid A* showing lower values than those of the standard Hybrid A*. Finally, all three algorithms record practically identical values regarding the maximum acceleration.

In conclusion, the simulations conducted suggest that the Roadmap Hybrid A* and Waypoint Hybrid A* algorithms are more efficient and effective path planners compared to the standard Hybrid A* algorithm in terms of computation time, total path length, reverse length, and path time. By utilizing knowledge of the map and plant topology to reduce route complexity, Roadmap and Waypoint Hybrid A* algorithms can generate better routes in less time while maintaining flexibility and obstacle avoidance capability where needed. Furthermore, both algorithms address the critical aspects of eliminating Hybrid A* oscillations within the corridors and controlling machine entry and exit trajectories.

However, one disadvantage of Roadmap Hybrid A* and Waypoint Hybrid A* is that they require prior knowledge of the robot's operating environment. Additionally, the operation of these algorithms requires the design of a topological map and fixed Bezier curves, which can be time-consuming for large workspaces, particularly for Roadmap Hybrid A*. Therefore, if the AMR does not have strict constraints on the execution time of the path planning algorithm, the use of Waypoint Hybrid A* can be more convenient as it eliminates the need to draw the segments in the corridors by taking advantage of the automatically generated waypoints from the topological map. However, if drawing segments in the corridors is not a problem, as it is a one-time operation that needs to be performed, Roadmap Hybrid A* demonstrated the best performance in the simulation of the two scenarios studied in this research.

V. Conclusions

Two new global path planning algorithms for autonomous mobile robots have been developed in this study. They are both derived from the standard version of Hybrid A* and thus are made to account for the nonholonomic constraints of the robot. Both algorithms enabled the elimination of Hybrid A* oscillations within the corridors by producing smooth paths, as well as the control of machine entry and exit trajectories, which are critical requirements in industrial settings.

In terms of computational time, total path length, reverse length, travel time, and vehicle acceleration, simulation results in two industrial environments showed that both Waypoint Hybrid A* and Roadmap Hybrid A* outperformed the standard version of Hybrid A*. These results indicate that knowledge of the topology of the environment and the definition of fixed-segment curves can significantly reduce the complexity and number of nodes sought to reach the goal while improving the quality of the path and maintaining flexibility and obstacle avoidance capability.

Future research will aim to validate the results found in this research by testing Waypoint Hybrid A* and Roadmap Hybrid A* in different environments. These algorithms have the potential to enhance the performance of industrial AMR and contribute to the development of more efficient and reliable path planning methods.

References

[1] Alessandro Bonetti, Simone Guidetti, and Lorenzo Sabattini. Improved path planning algorithms for non-holonomic autonomous vehicles in industrial environments with narrow corridors: Roadmap hybrid a* and waypoints hybrid b*. roadmap hybrid a* and waypoints hybrid a* pseudocodes. https://doi.org/10.48550/arXiv.2304.14043, 2023.

[2] Chien Van Dang, Heungju Ahn, Doo Seok Lee, and Sang C. Lee. Improved analytic expansions in hybrid a-star path planning for non-holonomic robots. *Applied Sciences*, 12(12), 2022.

[3] SA Eshtehardian and S Khodaygan. A continuous rrt*-based path planning method for non-holonomic mobile robots using b-spline curves. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–10, 2022.

[4] Daniel Foead, Alifio Ghifari, Marchel Budi Kusuma, Novita Hanafiah, and Eric Gunawan. A systematic literature review of a* pathfinding. *Procedia Computer Science*, 179:507–514, 2021. 5th International Conference on Computer Science and Computational Intelligence 2020.

[5] James D. Foley, Andries van Dam, Steven Feiner, and John Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, MA, 1990.

[6] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[7] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning, 2011.

[8] Felipe Kuhne, Walter Fetter Lages, and J Gomes da Silva Jr. Model predictive control of a mobile robot using linearization. In *Proceedings of mechatronics and robotics*, pages 525–530. Citeseer, 2004.

[9] F Künhe, J Gomes, and W Fetter. Mobile robot trajectory tracking using model predictive control. In *II IEEE latin-american robotics symposium*, volume 51, 2005.

[10] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.

[11] Chengyuan Li, Dongdong Yu, Wei Lu, and Ming Li. Variable-curvature hybrid a-star search for amr path planning in limited space. In *2021 3rd International Symposium on Robotics & Intelligent Manufacturing Technology (ISRIMT)*, pages 61–65, 2021.

[12] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. *Junior: The Stanford Entry in the Urban Challenge*, pages 91–123. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[13] Luigi Palmieri, Sven Koenig, and Kai O. Arras. Rrt-based nonholonomic motion planning using any-angle path biasing. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2775–2781, 2016.

[14] James Reeds and Lawrence Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, oct 1990.

[15] Atsushi Sakai, Daniel Ingram, Joseph Dinius, Karan Chawla, Antonin Raffin, and Alexis Paques. Pythonrobotics: a python code collection of robotics algorithms, 2018.

[16] Saeid Sedighi, Duong-Van Nguyen, and Klaus-Dieter Kuhnert. Guided hybrid a-star path planning algorithm for valet parking applications. In *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, pages 570–575, 2019.

[17] Weitian Sheng, Bai Li, and Xiang Zhong. Autonomous parking trajectory planning with tiny passages: A combination of multistage hybrid a-star algorithm and numerical optimal control. *IEEE Access*, 9:102801–102810, 2021.

[18] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer Berlin, Heidelberg, Berlin, Heidelberg, 2008.

[19] Bijun Tang, Kaoru Hirota, Xiangdong Wu, Yaping Dai, and Zhiyang Jia. Path planning based on improved hybrid a* algorithm. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 25(1):64–72, 2021.

[20] Kwangjin Yang. An efficient spline-based rrt path planner for non-holonomic robots in cluttered environments. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 288–297, 2013.

Assisted Localization of MAVs for Navigation in Indoor Environments Using Fiducial Markers

André Kirsch[1], Malte Riechmann[1] and Matthias Koenig[2]

***Abstract*— Micro aerial vehicles (MAVs) are often limited due to weight or cost constraints. This results in low sensor variety and sometimes even in low sensor quality. For example, many MAVs only offer a single RGB camera to capture the environment, apart from simple distance sensors. On the other side, maps of complex environments are typically captured using depth sensors like Lidar, which are not found on such drones. For MAVs to still benefit from and use these maps, it is necessary to implement a connection layer that enables the localization of the MAV in these maps. In this paper, we propose to use fiducial markers that can be recorded by an assisting device, e.g., a mobile phone or tablet, responsible for map creation. These fiducial markers have a known pose in the map and can be detected by a drone's RGB camera to localize itself. We show that the markers are localized in the map creation process with high precision and that the drone is able to determine its pose based on detected markers. Furthermore, we present a ROS 2 based drone controller for a Ryze Tello EDU MAV that uses an occupancy voxel map for navigation.**

I. Introduction

When multiple robots or devices work together and exchange positional information, they need to operate in the same coordinate frame. For example, vacuum cleaning robots and mowing robots build their own maps. They are equipped with the necessary sensors to construct such maps of the working area in their initialization phase. When they are operating, the same sensors are then used to localize themselves in the previously created map. This is possible because the initial and previous pose of the robot in the map is known. But sharing the positional knowledge with other robots or devices is not possible until their pose in the coordinate frame of the map is also known. This problem is known as the global localization problem. A popular method for solving the problem is the particle filter, also known as Monte Carlo localization [1], where sets of pose estimates are placed into the map and evaluated in a Predict-Update-Resample loop.

But not all robots and devices share the same types of sensors. For example, occupancy maps are typically generated using range sensors like Lidar, which are not present in all devices. Micro aerial vehicles (MAVs) are often only equipped with a single RGB camera due to weight constraints that makes it unfeasible to create complex 3D occupancy maps. Still, they might need such maps for path planning and navigation. This is especially true in GPS-denied areas like indoor environments. A solution to this is to assist in the map creation process by utilizing a second device with better capabilities. The second device is able to create a much more precise map with richer information that allows for global localization of the MAV for navigation and path planning by incorporating marker locations into the map data.

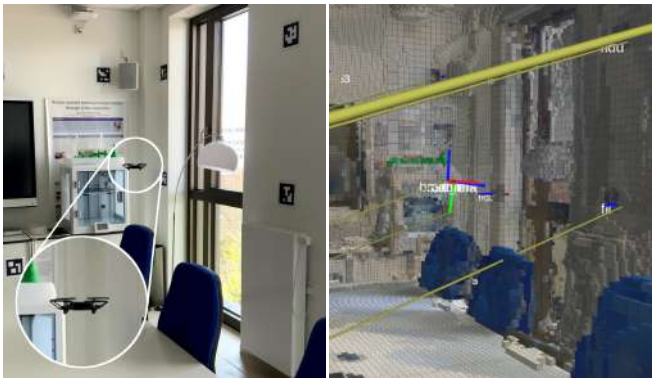


Fig. 1. Comparison of the real environment on the left and the generated 3D occupancy map including the localized fiducial markers and the localized MAV on the right. The left image shows the markers printed on paper and hung up on the walls as well as the MAV. In the right image, the markers can be seen as the darker colored voxels in the occupancy map. The MAV is shown as the coordinate frame in the center of the right image. The yellow rays show the relation between the map origin and frame positions.

In this paper, we propose to use a tablet equipped with a Lidar sensor to create a 3D occupancy map and localize fiducial markers using the same sensor data as for the map creation. A 3D occupancy map containing localized markers in comparison to the real environment is shown in Figure 1. The fiducial markers allow the MAV to estimate its position by detecting and localizing the pose of the markers in its own coordinate frame. Since the pose of a fiducial marker is known for both coordinate frames, the MAV's local frame and the occupancy map frame, the pose of the MAV in the occupancy map frame can be determined. The pose enables the MAV to use the occupancy map for path planning and navigation. In summary, the main contributions of this paper are

1) a pipeline for localizing fiducial markers while creating a 3D occupancy map using an assisting device and
2) a ROS 2 [2] package for autonomous navigation of a Ryze Tello EDU MAV using fiducial markers for localization and a voxel map for navigation planning.

The remaining of the paper is split into related work (Section II) followed by Section III in which the process of map creation and fiducial marker localization is described.

[1]Campus Minden, Bielefeld University of Applied Sciences and Arts, 32427 Minden, Germany firstname.lastname@hsbi.de

[2]University of Southern Denmark, 6400 Sønderborg, Denmark matthiask@iti.sdu.dk

Acknowledgements: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project number 465783762.

Section IV is about the MAV controller package for ROS 2. It describes how it uses the information, generated as described in the previous section, for localization and path planning. In Section V, the proposed work is evaluated, and a conclusion is given in Section VI.

II. Related work

In the following, we describe related work regarding robot localization and fiducial markers.

A. Robot Localization

One major problem in robotics is localization, which is often coupled with mapping to form the SLAM problem. But in many cases, a map has already been captured and a robot only needs to be localized in that map. A popular option is to use visual natural or artificial features. For example, FAPM-L [3] uses a feature-annotated polygon map to localize UAVs in an indoor environment. The map was designed to allow for localization. They extract keypoints from 2D images and find matching 3D landmarks in the polygon map that have been extracted from the map a priori. Similarly, Yu et al. [4] obtain coarse 2D-3D line correspondences between 2D images and Lidar maps based on visual-inertial odometry, which are refined in a second step. Wang et al. [5] employ visual features known as ORB [6] to relocalize a mobile robot to avoid drift in a 2D case. All three approaches use the information that is available in the environment, but might be susceptible to a low number of natural features.

A possible solution to this is to add artificial features like fiducial markers to the environment, like in [7] and [8]. Javierre et al. [7] use fiducial markers to assist in 2D localization combined with omnidirectional vision. Houben et al. [8] localize sparsely distributed fiducial markers while building a map using a laser scanner mounted on an MAV. The MAV is first localized through the laser scanner and can later be localized through marker detection with the advantage of much higher frequency compared to the laser scanner. Our approach differs in that we require an assisting device that is used for the map creation and marker localization process. The MAV only relies on the fiducial markers for localization in our case. Also, we go one step further and create a voxel-based occupancy map for navigation.

B. Fiducial markers

Because of the possible lack of natural features in the environment, fiducial markers are a popular method to add artificial features into the environment. Some of their biggest advantages compared to natural features is their uniqueness and the ease of detectability. Many fiducial marker implementations like ARTag [9], AprilTag [10], and ArUco [11] have a square border, in which a unique ID number is encoded. This ID number enables the distinct identification of the marker. The individual implementations typically differ in *false detection rate*, *number of unique IDs* and their *inter-marker distance*, *handling of occlusion*, and *localization accuracy* as well as their visual design. While the inner image of a marker encodes the ID number, a square border of a fiducial marker is used for 6 DOF localization. A marker commonly used in robotics is the ArUco marker [11]. It is a highly configurable marker with different code lengths, allowing either high marker count or high inter-marker distance. Furthermore, it includes error detection and correction. The pose is estimated using a Perspective-n-Point algorithm.

In contrast to square markers, there also exist circular markers like RUNE-Tag [12]. RUNE-Tag encodes its ID numbers using smaller circles inside rings of a larger circle. Circular fiducial markers have the downside, that they require multiple markers to be detected for localization. The RUNE-Tag marker has the disadvantage, that it can only be detected at lower distances. On the other hand, ist has a more accurate detection as well as better robustness against occlusion because it does not rely on few geometrical features like the corners of a square marker. STag [13] is a marker that combines the advantages of the square and circular approach by incorporating a circular encoding and a larger circular border as well as a square border into the marker design. Despite the advantages of STag, we use the ArUco marker in our approach, as this is the more commonly used marker.

III. Assisting device

Due to the lack of required sensors, an MAV might be unable to capture all the necessary information about the environment. To provide such data, an assisting device can be used to capture additional map data and make it available to the MAV through a localization layer. We chose an Apple iPad 2020 Pro as the assisting device, which features a Lidar sensor. It runs an application that captures depth images as well as corresponding RGB images and pose information. This data is made available to a ROS 2 node that is responsible for creating the 3D occupancy map and localizing fiducial markers.

A. Map creation

The map that is created is a 3D probability-based occupancy map. The map creation process is GPU-accelerated and requires a depth image with a corresponding RGB image, the intrinsic camera parameters, and the tablet's pose information. Based on the intrinsic camera parameters, the depth image is transformed to a point cloud with attached color information. The point cloud is inserted into a voxel grid with regard to the tablet's pose by updating the occupancy probability of each voxel. To use the voxel map with ROS 2 and the MAV controller, it is converted into an octomap [14].

B. Fiducial marker localization

The main goal of the fiducial marker localization is to make their global poses available to the MAV. The fiducial marker localization is done using a three-step procedure:

1) First, the fiducial markers are detected on the RGB images captured by the tablet. The detection is a standard method that returns the image coordinates of the four corner points of the ArUco marker and its ID.

2) Using the camera's intrinsic parameters and the image coordinates of the corner points, a Perspective-n-Point (PnP) pose computation is applied to get the 3D pose $T_{tablet \rightarrow fiducial\_X}$ of the detected marker relative to the camera.
3) Since the tablet's global pose $T_{map \rightarrow tablet}$ in the map is known, the global pose of the detected markers can be calculated using the formula

$$T_{map \rightarrow fiducial\_X} = T_{map \rightarrow tablet} \times T_{tablet \rightarrow fiducial\_X}. \quad (1)$$

While generating the 3D occupancy map, a second map containing all detected markers is created as well. Both maps share the same origin. For each marker, the pose and an object (localization) error is known. When the fiducial marker localization detects a new marker, it is directly inserted into the map. If the map already contains that marker, its pose is updated when the object error of the new detection is less than the current object error. The object error e_o is based on the reprojection error e_r, which is calculated using the formula

$$e_r = \frac{1}{4} \sum_{i=1}^{4} d(o_i, r_i) \quad (2)$$

where 4 refers to the number of outer corner points of the fiducial marker, and with $d(o_i, r_i)$ being the distance between the original corner point o_i and the reprojected corner point r_i based on the pose. The object error can be calculated using

$$e_o = \frac{e_r}{d} \times \frac{\|t\|}{L}, \quad (3)$$

where d is the distance between two diagonal corners of the fiducial marker, t is the translational part of T_{rel_marker} and L is the length of the marker. The poses and IDs of the fiducial markers are made available to the MAV controller.

IV. MAV CONTROLLER

The MAV controller is a ROS 2 node that can control a Ryze Tello EDU MAV. The Ryze Tello EDU is a small drone, weighing only 80 grams and allowing up to 13 minutes of continuous flight. It comes with a 5 MP front camera and can be controlled through Wi-Fi. An official SDK that can be accessed through UDP is provided by the manufacturer, which supports sending text-based commands and receiving status information and a video stream. The Ryze Tello EDU supports the version 2.0 of the SDK by default, but can be updated to support commands from the current version 3.0. The library that the developed ROS 2 node uses internally is CTello [15]. CTello is a small library written in C++, which allows direct access to the SDK commands.

The ROS 2 node is split into three parts, where one is responsible for publishing state information made available by the drone. The drone publishes its state in 10 Hz intervals. The state information contains IMU sensor data, battery percentage, and height among others. This information is made accessible through standard ROS topics, which implement the publish–subscribe pattern. The second part, the camera publisher, handles the video stream of the MAV. It updates the drone's camera settings like resolution and fps and starts the camera stream. A UDP socket is created to listen for incoming video data and publish it through ROS 2. Furthermore, the intrinsic camera parameters are published, which have been determined prior using camera calibration.

The third part of the ROS 2 node is the movement controller. It provides ROS topics to directly send SDK commands like *takeoff*, *land*, or *emergency* to the drone. Two additional features of the movement controller are auto cooling to prevent automatic drone shutdown and auto landing interrupt to prevent the drone from landing automatically when it does not receive a new command within ten seconds. Furthermore, it supports sending remote control commands through a */cmd_vel* topic for joystick control. An additional ROS 2 node allows for converting button presses into the other necessary commands that were mentioned prior. Another ROS topic */move_to* accepts global position commands and makes the drone move to that location based on its localized position. The third option for moving the drone is to use the move action. The move action is capable of handling more complex movement commands by using path planning.

A. MAV localization

To localize the drone in the map, we use odometry data provided by the drone and the fiducial markers detected in the map creation process. The odometry data allows for smooth, continuous pose determination which is described as the relation between the *odom* and *base_link* frames, while the fiducial markers ensure that the drone is correctly positioned globally. This is shown by the relation between the *map* and *odom* frames. The relation between *map* and *base_link* therefore is the pose of the drone relative to the map.

a) odom → base_link: The SDK of the Ryze Tello EDU makes available only an acceleration vector, a velocity vector and the height with an update rate of 10 Hz. The acceleration vector contains the unprocessed data of the accelerometer. The velocity vector and height are measured in dm/s and cm, with both having a resolution of ten centimeters. The height value is relative to the height at takeoff. We decided to use the velocity vector and the height value for determining the translation from *odom* to *base_link* because they are more accurate over a longer time period than the acceleration data. With p_t being the drone's position at time t and v_t being the velocity at that time, the position at time $t + 1$ is calculated by $p_{t+1} = p_t + v_t$.

Since an absolute value for the height of the drone is known, the calculated height value is replaced by the measurement. As the roll, pitch, and yaw values are provided by the gyroscope of the drone, they are published as the rotational part of the transformation.

b) map → odom: The *map → odom* transformation determines the global pose of the drone in the map and corrects the *odom → base_link* transformation if necessary. For calculating the transformation, fiducial markers are detected

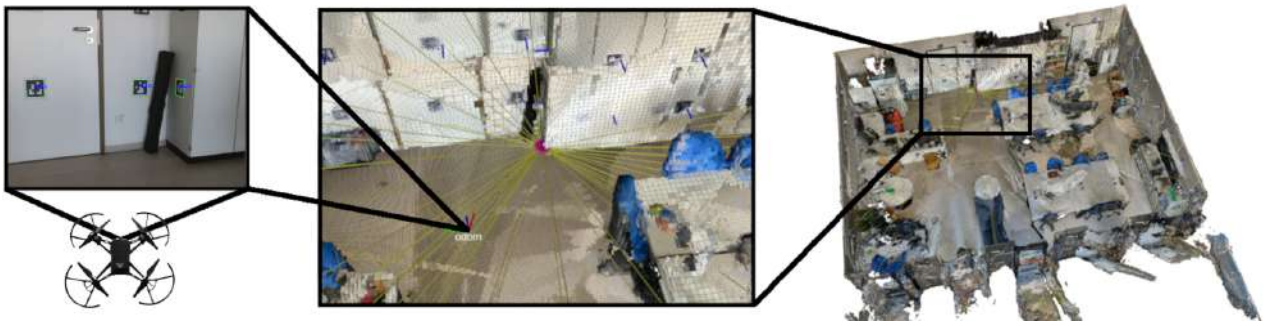


Fig. 2. The MAV captures RGB images, on which fiducial markers are localized. Since the global poses of the markers are known, the pose of the MAV can be inferred. The center image shows a section of the map, in which the MAV is positioned. Fiducial markers can be seen on the walls. The yellow rays in the image show the relation between individual markers and the map origin. On the right, an overview of the occupancy map with a size of 10 × 8 m is shown, which is also used for evaluation.

and localized like described in Section III-B except that only the relative pose to the drone's camera is determined.

Because the global pose of the fiducial marker as well as its pose relative to the drone camera is known, a loop is formed. Within the loop, the transformation between *map* and *odom* can be calculated using

$$T_{map \to odom} = T_{map \to fiducial\_X} \times T^{-1}_{odom \to fiducial\_X}, \tag{4}$$

where $T_{odom \to fiducial\_X}$ is calculated by

$$T_{odom \to fiducial\_X} = T_{odom \to base\_link} \times T_{base\_link \to camera} \times T_{camera \to fiducial\_X}. \tag{5}$$

Every T is a transformation matrix describing the transformation between the frames mentioned in the subscript.

We differentiate between two different methods to determine the final pose of the MAV, which will both be evaluated. If only a single marker is detected in the image, both methods lead to the same result. When multiple markers are detected, the first method uses the fiducial marker with the lowest object error for localization. The second method calculates a weighted average pose for all fiducial markers based on their object error. With both transformations *map→odom* and *odom→base_link* given, the global pose of the drone is known. This enables the use of the occupancy map for path planning.

B. Path planning and navigation

The navigation part of the MAV controller allows a user to define a goal position, to which the drone is able to fly autonomously. To this end, three steps need to be executed. The first step is to generate a costmap based on the occupancy map provided by the assisting device. In the second step, a path is planned based on the generated costmap. The third step is to move the drone based on the planned path.

a) Costmap generation: The costmap is required for global path planning for the drone. Its difference to the occupancy map is that it has a lower resolution and occupied areas are inflated. We decided to use a resolution of 0.2 m compared to 0.025 m of the occupancy map as this is the minimum distance the Ryze Tello EDU drone requires to execute the *go* command of the SDK. Also, it reduces the time required for path planning and generated the costmap. The inflation is done by calculating the distance of each voxel that is not occupied to the nearest occupied voxel. If the distance is less than 0.3 m, the current voxel is marked as occupied in the costmap. Inflating ensures that the drone does not move too close to obstacles and makes room for localization errors. If a voxel is already marked as occupied in the occupancy map, it is directly set as occupied in the costmap as well. Voxels with a larger distance to the nearest occupied voxel are marked as free. We do not differentiate between free and unknown voxels, as we assume that the occupancy map has been fully created and the goal position is placed in known space.

b) Path planning: For path planning, we use the A* algorithm, where the nodes are represented by the voxels of the costmap. The standard A* algorithm is improved in the following three ways:

1) When the drone is localized inside an occupied voxel, a tunnel is created to find the nearest free voxel for path planning. This is necessary for a drone that is landed because the ground is inflated as well.
2) The generated path contains many positions with a small distance between each other. To make the drone movement more fluent, only the last position of a straight line of unoccupied voxels are kept.
3) Since there is the possibility that no path can be found, the path planning uses a timeout. If the path planning takes too long, the current best path is returned even if it is not finished.

c) Navigation: The navigation is done by using the provided *move_to* topic that accepts a single goal position. It is required that the drone has already taken off. A flight state is published by the MAV controller that contains information about the flying and hovering state of the drone. After the first movement command is sent, the navigation part waits until the drone has entered the hovering state again. Then, it sends the next movement command using the next position of the path until the drone has reached its final destination.

V. Evaluation

In this section, the accuracy of the fiducial marker localization and the accuracy of the drone localization are evaluated. For both parts, the method is described first. Then, the results are presented and discussed.

A. Fiducial marker localization

For accurate drone localization, the fiducial markers need to be positioned accurately in the map. Therefore, we first validate the correct localization of the fiducial markers by the assisting device. The goal of this experiment is to show that the localization results in later experiments are valid for both the occupancy map and the fiducial markers.

a) Method: We compare the poses of the fiducial makers based on the localization using RGB images as described in Section III-B and their poses in the occupancy map. The pose in the occupancy map is used as ground-truth. Because the occupancy map we use is colored, the fiducial markers are visible in the occupancy map and their center and rotation can be determined. The resolution of the occupancy map is 0.025 m and the length of the markers is 0.135 m. Due to the voxel-based structure of the occupancy map, we restrict the placement of fiducial markers to only 90 degree angles in the real world. When capturing an occupancy map, we make sure that the assisting device is oriented correctly so that a fiducial marker has a flat surface in the occupancy map.

For evaluation, we generate 10 maps, with each containing 76 markers. For every map, we randomly select 10 out of the 76 markers for evaluation, resulting in a total of 100 evaluated markers. If the occupancy map visualization of a marker is not sufficient for ground truth pose estimation, we randomly select a new marker. For each marker, the translational and rotational error is determined. The translational error is calculated using the Euclidean distance between the marker position based on RGB image localization and the position of the center of the marker in the occupancy map. The rotational error is calculated using

$$d(q_1, q_2) = 1 - \langle q_1, q_2 \rangle^2, \tag{6}$$

where q_1 and q_2 are the orientations of the fiducial marker as quaternions.

b) Results: The mean translational error is 0.083 m with a standard deviation of 0.077 m. For the rotational error, the mean is 0.22° with a standard deviation of 0.62°. The median values are 0.064 m and 0.09°, respectively. These results show that it is possible to localize the fiducial markers with high accuracy. This enables the use of the occupancy map for navigation, while for localization only the markers are required.

B. Drone localization

For the final evaluation, we test the accuracy of the proposed drone localization. We compare both methods of drone localization using a single marker selected based on the object error and using the weighted average of all detected markers in an image.

a) Method: To evaluate the localization accuracy, we simulate a drone flight by manually positioning a drone 200 times in the map and determining its pose based on the fiducial marker localization using both methods and based on the localization of the assisting device, which we use as ground-truth data. We first create a map and localize the fiducial markers using the assisting device. Then, we position both the drone and the assisting device at the same random locations in the environment and determine their pose. We use a static offset for the pose of the assisting device, so that we can position both the device and the drone at the same time. We evaluate the localization of the drone by determining a translational and rotational error using the same formula as in the first experiment. Since we want to manually position the drone as realistically as possible, we first conduct an actual drone flight in the test environment to determine the distribution of the marker counts in the drone images. We then collect drone images that resemble the distribution as close as possible.

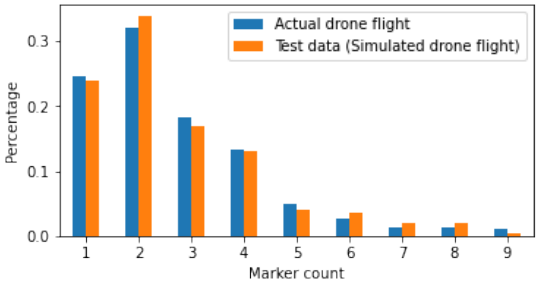


Fig. 3. Distribution of marker counts of the statically collected test data compared to an actual drone flight. Zero markers were detected in 24.2 % of all captured images of the drone flight.

b) Results and discussion: Test data has been captured with a similar marker count distribution as shown in 3. The mean distance between the drone and the markers is 2.18 m.

The results of the drone localization show that both methods have a similar overall accuracy for translation and rotation. The mean translational errors are 0.26 m±0.40 m and 0.28 m±0.42 m for single marker localization and weighted average localization, respectively. The rotational errors are 2.0°±3.8° and 2.4°±5.2°, respectively. The median error is lower for both methods, with 0.13 m and 1.1° for single marker localization and 0.12 m and 1.1° for weighted average localization. Figure 4 contains more detailed information about the results. Our results are slightly better compared to Houben et al. [8] who localized the markers while creating the map using a drone instead of an assisting device. They measured an accuracy of 0.50 m±0.85 m (median: 0.19 m) and 10°±15° (median: 4°) for images with a single marker taken at distances between 0.61 m and 4.99 m in their test environment.

We noticed that the accuracy in translation heavily depends on the distance between the drone and the detected markers. This is not the case with the rotation. Since a larger distance to the markers is required to capture multiple markers, the accuracy for images with multiple markers decreases for both

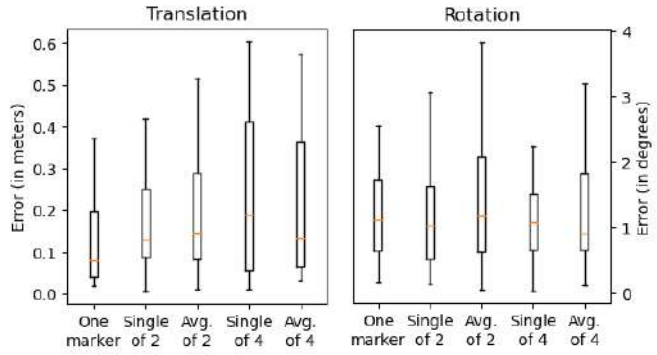


Fig. 4. Translational and rotational error of the drone localization for a single, two, and four markers. The figure shows the results for single marker localization (Single) and weighted average localization (Avg.). The results for one marker in an image is only shown once, since both methods share the same results.

methods. Additionally, we noticed that the marker detection algorithm started not detecting markers at a distance larger than 5 m. Since the markers are mostly placed on the walls of the room, this makes it impossible to localize the MAV when it is looking towards the center of the test environment.

When comparing the two methods, we observed that weighted average localization is more stable for subsequent images, as shown in Figure 5. Both share the same accuracy for lower distances towards markers, as this typically means that only one marker is visible. But on further distances, the single marker localization leads to unstable pose estimation. On the other hand, both methods can lead to jumps in position estimation when different markers are visible in subsequent images. Therefore, instability can also occur in weighted average localization.

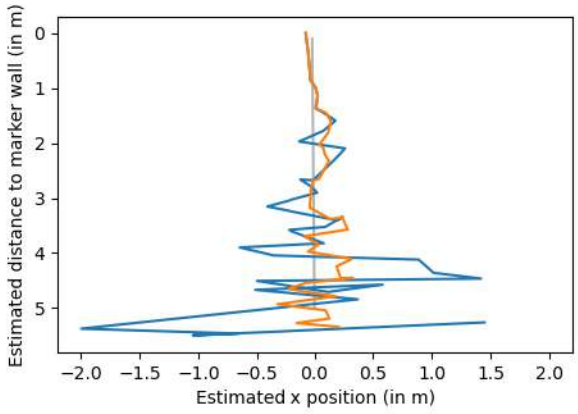


Fig. 5. Single marker localization (blue) and weighted average localization (orange) of an image sequence of the drone flying with constant speed towards (from y = 5 m to y = 0 m) a 3×3 marker matrix on a wall, with the markers placed 1 m apart horizontally and 1 m (upper two rows) and 0.6 m (lower two rows) vertically. The x- and y-axis show the estimated positions of the drone. The gray line is the ground truth.

VI. Conclusion and Future Work

In this paper, we presented a pipeline for assisted localization for MAVs by pregenerating a 3D occupancy map and detecting fiducial markers using a second device equipped with the necessary sensors. The occupancy map can be used by the MAV for navigation and path planning, while the fiducial markers enable the drone to localize itself in the map. When the drone detects a known marker, it can infer its pose based on the marker's pose. We demonstrated that the fiducial markers can be correctly localized by the assisting device while it builds the occupancy map. Also, we showed that a drone can then detect these markers and localize itself in the environment. But we noticed that the drone localization fails at larger distances between the drone and the markers. Further, we presented a ROS 2 based MAV controller for localization, path planning and navigation among other things.

Future work will focus on improvements in localizing drones using fiducial markers, with the goal to reduce the large variance in pose estimation using a Kalman filter and enable localization at larger distances. A second research path will concentrate on localizing a drone without the use of artificial features like markers.

References

[1] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1999, pp. 1322–1328.

[2] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, 2022.

[3] A. Mock, T. Wiemann, and J. Hertzberg, "Monocular localization in feature-annotated 3d polygon maps," in *European Conference on Mobile Robots (ECMR)*, 2021.

[4] H. Yu, W. Zhen, W. Yang, J. Zhang, and S. Scherer, "Monocular camera localization in prior lidar maps with 2d-3d line correspondences," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4588–4594.

[5] E. Wang, D. Chen, T. Fu, and L. Ma, "A robot relocalization method based on laser and visual features," in *IEEE Data Driven Control and Learning Systems Conference (DDCLS)*, 2022, pp. 519–524.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[7] P. Javierre, B. P. Alvarado, and P. de la Puente, "Particle filter localization using visual markers based omnidirectional vision and a laser sensor," in *IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 246–249.

[8] S. Houben, D. Droeschel, and S. Behnke, "Joint 3d laser and visual fiducial marker based slam for a micro aerial vehicle," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2016, pp. 609–614.

[9] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2005, pp. 590–596.

[10] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3400–3407.

[11] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, pp. 2280–2292, 2014.

[12] F. Bergamasco, A. Albarelli, L. Cosmo, E. Rodolà, and A. Torsello, "An accurate and robust artificial marker based on cyclic codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 12, pp. 2359–2373, 2016.

[13] B. Benligiray, C. Topal, and C. Akinlar, "Stag: A stable fiducial marker system," *Image and Vision Computing*, vol. 89, pp. 158–169, 2019.

[14] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[15] C. P. López, "Ctello," https://github.com/carlospzlz/ctello, 2020.

Stable Yaw Estimation of Boats from the Viewpoint of UAVs and USVs

Benjamin Kiefer[1], Timon Höfer[1] and Andreas Zell[1]

***Abstract*— Yaw estimation of boats from the viewpoint of unmanned aerial vehicles (UAVs) and unmanned surface vehicles (USVs) or boats is a crucial task in various applications such as 3D scene rendering, trajectory prediction, and navigation. However, the lack of literature on yaw estimation of objects from the viewpoint of UAVs has motivated us to address this domain. In this paper, we propose a method based on HyperPosePDF for predicting the orientation of boats in the 6D space. For that, we use existing datasets, such as PASCAL3D+ and our own datasets, SeaDronesSee-3D and BOArienT, which we annotated manually. We extend HyperPosePDF to work in video-based scenarios, such that it yields robust orientation predictions across time. Naively applying HyperPosePDF on video data yields single-point predictions, resulting in far-off predictions and often incorrect symmetric orientations due to unseen or visually different data. To alleviate this issue, we propose aggregating the probability distributions of pose predictions, resulting in significantly improved performance, as shown in our experimental evaluation. Our proposed method could significantly benefit downstream tasks in marine robotics.**

I. Introduction

Yaw estimation of objects from the viewpoint of unmanned aerial vehicles (UAVs) and unmanned surface vehicles (USVs) or boats is an essential task in various applications such as 3D scene rendering, trajectory prediction, and navigation. Accurate pose estimation is crucial for safe and efficient operations in the marine environment, where the ability to locate and track objects such as boats and ships is essential for collision avoidance, search and rescue, and marine surveillance. Furthermore, it is vital to have robust yaw predictions in augmented reality applications, to better aid a human operator.

AIS (automatic identification system) data only helps for boats that emit these signals. Smaller boats do not send AIS data. Furthermore, radar is expensive and only provides a very coarse position of boats. It requires a correct set-up of the radar and is harder to interpret for non-experts. Computer vision-based orientation prediction on the other hand offers a cheap and direct method.

Furthermore, there is a lack of literature on heading estimation of objects from the viewpoint of UAVs and USVs. In particular, predicting the orientation of objects far away from the camera is a challenging task due to the inherent uncertainty in the visual data. Methods based on 3D bounding box detection rely on precise box labels and are inherently error-prone for distant objects [1].

In this paper, we propose a method based on HyperPosePDF [2] for predicting the orientation of boats in the 6D space. HyperPosePDF is a recent method that models the uncertainty of predictions and has shown promising results in the field of 6D pose estimation. We train this method on existing datasets, such as PASCAL3D+, and on our own datasets, called SeaDronesSee-3D and BOArienT, which we manually annotate with bounding boxes and pose information for evaluation purposes.

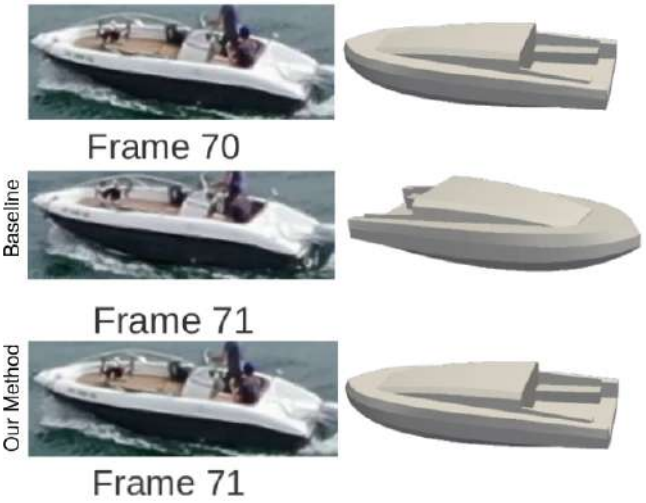


Fig. 1. Ignoring the temporal domain results in false, near-symmetric orientation prediction of a boat from frame 70 (top) to frame 71 (middle). Tracking the probability distributions alleviates this problem (bottom).

To speed up the bounding box annotation, we develop an annotation tool based on the recently published "Segment Anything" method [3]. We make this tool together with the data publicly available.

We extend HyperPosePDF to work in video-based scenarios, where the prediction of the orientation of objects across time is essential. Naively applying HyperPosePDF on video data yields single-point predictions, often resulting in far-off predictions and incorrect symmetric orientations due to unseen or visually different data. Therefore, we propose aggregating the probability distributions of pose predictions over time, resulting in significantly improved performance, as shown in our experimental evaluation.

Furthermore, naively predicting the yaw of boats based on analyzing their trajectory in 3D space does not work for standing or slowly moving boats. Moreover, formulating yaw prediction in this way is error-prone due to an ill-posed 2D $\longleftrightarrow$ 3D projection, which is not reliable in heading estimation as we will see in subsequent sections.

Lastly, we demonstrate a full pipeline with detection and tracking of objects and subsequent orientation prediction for a downstream synthetic rendering of a scene. Our proposed method could significantly benefit downstream tasks in ma-

[1]All authors are with the Faculty of Computer Science, University of Tuebingen, Germany. prename.surname@uni-tuebingen.de
979-8-3503-0704-7/23/$31.00

rine robotics.

In summary, our contributions are as follows:

- We pose the novel problem of predicting the heading of boats via purely vision-based methods.
- We propose a novel method to aggregate pose predictions by tracking the probability distributions to capture uncertainties due to symmetries and ambiguous appearances.
- We create a new dataset BOArienT, a benchmark featuring 30 FPS manually annotated video, featuring precise object detection and pose labels. Furthermore, we annotate parts of SeaDronesSee-MOT with pose data, which we call SeaDronesSee-3D.
- We show in multiple experiments on diverse benchmarks the utility of our method. Lastly, we demonstrate the utility of our method on a full pipeline with detection and tracking to synthetically render a scene.
- We make code, data, and adapted labeling tools publicly available on `www.macvi.org`.

II. Related Work

Pose estimation of common or close industrial objects has been explored in several methods [4]–[6]. Analyzing the static images, they split the task into two stages - object detection and subsequent 6D pose estimation of the predicted bounding boxes. However, their focus is on close objects that are dominant in the image plane. On the other hand, we focus on yaw estimation of boats that are distant and occasionally hardly visible. This makes an accurate yaw estimation hard as many plausible predictions exist. Several works explored how to model the uncertainty of pose predictions [2], [7]. They output probability distributions over many different poses, effectively capturing the symmetries inherent in the poorly visible objects. While they only experiment with common objects in static scenes, we aim to build on top of their methods to predict stable yaw predictions across time.

The last years have shown a great influx in works in maritime computer vision [8]–[11]. Most works focus on the detection or tracking of objects from the viewpoint of UAVs, USVs or boats. There is a great corpus of works working on simulation and trajectory prediction [12], [13]. However, these methods only operate on map data as opposed to image/video data.

Likewise, the general UAV-/USV-based research focuses on object detection and tracking, and anomaly detection [9], [14]–[18], but neglects the yaw estimation aspect.

III. 3D Geometry Prerequisites

There are three principal axes in any boat, called longitudinal, transverse and vertical axes. Figure 2 shows the rotations around these. These are absolute orientations, i.e. while our method outputs an orientation estimation, it is relative to our camera view. Therefore, we may obtain the absolute orientation using an onboard magnetometer or dual GNSS solutions.

We note that we focus on the case of zero roll and pitch angle, i.e. only the orientation is predicted.

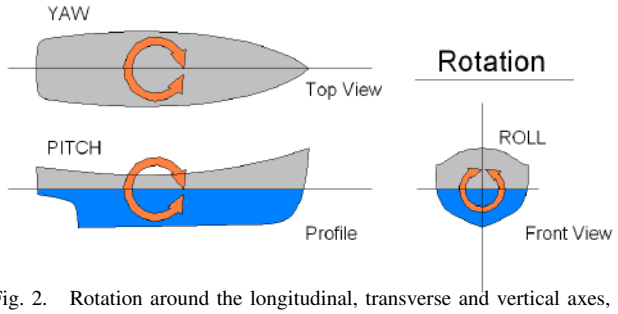


Fig. 2. Rotation around the longitudinal, transverse and vertical axes, i.e. roll, pitch, and yaw [19].

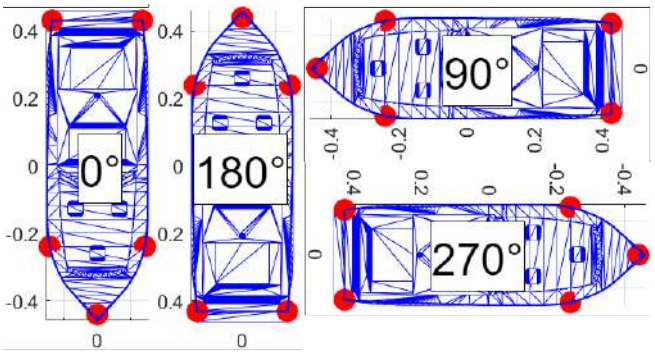


Fig. 3. Orientation relative to the camera. At 0°, the boat's nose is facing directly us. Note that we did not include the roll and pitch angles.

For downstream tasks, such as trajectory prediction for collision avoidance but also for rendering synthetic scenes visually smooth and stable, we need to map our predictions to 3D space. For that, we compute 3D object coordinates relative to the UAV, and then use these to obtain actual world coordinates via passive geolocation.

For the relative object coordinates, we consider a mathematical perspective projection camera model since this resembles the common use case for cameras on UAVs and USVs. We assume our camera to look down at a certain angle, which may be a variable gimbal or static camera. A gimbal balances a potential UAV roll angle so that we assume there to be a zero camera roll angle. If there is no gimbal in the USV case, we apply a CV-based roll correction by levelling the horizon line using the IMU roll angle.

Using the relative coordinates of an object (x- and y-ground distances to UAV), we compute its GPS coordinates based on the UAV's GPS coordinates as follows. Given the camera heading angle θ, we compute the rotation matrix and rotate the relative coordinates of an object to obtain

$$\begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{1}$$

Finally, we map the relative coordinates to GPS coordinates via

$$la^{object} = la + \frac{y_r}{r}\frac{180}{\pi}, \tag{2}$$

$$lo^{object} = lo + \frac{x_r}{r}\frac{180}{\pi}\frac{1}{\cos(lat\ \pi/180)}. \tag{3}$$

Fig. 4. Left: Example orientation of a boat taken from 50m of altitude and looking down with a pitch angle of 40°. The highlighted boat has a yaw angle of 280° relative to our viewpoint. Since the UAV's heading is 170° (close to true south), we know that the boat has an absolute heading of 260° (close to true west). Right: Cad overlays on a frame of one the videos we took. Note the very small objects in the left part of the frame.

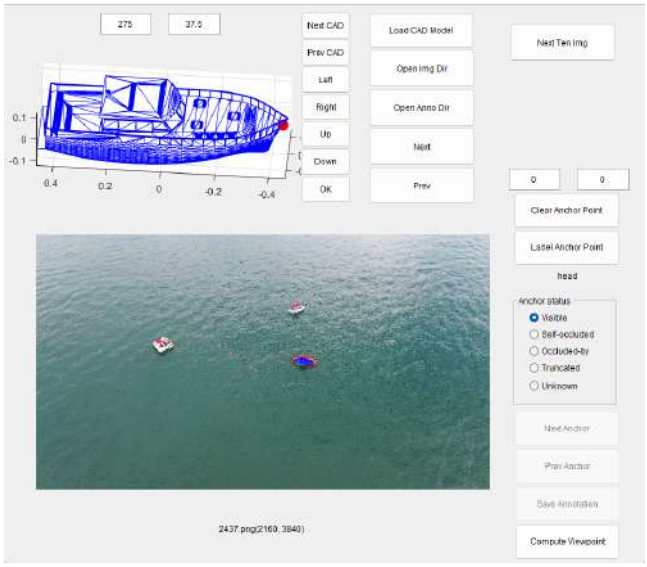


Fig. 5. Example view of pose labeling tool. First, we align the view coarsly in steps of 5°, then we put the visible anchor points (see Figure 3) in the image plane. These are used to obtain a better pose label. We optimize the orientation to match these anchors (see [7]).

We refer the reader to [20] for a more comprehensive derivation of the 2D $\longleftrightarrow$ 3D projection. Concretely, we would like to note that the projection may especially be critical for a distant object in the USV scenario as here, we encounter a very acute viewing angle. Small errors in pixel space result in large distance errors in world space. It is an open problem of how to correctly project distant objects in world space. For our consideration, we are mostly concerned with obtaining correct heading estimations for either close detections that may ultimately pose an immediate threat. For distant objects, we mostly care about stable heading predictions over time.

IV. Data Collection and Labeling

Because of the lack of available datasets for yaw estimation, we capture and annotate our own. For the UAV scenario, we leverage the already existing SeaDronesSee-MOT [8] dataset, which comes with bounding boxes and instance ids for boats. Furthermore, we annotate the 6D pose of boats from various sample scenes by adapting the annotation tool provided in [21]. Figure 5 shows an example scene where a boat is labelled from a viewpoint of a UAV. We leverage the provided metadata from the UAV to automatically infer coarse pitch and roll angles relative to the camera. Herein, we assume the world pitch and roll angle of boats to be zero, such that we only need to annotate the heading direction. For that, we manually provide a coarse heading and, upon selecting anchor points from the CAD in the corresponding real objects, we optimize for the precise 6D pose using their optimization procedure [21]. For annotation efficiency, we only annotate every 10th frame and interpolate the pose in between.

For the USV scenario, we capture our own data from the viewpoint of a fixed camera installed on a small motorboat. We use the ZED2 camera[1] with integrated IMU to infer the orientation at which we look at the scene. As before, we may also infer a coarse estimation of the roll and pitch angle for subsequent finer annotating via pose optimization. Before that, we annotate the scenes with bounding boxes using our tool, which we built on top of SAM (Segment Anything Model [3], see Fig. 6). We leveraged their largest ViT-H (636M parameters) model and built a user interface and labeling logic around it, such that objects can be assigned their bounding boxes by just clicking on them. Analogous to before, we annotate every 10th frame and interpolate in between. Table I shows a timing comparison between conventional labeling tools and our method. Every method was required to yield precise bounding boxes as rated by human experts. We repeated this experiment with five experts knowledgeable in the field of object detection. Each experiment lasted for half an hour. Our method clearly outperforms the others by 8.7 FPM. We hypothesize that fatigue symptoms occur later because annotating with a single click already covers the entire object. In contrast, when setting bounding boxes, precise outlining of the object is required, which becomes more exhausting over time. While this effort can be reduced by tracking, there are often errors

[1] https://www.stereolabs.com/zed-2/

TABLE I
ANNOTATION SPEED GIVEN IN FRAMES PER MINUTE USING OUR ANNOTATION TOOL BASED ON SAM AND VIT-H.

Annotation method	Labeling Speed (FPM)
DarkLabel [22]	3.8
DarkLabel + Interpolation	19.6
DarkLabel + Tracking	20.2
SAM-based	5.5
SAM-based + Interpolation	**28.9**

Fig. 6. Faster bounding box annotations by means of "Segment anything" [3]. We leverage this method to accelerate 2D bounding box labeling. A user just needs to click on the object, the corresponding bounding box will be set and saved automatically.

in tracking objects that are far away, requiring the annotator to stay alert and relabel bounding boxes.

We want to note that this method can fail in scenarios of low contrast or very distant objects. In this case, one has to resort to standard bounding box detection. Moreover, it requires a GPU to process ViT-H (in our case an RTX 2080Ti). Furthermore, a more exhaustive study on the benefits of segmentation-based labeling needs to be done to obtain a more comprehensive overview. In particular, a more comprehensive experiment considering object number, size, shape and movement needs to be done. We release both (adapted) annotation tools for further studies.

V. METHOD: AGGREGATING PROBABILITY DISTRIBUTIONS OVER TIME

Our approach is based on HyperPosePDF [7]. For an input image $x \in \mathcal{X}$, it aims to obtain a conditional probability distribution $p(\cdot|x)\colon \mathbf{SO}(3) \mapsto \mathbb{R}^+$, representing the distribution of the inherited pose of an object in the image x. For that, we train a vision backbone network, e.g. ResNet to predict the networks of a second network. While the vision network acts as a hypernetwork, the architecture of the second network is inspired by an implicit neural representation. The implicit neural representation acts on the rotation manifold and outputs for each pose, the corresponding probability of it being the underlying rotation of the object present in the image. Hence, it acts as an approximation of the probability distribution $p(R|x)$ by marginalizing over $\mathbf{SO}(3)$. During training, we maximize $p(R|x)$ by providing pairs of inputs x and corresponding ground truth R. To make a single pose prediction, we solve $\arg\max_{R\in\mathbf{SO}(3)} f(x, R)$ with gradient ascent, projecting the values back into the manifold after each step. To predict a full probability distribution, we evaluate $p(R_i|x)$ over the $\mathbf{SO}(3)$ equivolumetric partition R_i. This model can estimate complex distributions on the manifold without prior knowledge of each object's symmetries, and appropriate patterns expressing symmetries and uncertainty emerge naturally in the model's outputs. This is indeed, the most general way to conduct pose estimation. Specifically, in our scenario where we want to predict the pose to predict the trajectory it is possible to include uncertainty information of the pose to improve the performance.

The posterior distribution

$$P(R_{k+1}|Z_k), \tag{4}$$

based on the observations Z_k for time steps $\{1, \ldots, k\}$ can be approximated by

$$P(R_{k+1}|Z_k) \approx P(R_k|Z_k) + \Delta_{\text{pose}}, \tag{5}$$

where Δ_{pose} is defined as a weighted running average

$$\Delta_{\text{pose}} = \frac{1}{k}\sum_{l=0}^{k-1} \omega_l \Big(P(R_{l+1}|Z_{l+1}) - P(R_l|Z_l) \Big). \tag{6}$$

For $l < k+1$ the probabilities $P(R_l|Z_l)$ are known and approximated by the HyperPosePDF network. Therefore, the calculation of the pose at a future time point is deterministic. The weights ω_l fulfill $\sum_l \omega_l = 1$. To reduce the effect of earlier pose transitions, which have a lesser effect on the current pose movement it is plausible to simply set the initial weights as 0 and average the remaining over a smaller time interval chosen such that $0 < t < k$

$$\omega_l = \begin{cases} 0 & \text{for } l < t, \\ \frac{1}{k-t} & \text{for } l \geq t. \end{cases} \tag{7}$$

This especially comes in helpful, when we try to predict the movement of a boat that is in the middle of a turn manoeuvre and the respective trajectory resembles a curve. Furthermore, this allows us to detect false pose predictions in the case that the pose prediction in the next time step differs to much from the previous path. E.g., in the case of nearly symmetric boats, we experienced the appearance of $180°$ miss-predictions, which now can be easily excluded.

VI. EXPERIMENTS

First, we conduct experiments on the single-image Pascal3D+ set to illustrate the performance and expressiveness of HyperPosePDF. Similar to [2], we choose a pretrained ResNet-101 backbone for our vision module. Then, we train the model to predict the weights of a one-layer network with a width of 256. Using the Adam optimizer, we evaluate our model after 150k iterations using a batch size of 16. A learning rate of 10^{-5} is used for the first 1000 iterations, and then a cosine decay is applied. We choose a time horizon window of $k = 3$ for our experiments.

We report the performance of the category *boat* using the two commonly used metrics *accuracy at 30° (Acc)* and *median error in degrees (ME)*. Table II shows that this method is on par with SOTA methods (ImplicitPDF [7]).

Naively applying HyperPosePDF on video data yields single-point predictions (i.e. orientations) at each time step.

Fig. 7. Sample boat heading probability distribution predictions (given in radian). Ground truth values are 4.8, 0.4, 6.0, 4.0. Almost all the captured distributions are uni-modal and the best single-point estimator would yield a fairly close prediction of the orientation. The first image already provides a glance at the benefits of capturing the uncertainty. It is not clear whether the sailboat is sailing at an angle of 270° or slightly less.

TABLE II

YAW ESTIMATION RESULTS ON PASCAL3D+, SEADRONESSEE3D AND BOARIENT. NOTE THAT PASCAL3D+ ONLY FEATURES STILL IMAGES.

Method	Dataset	Acc↑	ME↓
ImplicitPDF	PASCAL3D+	56.0	**23.4**
HyperPosePDF	PASCAL3D+	**56.2**	22.8
HyperPosePDF	SeaDronesSee3D	65.6	20.1
+Run. Mean	SeaDronesSee3D	**71.9**	**16.7**
HyperPosePDF	BOArienT	42.5	41.8
+Run. Mean	BOArienT	**50.2**	**18.3**

However, uncertainty due to unseen data yields far-off predictions, often resulting in wrong symmetric orientations. For example, compare to Figure 1.

We evaluate on SeaDronesSee3D and BOArienT, where we manually annotated the orientations. Table II shows that our method yields higher accuracy at a maximum of 30° error tolerance as well as lower median angle error. Figure 1 shows an example sequence of SeaDronesSee3D where the single-image predictor miss-predicts the orientation by 180° due to the slight symmetric shape of the boat.

To test our approach in a complete pipeline, we employ a state-of-the-art multi-object tracker and apply the yaw estimator on the predicted bounding boxes. For the UAV scenario, we train on SeaDronesSee-MOT, and for the boat scenario, we take a pre-trained tracker on COCO. We report the performance of the trackers on SeaDronesSee3D and BOArienT in Table III.

Now, we apply the yaw estimator on top of the predicted bounding boxes with associated ids. Whenever a new tracklet is starting, we initialize a new probability distribution running mean. We only measure the orientation estimation performance on objects that have successfully been detected.

Table IV shows that our method still outperforms the single-image approach since the multi-object tracker is quite

TABLE III

MULTI-OBJECT TRACKING ACCURACY ON SEADRONESSEE3D (SDS3D) AND BOARIENT (BT). FOR SDS3D, WE USED THE METHODS FROM THE WORKSHOP COMPETITION [11]. ADDITIONALLY, WE BUILT ON TOP DEEPSORT A MEMORY MAP (MM) [20] TO BECOME MORE ROBUST TOWARDS ID SWITCHES AND FRAGMENTATIONS. FOR BT, WE USED OFF-THE-SHELF TRACKTOR & DEEPSORT.

	Model	HOTA↑	MOTA↑	IDs↓	Frag↓
SDS3D	ByteTracker	79.9	89.8	23	678
	DeepSORT	80.8	91.0	20	642
	+MM	**82.6**	**91.9**	**19**	**635**
BT	Tracktor	65.6	67.0	69	876
	DeepSORT	66.2	80.0	51	801

robust already (only a few ID switches degrade our method to effectively become a single-image method at these time points). Remarkably, we can even improve the point prediction over the naive mode running mean method, which simply applies a running mean on the modes of the distributions. We note, that this is on top of the higher expressiveness coming from our probability distributions: we may incorporate the uncertainty of heading estimations in downstream tasks, such as trajectory prediction, collision avoidance or for visualization purposes in augmented reality applications.

Finally, we compare our heading estimation approach with a naive trajectory-based approach. For every detection in every frame, we map its center box location to 3D via a perspective projection camera model [20] and capture the trajectory in world coordinates. We predict the next trajectory point by a constant velocity model coming from the previous three time steps. We take the resulting heading to be the final prediction of this baseline. If an object is lost, we need to re-initialize the heading which is a critical shortcoming of this approach. Furthermore, Table IV shows that the trajectory-based method fails on both scenarios due to stationary boats

TABLE IV

YAW ESTIMATION RESULTS ON SEADRONESSEE3D AND BOARIENT.

	Method+Tracking	Acc↑	ME↓
SDS3D	Trajectory-based	23.1	123.3
	HyperPosePDF	63.2	22.1
	+Mode Running Mean	64.3	21.6
	+Distribution Running Mean	**70.3**	**17.3**
BOArienT	Trajectory-based	20.2	72.6
	HyperPosePDF	39.6	43.9
	+Mode Running Mean	40.1	42.0
	+Distribution Running Mean	**49.8**	**19.5**

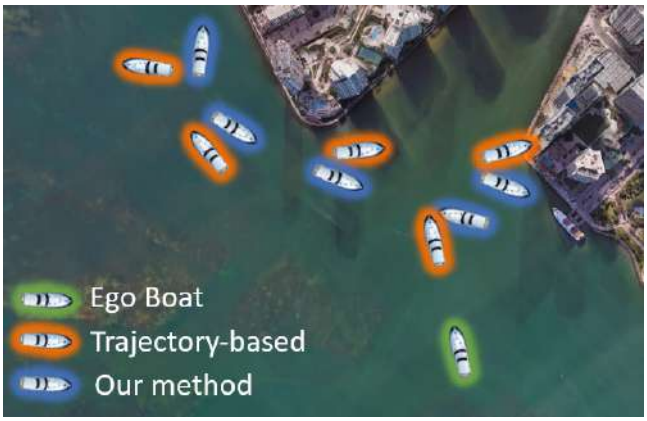


Fig. 8. Sample synthetic rendering of the scene from Figure 4. Detected boats and their heading are put into Google Earth. The big yacht on the right was already contained in the Google Earth image. We add a slight offset to the predicted locations (which are the same for the two methods) for visualization purposes.

and a challenging and noisy 2D $\rightarrow$ 3D projection.

Figure 8 shows the predicted positions and headings in BOArienT coming from our method and from this baseline via 2D $\rightarrow$ 3D projection. Because some boats are stationary, the heading information for the baseline is incorrect. Furthermore, the heading information from slowly driving boats is very noisy as the underlying 2D $\longleftrightarrow$ 3D projection is error-prone. Single-image predictions are better, but the smallness of the objects makes these predictions also very noisy.

VII. CONCLUSION AND DISCUSSION

In this paper, we addressed the novel problem of predicting the yaw of boats from the viewpoint of unmanned aerial vehicles (UAVs) and unmanned surface vehicles (USVs) or boats. We proposed a method based on HyperPosePDF, which models the uncertainty of predictions and yields robust orientation predictions across time in video-based scenarios. To demonstrate the utility of our method, we created two new datasets, SeaDronesSee-3D and BOArienT, manually annotated with bounding boxes and pose information, and made them publicly available. Our experimental evaluation showed that our method significantly improves performance compared to naive single-point predictions. Our proposed method has potential applications in marine robotics, including 3D scene rendering, trajectory prediction, and navigation.

REFERENCES

[1] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3d object detection methods for autonomous driving applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3782–3795, 2019.

[2] T. Höfer, B. Kiefer, M. Messmer, and A. Zell, "Hyperposepdf - hypernetworks predicting the probability distribution on so(3)," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2023, pp. 2369–2379.

[3] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.

[4] T. Höfer, F. Shamsafar, N. Benbarka, and A. Zell, "Object detection and autoencoder-based 6d pose estimation for highly cluttered bin picking," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 704–708.

[5] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*. Springer, 2020, pp. 574–591.

[6] M. Sundermeyer, Z.-C. Marton, M. Durner, and R. Triebel, "Augmented autoencoders: Implicit 3d orientation learning for 6d object detection," *International Journal of Computer Vision*, vol. 128, pp. 714–729, 2020.

[7] K. Murphy, C. Esteves, V. Jampani, S. Ramalingam, and A. Makadia, "Implicit-pdf: Non-parametric representation of probability distributions on the rotation manifold," *arXiv arXiv:2106.05965*, 2021.

[8] L. A. Varga, B. Kiefer, M. Messmer, and A. Zell, "Seadronessee: A maritime benchmark for detecting humans in open water," *arXiv preprint arXiv:2105.01922*, 2021.

[9] B. Kiefer and A. Zell, "Fast region of interest proposals on maritime UAVs," *arXiv preprint arXiv:2301.11650*, 2023.

[10] B. Bovcon, J. Muhovič, D. Vranac, D. Mozetič, J. Perš, and M. Kristan, "MODS–A USV-Oriented Object Detection and Obstacle Segmentation Benchmark," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, 2021.

[11] B. Kiefer, M. Kristan, J. Perš, L. Žust, F. Poiesi, F. Andrade, A. Bernardino, M. Dawkins, J. Raitoharju, Y. Quan, *et al.*, "1st workshop on maritime computer vision (macvi) 2023: Challenge results," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 265–302.

[12] B. Sullivan, C. Ware, and M. Plumlee, "Predictive displays for survey vessels," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 50, no. 22. Sage Publications Sage CA: Los Angeles, CA, 2006, pp. 2424–2428.

[13] A. W. Browning, "A mathematical model to simulate small boat behaviour," *Simulation*, vol. 56, no. 5, pp. 329–336, 1991.

[14] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian, "The unmanned aerial vehicle benchmark: Object detection and tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 370–386.

[15] P. Ruiz-Ponce, D. Ortiz-Perez, J. Garcia-Rodriguez, and B. Kiefer, "Poseidon: A data augmentation tool for small object detection datasets in maritime environments," *Sensors*, vol. 23, no. 7, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/7/3691

[16] B. Kiefer, D. Ott, and A. Zell, "Leveraging synthetic data in object detection on unmanned aerial vehicles," *arXiv preprint arXiv:2112.12252*, 2021.

[17] M. Messmer, B. Kiefer, and A. Zell, "Gaining scale invariance in uav bird's eye view object detection by adaptive resizing," *arXiv preprint arXiv:2101.12694*, 2021.

[18] B. Kiefer, D. Ott, and A. Zell, "Leveraging synthetic data in object detection on unmanned aerial vehicles," in *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, 2022, pp. 3564–3571.

[19] Jmvolc. Rotations around axes. [Online]. Available: https://en.wikipedia.org/wiki/Ship_motions#/media/File:Rotations.png

[20] B. Kiefer, Y. Quan, and A. Zell, "Memory maps for video object detection and tracking on uavs," *arXiv preprint arXiv:2303.03508*, 2023.

[21] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3d object detection in the wild," in *IEEE winter conference on applications of computer vision*. IEEE, 2014, pp. 75–82.

[22] U. darkpgmr, "DarkLabel Annotation Tool, Github," https://github.com/darkpgmr/DarkLabel, accessed: 2022-07-05.

Graph-based Simultaneous Localization and Mapping with incorporated dynamic object motion

Peter Aerts[1*], Peter Slaets[2] and Eric Demeester[1]

***Abstract*— Over the last years, Simultaneous Localization and Mapping (SLAM) in dynamic environments has received more attention. This paper presents a SLAM algorithm in which dynamic object information is included within the graph-based optimization approach. By exploiting knowledge about the object's motion within the scene, the constructed map is a more accurate representation of the environment. Using data from simulation, we show that the robot's trajectory and the dynamic object's trajectory better aligns with respect to the ground truth. Real-world experiments, which includes human motion within the optimization, show that the robot's trajectory and thus the environment map is improved. This is verified based on the comparison between the constructed maps with and without the incorporation of the human motion. The validity of the map is obtained by evaluating three metrics from literature and a comparison to the building plans of the environment.**

I. Introduction

Simultaneous localization and Mapping, also known as SLAM, refers to a robot determining its location and surroundings within an unknown environment based on information gained from proprioceptive (i.e. wheel encoders, IMU, etc.) and exteroceptive sensors (i.e. LiDAR, camera, etc.).

In recent years, techniques have arisen attempting to integrate the information of dynamic objects within the SLAM optimization problem to simultaneously track and update the map of the environment.

In 2019, Simas et al. [1] presented a SLAMMOT (Simultaneous localization and Mapping and Mobile Object Tracking) approach based on an Extended Kalman Filter (EKF) in uncertain dynamic environments. They incorporate the EKF together with Multiple Hypothesis Tracking (MHT) to identify the motion model of each object. Zhang et al. [2] propose MOTSLAM. It uses sequential monocular frames and extracts objects using 2D3D object detection and semantic segmentation. The observations are split into foreground and background features, determining the transformation of objects based on fixed map points and using bundle adjustment to find the camera poses as well as the static and dynamic poses. Huang et al. [3] present a stereo visual odometry to simultaneously cluster and estimate the motion of the camera and surrounding objects. The ego motion and dynamic object poses are estimated through a sliding-window optimization. Zhang et al. [4] exploits semantic information to estimate the motion and tracking of dynamic objects while building the map. They implement bundle adjustment to estimate the motion of the dynamic object within the map giving good results, but this is computationally complex to run in real time. In 2021, Bescos et al. [5] present DynaSLAMII. Here, bundle adjustment is used together with the assumption of constant motion model of the camera and dynamic objects to improve the ego motion estimation together with those of the dynamic objects. We previously published [6], which describes a graph-based optimization technique to incorporate dynamic objects within the graph structure assuming a constant velocity model of the object. The estimation of the motion model parameters was executed using an Unscented Kalman Filter. It was shown that robot poses can be optimized using graph-based optimization without using static objects in the graph and solely using the pose measurement information of dynamic objects when a constant motion model of the object is estimated. However, the implementation of static features within the experiments was considered future work. Within this paper, we incorporate static features as well and simplify the determination of the parameters of the dynamic object.

The paper is structured as follows: Section II describes the construction of the graph for the optimization as well as the determination of static landmarks and human poses. Section III shows the experiments and results from simulation and real world experiments. Section IV concludes this paper with a short summary.

II. Graph construction

The proposed 2D graph-based method builds upon the pose-landmark-graph optimization algorithm. The basis of this approach is well described by Grisetti et al. in [7].

Figure 1 shows a visual representation of the construction of the graph to be optimized. The gray triangles represent the robot while the white stars represent the static landmarks. In

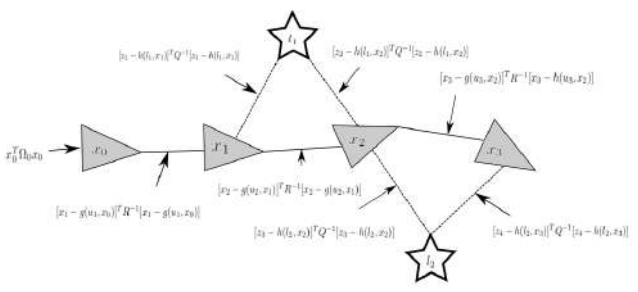


Fig. 1: Visualisation of general graph-based SLAM approach.

[1]Peter Aerts and Eric Demeester are with Faculty of Mechanical Engineering research group ACRO, University of Leuven, Belgium.

*Peter Aerts is affiliated to Flanders Make@KU Leuven

[2]Peter Slaets is with Faculty of Mechanical Engineering research group IMP, University of Leuven, Belgium.

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

this figure, the robot states are given by $x_t = [x, y, \theta]^T$ where x and y describe the robot's position and θ its orientation. These states are referenced to a fixed reference frame, generally the first robot state $x_0 = [0, 0, 0]$. These poses are consecutively linked via constraints. The constraints are derived from the odometry of the robot u_t. All landmark positions are represented by $l_i = [x, y]^T$. The visible landmarks are constrained to one or more robot poses derived from the observation z_t. The robot poses and landmark positions of which the optimal state is to be determined are called *vertices* or *nodes*. All constraints derived from odometry and observations are referenced as *edges*. The goal of graph-based SLAM is to find the optimal configuration of nodes which represent the robot poses and landmark locations. This can be obtained by minimizing equation (1). Here e_{ij} represents the error function regarding the expected measurement and actual measurement. Ω_{ij} is the information matrix and accounts for the uncertainty. We seek to find the optimal state for all nodes X^* which is obtained by minimizing $J(X)$ as described in (2).

$$J(X) = \sum e_{ij}^T \Omega_{ij} e_{ij} \tag{1}$$

$$X^* = \arg\min_X J(X) \tag{2}$$

Equation (3) represents equation (1) where each summation is the constraint pertaining to a certain type of node (robot pose or landmark). These constraints are also depicted in figure 1. Here R^{-1} and Q^{-1} are the information matrices of their respective terms. The term $x_0^T \Omega_0 x_0$ ensures that the first pose of the robot is fixed. This creates an anchor position for the optimization and is considered the starting point of the path and subsequently the map.

$$\begin{aligned} J(X) =& x_0^T \Omega_0 x_0 + \sum_t e_x(t)^T R^{-1} e_x(t) + \\ & \sum_t e_{l,x}(t)^T Q^{-1} e_{l,x}(t) \end{aligned} \tag{3}$$

The error function between consecutive poses is given by equation (4) where the motion model $g(\cdot)$ describes the robot's next pose x_t given the previous pose x_{t-1} and some form of odometry measurement u_t:

$$e_x(t) = x_t - g(u_t, x_{t-1}) \tag{4}$$

For the measurements of static objects, the error function is given by equation (5) where $k(\cdot)$ represents the expected measurement given the robot pose x_t and landmark position l_t with z_t being the actual sensor measurement:

$$e_{l,x}(t) = z_t - k(l_t, x_t) \tag{5}$$

Minimizing equation (3) provides the optimal configuration of nodes X^*. This is the standard graph-based SLAM approach using robot poses and landmarks.

We extend on this 2D pose-landmark-graph optimization with the implementation of dynamic objects within the graph-based approach. Figure 2 shows the connected graph where static objects as well as dynamic objects are registered. For the implementation of dynamic objects, the following statements are assumed:

- The association of dynamic objects is known in consecutive data points;
- The motion models of the dynamic objects within the scene resemble a constant velocity model.

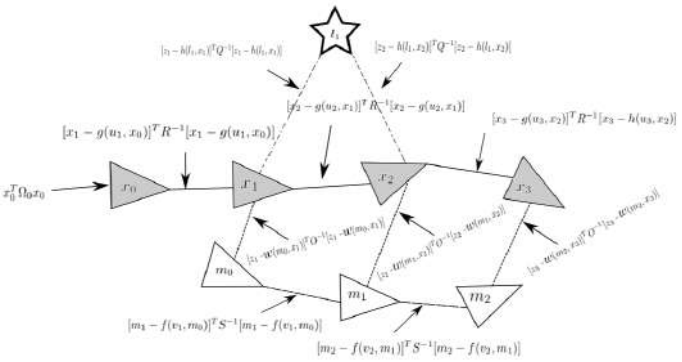


Fig. 2: Visualisation of the dynamic SLAM approach. Here, dynamic objects as well as static objects are included in the optimization algorithm. From every robot pose a constraint is made either to the dynamic objects and/or the static features within the scene. An additional constraint is created between consecutive object poses based on their velocity model estimation.

Dynamic objects are represented as $m_t = [x, y, \theta, v, w]^T$. For pedestrian motion, a constant velocity model is considered. Schöller et al. [8] show that a constant velocity model for pedestrian motion prediction yields good results compared to more complex models. For such models, v and w represent the linear and angular velocity of the object and these two parameters are considered constant for each instance of m_t.

Extending on equation (3), equation (6) describes equation (1) with the addition of the constraints pertaining to the dynamic object poses. These constraints are also depicted in figure 2. Here R^{-1}, Q^{-1}, O^{-1} and S^{-1} are the information matrices of their respective terms.

$$\begin{aligned} J(X) = x_0^T \Omega_0 x_0 + \sum_t e_x(t)^T R^{-1} e_x(t) + \\ \sum_t e_{l,x}(t)^T Q^{-1} e_{l,x}(t) + \sum_t e_{m,x}(t)^T O^{-1} e_{m,x}(t) + \\ \sum_t e_m(t)^T S^{-1} e_m(t) \end{aligned} \tag{6}$$

The proposed technique includes two additional terms; those of the measurements of the dynamic objects and the constant velocity model of the dynamic objects linking their consecutive poses.
The error functions between consecutive robot poses and measurements to static objects remain unchanged as represented in equation (4) and (5) respectively.
Equation (7) represents the error function of the measurement to the dynamic object. Here w represents the measurement

function given the poses of the robot and dynamic object m_t and x_t respectively with z_t being the measurement of the dynamic object:

$$e_{m,x}(t) = z_t - w(m_t, x_t) \tag{7}$$

The last term consists of the error function between the estimated pose of the dynamic object m_t and the pose of the constant velocity model $f(v_t, m_{t-1})$ shown in equation (8):

$$e_m(t) = m_t - f(\nu_t, m_{t-1}) \tag{8}$$

Here ν_t represents the constant velocity parameters v_t and w_t of the dynamic object. Parameters v_t and w_t describe the constant linear and angular velocity respectively. Function $f(\cdot)$ describes a straightforward constant velocity model based on the object pose given by $m = [m_x, m_y, m_\theta]$.
To solve equation (6) the error (e_{ij}) terms are derived. This is shown in equation (9) where X represents the set of nodes within the graph. The indices i and j represent the relation between the observations. The obtained Jacobian is a sparse matrix as shown in equation (10).

$$\frac{\partial e_{ij}(X)}{\partial(X)} = (0, ..., \frac{\partial e_{ij}(X_i)}{\partial(X_i)}, ..., \frac{\partial e_{ij}(X_j)}{\partial(X_j)}, ..., 0) \tag{9}$$

$$J = (0, ..., A_{ij}, ..., B_{ij}, ..., 0) \tag{10}$$

Using the sparse Jacobian matrix, the coefficient vector b and matrix H can be calculated as shown in equations (11) and (12).

$$b^T = \sum b_{ij}^T = \sum e_{ij}^T \Omega_{ij} J_{ij} \tag{11}$$

$$H = \sum H_{ij} = \sum J_{ij}^T \Omega_{ij} J_{ij} \tag{12}$$

Once the vector b and the sparse matrix H are calculated, equation (13) can be solved.

$$H\Delta X = -b \tag{13}$$

Here, ΔX is the correction vector which is added to the set of all nodes X. This process is executed iteratively until the change in outcome of the minimization function is below a predefined threshold as shown in algorithm 1. In this algorithm the construction of H and b is formed in the pseudo function $buildLinearSystem(X)$ with X being the set of nodes. After this, the $solveSparse$ function finds ΔX which is added to the node vector X. Lastly, the function $computeGlobalError$ sums up all the errors from the aforementioned error functions using the newly corrected set X. The outcome of this error is compared to the previous error. If the difference is below a certain threshold, the optimization is complete.

Algorithm 1 Graph-based optimization

```
optimize(X):
while !converged do
    (H, b) = buildLinearSystem(X)
    ΔX = solveSparse(HΔX = −b)
    X' = X + Δx
    F_X = computeGlobalError(X)
    F_X' = computeGlobalError(X')
    if abs(F_X − F_X') <= threshold then
        converged = True
    end if
end while
return X
```

A. Static features

In our proposed approach, we assume that a LiDAR provides a perception of the environment. Using a Split-and-Merge (SaM) approach, which originated from computer vision [9] and used in works such as [10] and [11], the intersection points of two line segments within the environment are captured from the data as illustrated in algorithm 2. These corners are considered to be the static features within the graph, with the measurements being the constraints between robot poses and these features. The algorithm splits the laser scan data into sections based on a certain threshold.

Once the data are split into the smaller sections $L = \{s_0, \ldots, s_i\}$, and the line parameters of each section are saved in R, the algorithm determines the corners in $corner(s_i, s_{i-1})$. The line parameters of s_i and s_{i-1} are obtained from R and the intersection is calculated (14, 15). Once the corner is known, it is added to a list of static corner features (16).

Algorithm 2 Split-and-Merge corner detection

```
1: set s_0 = {P}, L = {s_0}, C = ∅, L' = ∅, R = ∅
2: while True do
3:     L' = ∅
4:     R = ∅
5:     for s_i in L do
6:         l_s = line(P_0, P_k)            ▷ P_k last point in cluster s_i
7:         add l_s to R
8:         d_p, P_p = max_distance(l_s, s_i)
9:         if d_p >= threshold then
10:            s_j = {P_0, ..., P_p}
11:            s_k = {P_p, ..., P_k}
12:            add s_j and s_k to L'
13:        else if d_p < threshold then
14:            add s_i to L'
15:        end if
16:    end for
17:    if size(L) == size(L') then
18:        break from loop
19:    end if
20: end while
21: for s_i in L do
22:    c = corner(s_i, s_{i−1})
23:    add corner c to list C
24: end for
25: return L, C
```

$$x_{corner} = \frac{b_i - b_{i-1}}{m_{i-1} - m_i} \tag{14}$$

$$y_{corner} = m_i . x_{corner} + b_i \tag{15}$$

$$C = [[x_{corner,0}, y_{corner,0}]^T, ..., [x_{corner,i}, y_{corner,i}]^T] \tag{16}$$

This process relates to line 13 of the algorithm 2. These intersections are considered to be the static landmarks within the scene and are incorporated as such.

B. Dynamic object

Dynamic objects can range from large rigid objects such as cars or trucks, to people or other robots. Our focus lies in detecting the pose of a person and incorporating this

information into the graph-based optimization. A 3D camera can provide a color and depth image of the environment. Using Blazepose body tracking [12], we are able to detect certain feature points of a person within the image. Assuming that the torso of a person is oriented towards the movement of the person, we can determine the pose of a person.
First, we consider all points within the plane described by the two shoulder points and two hip points, obtained from the body tracking algorithm, to be the torso. From this set of points P_{torso}, the centroid c_p is calculated. This represents the position of the person:

$$c_p = \frac{1}{n}\sum_{i=1}^{n} p_i \text{ with } p_i \in P_{torso} \tag{17}$$

Here n is the number of points present within P_{torso}.
To determine the orientation of the person we first fit a plane through all points P_{torso} using singular value decomposition (SVD). From the SVD, the normal vector is determined which is considered the orientation of the person. Given the set of 3D points P_{torso} and the position of the plane is given by the centroid c_p, matrix A is introduced:

$$A = [P_{torso}] - c_p = [p_0 - c_p, ..., p_k - c_p] \tag{18}$$

$$A = USV^T \tag{19}$$

Computing the singular value decomposition of A provides the normal n obtained from the third column of matrix U:

$$n = U[:, 3] \tag{20}$$

The normal n together with the centroid c_p gives us the pose of the detected person.
Figure 3 shows this pose extraction from the depth camera. For our implementation, this 3D pose is reduced to a 2D pose within the xy-plane for the implementation into the graph-structure.

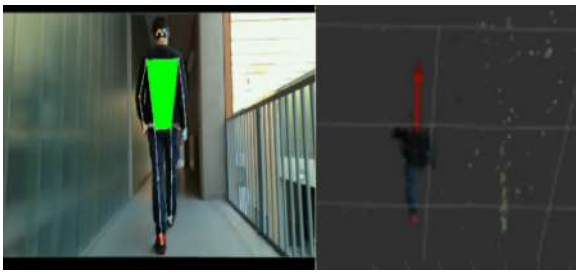


Fig. 3: Detection of pose of a person. The left: the body tracking algorithm and area defined as the torso. On the right, the pose of the person is given by the red arrow with origin at the centroid of the torso of the person.

C. Determining the motion parameters

In order to implement the constraints between consecutive poses of the dynamic objects, the necessary motion parameters need to be estimated which, under the constant velocity model assumption, are v and w. These linear and angular velocity parameters are necessary for equation (8) and represent $\nu_t = [v, w]^T$.

For this estimation a limited set of poses of a dynamic object $M = [m_0^0, ..., m_t^0]$, registered together with at least one landmark $L = [l_0^0, ..., l_t^0]$ and associated over the consecutive time frames, is used. From these poses in M, v and w are calculated using a mean average as illustrated in equations (21, 22).

$$v = \frac{\sum_{i=0}^{t} d(m_i, m_{i-1})}{t} \tag{21}$$

$$w = \frac{\sum_{i=0}^{t} (m_{\theta,i} - m_{\theta,i-1})}{t} \tag{22}$$

Function d in equation (21) calculates the Euclidean distance between the two given poses.
These calculations are executed and, the parameters are updated every iteration of the algorithm until equation 6 is minimized. In our experiments, it is shown that this calculation yields a significant benefit to estimate the pose.

III. Experiments & Results

The experiments are split into two sections. First, we evaluate the graph-optimization approach described in II through simulation. A comparison is made between the standard approach without dynamic objects and with dynamic objects. Second, data from sensors are gathered and processed. From the optimized robot poses, a 2D map is generated. These maps (with and without incorporating dynamic objects) are compared using metrics described by Filatov et al. [13]. Besides these metrics, the generated maps are evaluated against the building plans.

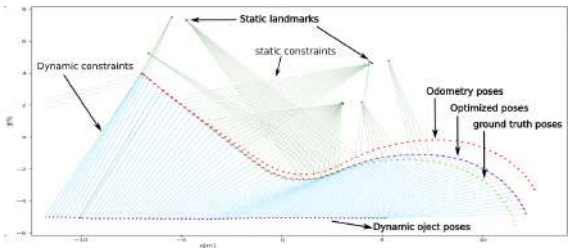
A. Validation through simulation

Within the simulation, several landmarks are sparsely distributed and a dynamic object moves at a constant speed. Gaussian noise is added to all measurements within the simulation. Table I shows the comparison between the standard approach with and without dynamic objects. The mean error of robot positions w.r.t. the actual positions are given as well as those of the dynamic object. For the case where the dynamic object is not incorporated within the optimization, the object poses are only derived from the direct measurements and not influenced by a motion model.

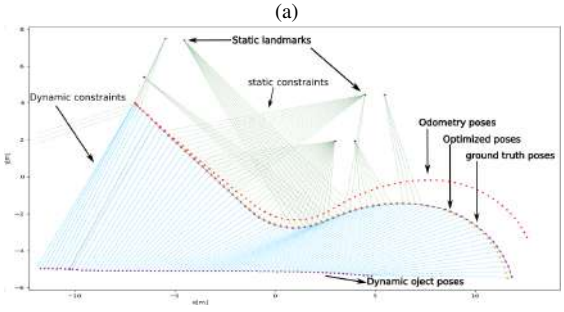
Figure 4 visualizes the output of the simulation. The trajectory of the robot is visualized in dark blue while the trajectory from odometry is given in red. The trajectory is given by yellow dots. The path of the dynamic object is represented by the purple dots. Figure 4a shows the

TABLE I: Mean error (m) of robot and object positions w.r.t. the actual positions. The error is calculated for the standard graph-based optimization without dynamic objects and for the presented approach.

Robot path error (m)	**run 1**	**run 2**	**run 3**	**run 4**	**run 5**
SLAM without dynamic object	0.1156	0.7007	0.1275	0.2715	0.3088
SLAM with dynamic object	0.0780	0.2937	0.1020	0.0745	0.0624
Object path error (m)	**run 1**	**run 2**	**run 3**	**run 4**	**run 5**
SLAM without dynamic object	0.2262	0.7886	0.3378	0.2861	0.3135
SLAM with dynamic object	0.1122	0.2531	0.2573	0.1397	0.1318



(a)



(b)

Fig. 4: Figure a) shows the optimized path without incorporating dynamic object. Figure b) shows the optimized path with the dynamic object incorporated.

optimization without the inclusion of the dynamic object. It can be seen that the estimation does approximate the actual path, but a deviation is still present. Figure 4b shows the results with the dynamic object included. The estimated path is almost identical to the actual robot poses, which shows that the incorporation of dynamic objects has its merits.

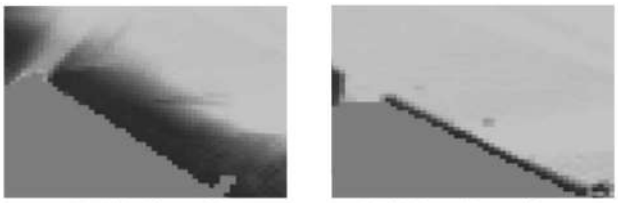
B. Validation through real-world experiments

In our experiments, data was obtained using a differential drive robot containing a Hokuyo UTM-30LX 2D LiDAR and an L515 RealSense depth camera. The platform is equipped with magnetic wheel encoders providing odometry information. The information of these measurements was then implemented in our approach. Using the optimized poses together with the LiDAR information, the 2D grid maps are generated. For our experiments we focused on a long hallway. This situation is prone to drift and leads to corridors being represented by curved walls which is not an accurate representation. During the collection of data, a person is walking in front of the robot through the hallway. The built maps are generated from a single pass through the environment. This experiment is executed multiple times
Figure 8 shows the difference between the standard graph-based SLAM approach and incorporating dynamic objects within the graph optimization. The figures visually show that by detecting, tracking and incorporating dynamic objects within the scene, the environment is represented more truthfully. This was the case in all our experiments.

Besides the visual confirmation that including dynamic objects is beneficial in map building, three metrics described by Filatov et al. in [13] are calculated to compare the generated maps. These metrics are:

- Proportion of occupancy;
- Number of corners present within the map;
- Number of enclosed areas present within the map.

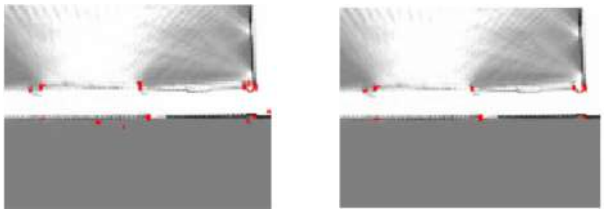
Proportion of occupancy refers to the ratio between cells of the 2D grid map considered to be occupied with high probability and those with low probability. For the same environment, a better representation is considered to have a lower proportion value. In an ideal map, occupied space is represented as a single grid cell with high probability of being occupied rather than a cluster of cells with various probabilities of occupation. Figure 5 shows what the



(a) With blurry effect (b) Without blurry effect

Fig. 5: The proportion metric calculates the amount of blur within the map. The less blur effect the better the map [13].

proportion metric determines. The results of this metric are given in table II. In order to calculate the proportion, the Otsu threshold of the occupied cells is calculated. Using this threshold, the amount of occupied cells above this value is divided by the amount of occupied cells below this value. Next to this, the standard deviation is also given.
Number of corners present within the map is a metric which indicates the smoothness of the map. The less corners a map has, the better the representation of the environment. For our experiments, a Harris corner detection algorithm [14] is implemented for calculating the corners within the map. This does not represent physical corners, but every pixel considered a corner by the algorithm. Figure 6 shows an example of a partial map and its calculated corners.



(a) More corners detected (b) Less corners detected

Fig. 6: The red dots represent the corners found by the Harris corner detection.

Table II also shows the results for this metric between the

maps generated with the standard approach and the proposed approach.

Number of enclosed areas present within the map especially indicates misalignment. If fewer enclosed areas are present within the map, the better the representation of the environment. Misalignment of laser scan data will create additional areas within the map. These contours are found by implementing [15], which describes Suzuki's contour tracing algorithm. As these contours describe the enclosed areas found within the map, the number of contours retrieved by the algorithm is a measure of accuracy of the map. Figure 7 is an example of all contours found in a map. The result of this metric is shown in the last column of table II.

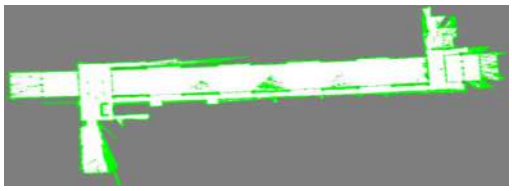
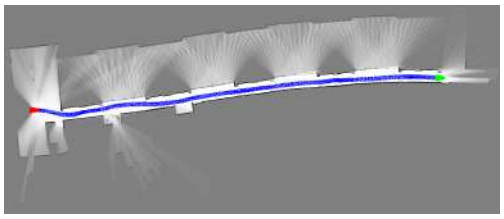
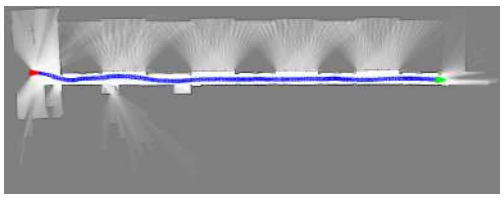


Fig. 7: Example of finding all contours within a map. The green lines form all enclosed areas.



(a)



(b)

Fig. 8: Figure a) is constructed with the standard graph-based SLAM approach. The straight hall is represented as a curvature within the map. Figure b) is constructed using the dynamic SLAM approach. In both images, the robot path is given from its starting point (red) to the end destination (green). All poses in between are represented in blue.

Looking at table II, the proportion between occupied and unoccupied space is similar for the maps created with and without the incorporation of a walking person within the scene. The difference between the values is rather small and sometimes being a positive or negative difference. We conclude that this metric does not provide a clear indication which map is a more truthful representation of the environment. The amount of Harris corners detected within the maps clearly indicates that the proposed approach ensures a smoother representation of the environment. In every run a significant decrease in number of corners is visible. The number of enclosed areas also indicates that the described approach has its merits. A decrease in the number of enclosed areas is present within the evaluation of every run. This indicates that there is less misalignment of the LiDAR data during mapping, thus representing the environment more accurately. Together with a visual inspection of the maps, it is clear that incorporating dynamic objects within the graph-based optimization provides an improved estimation of the robot's trajectory.

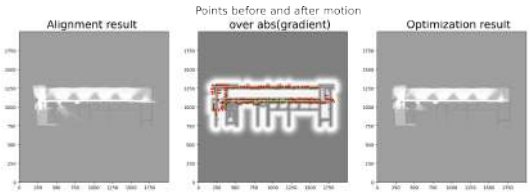


Fig. 9: Alignment of the constructed map with the building plans based on the work of [16].

When matching the images against the building plans, it is obvious that the generated map does represent the environment truthfully. This is visible in figure 11 especially when it is compared to figure 10, which is the graph based optimization without the incorporation of dynamic objects within the optimization. In order to properly compare the constructed

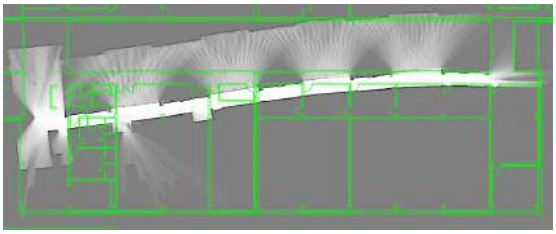


Fig. 10: Comparing building plans to the generated map with the standard graph-based optimization approach. The building plans, visible in green, are overlaid on the map.

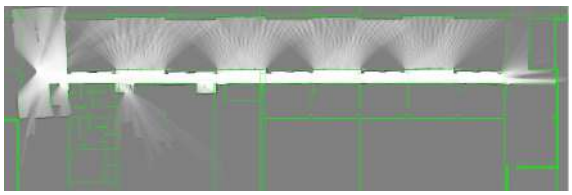


Fig. 11: Overlaying the building plan with the generated map of the proposed approach shows that the constructed map represents the environment more accurately.

map with the building plans, the plans are converted to an occupancy grid image. After an initial manual alignment of the generated maps, the affine transformation to match the map to the building plans is found based on the work of Shahbandi et al. [16] and is shown in figure 9. Once the map is aligned with the image of the building plans, as shown in figure 9, a binarization of the constructed map is executed. From the binarized map, the pixel distance to its nearest

TABLE II: Table showing the results of the metrics; proportion, corner count and number of enclosed areas.

✗= without incorporation of dynamic object			✓= with incorporation of dynamic object		
runs	**Otsu**	**STD**	**Proportion**	**Harris corners**	**enclosed areas**
✗ 1	70	0.9818	0.3849	5043	2725
✓ 1	78	0.9958	0.4372	4349	2683
✗ 2	68	0.8130	0.4372	6358	1726
✓ 2	63	0.7230	0.4663	4111	1601
✗ 3	69	0.8400	0.4260	5077	1229
✓ 3	66	0.7990	0.4414	4073	1143
✗ 4	63	0.7320	0.4630	5718	1641
✓ 4	58	0.4920	0.4924	4317	1559
✗ 5	76	0.9474	0.3930	4614	2915
✓ 5	77	0.9340	0.3950	4506	2723

neighbour on the image of the building plan is calculated as a metric representing the similarity to the real environment. Table III shows the results of this metric. The mean distance error over several mapping runs is limited to around 2 pixels. The standard deviation is also limited to around 3.5 pixels. Considering that each pixel of the map and building plans is equivalent to an area of 0.05 x 0.05 meters, and the hallway describing a distance of 65 meters, we can conclude that the maps represent the environment fairly accurate.

TABLE III: Table showing the comparison of the binarized map w.r.t. the building plan expressed in pixel amount. Every pixel represents an area of 0.05 x 0.05 meters.

run	**1**	**2**	**3**	**4**	**5**
mean error	2.11 px	1.98 px	1.76 px	1.57 px	2.32 px
standard dev.	4.38 px	3.85 px	3.20 px	2.54 px	3.60 px

IV. Conclusion

This paper describes a procedure to include dynamic objects within graph-based SLAM. A method is defined to register a human pose together with a determination of the constant velocity model parameters. Our experiments show that the incorporation of dynamic objects can improve the robots trajectory as shown in simulation I and that the representation of the environment is can be more accurate as shown in the real world experiments and based on several metrics from literature II. Lastly, the constructed maps are compared to the building plans of the hallway. An alignment is performed between the maps and the building plan and the mean distance error is calculated. These results shown in table III confirm that the newly constructed maps represent the environment with a high accuracy.

Future work can focus on eliminating the assumptions made in this paper. An important aspect to further development is the implementation of a detection and tracking algorithm to register several dynamic objects. We would also like to address the constant velocity model assumption and investigate other motion models of the dynamic features. To this end several examples in literature exist [17]- [19].

References

[1] M. Simas, B.J. Guerreiro and P. Batista, "Preliminary Results on 2-D Simultaneous Localization and Mapping for Aerial Robots in Dynamics Environments", International Conference on Robot Intelligence Technology and Applications, 2019

[2] H. Zhang, H. Uchiyama, S. Ono and H. Kawasaki MOTSLAM: MOT-assisted monocular dynamic SLAM using single-view depth estimation. IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 4865-4872), 2022.

[3] J. Huang, S. Yang, T. Mu, and S. M. Hu. Clustervo: clustering moving instances and estimating visual odometry for self and surroundings. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.2020, pp. 2168-2177.

[4] J. Zhang, M. Henein, R. Mahoney and V. Ila. "VDO-SLAM: a visual dynamic object-aware SLAM system." arXiv preprint arXiv:2005.11052 (2020)

[5] B. Bescos, C. Campos, J. D. Tardos and J. Neira. "DynaSLAMII: Thightly-coupled Multi-object Tracking and SLAM" IEEE robotics and automation letters, 2021, 6(3), 5191-5198.

[6] P. Aerts, P. Slaets, and E. Demeester. Incorporating Moving Landmarks within 2D Graph-Based SLAM for Dynamic Environments. In 6th International Conference on Mechanical Engineering and Robotics Research (ICMERR) (pp. 1-7). IEEE. 2021.

[7] G. Grisetti and R. Kummerle and C. Stachniss and W. Burgard: A tutorial on graph-based SLAM. IEEE Intelligent Transportation Systems Magazine, 2(4). 31-43 2010

[8] C. Schöller and V. Aravantinos and F. Lay and A. Knoll: The simpler, the better: Constant velocity for pedestrian motion prediction. arXiv preprint arXiv:1903.07933, 5(6), 7. 2019

[9] T. Plavidis, S.L. Horowitz. Segmentation of plane curves. IEEE transactions on Computers, 100(8), 860-870, 1974.

[10] G.A. Borges, M.J. Aldon. Line extraction in 2D range images for mobile robotics. Journal of intelligent and Robotic Systems, 40(3), 267-297, 2004.

[11] L. Zhang, B.K. Ghosh. Line segment based map building and localization using 2D laser rangefinder. In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH370645)(Vol. 3, pp.2538-2543), 2000

[12] V. Bazarevsky and I. Grishchenko and K. Raveendran and T. Zhu and F. Zhang and M. Grundmann. Blazepose: On-device real-time body pose tracking. arXiv preprint arXiv:2006.10204. 2020

[13] A. Filatov, K. Krinkin, B. Chen, and D. Molodan: 2D SLAM quality evaluation methods. IEEE 21st conference of Open Innovations Association (FRUCT). 2017, pp.120-126

[14] C. Harris and M. Stephens: A combined corner and edge detector. In Alvey vision conference (Vol 15, No 50, pp10-5244) 1988

[15] S. Suzuki and K. Abe: Topological Structural Analysis of Digitized Binary Images by Border Following. In Computer vision, graphics and image processing, 1985, 30(1), 32-46

[16] S.G. Shahbandi and M. Magnusson and K Iagnemma: Nonlinear optimization of multimodal two-dimensional map alignment with application to prior knowledge transfer. IEEE Robotics and Automation Letters, 3(3), 2040-2047 2010

[17] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. Physical Review E, 1995.

[18] R. Baxter, M. Leach, S. Mukherjee, and N. Robertson. An adaptive motion model for person tracking with instantaneous head-pose features. Signal Processing Letters, 2015.

[19] M. Koschi, C. Pek, M. Beikirch, and M. Althoff. Set-based prediction of pedestrians in urban environments considering formalized traffic rules. In International Conference on Intelligent Transportation Systems (ITSC), 2018.

Visual-LiDAR Odometry and Mapping with Monocular Scale Correction and Visual Bootstrapping

Hanyu Cai[1], Ni Ou[1] and Junzheng Wang[1,*]

***Abstract*— This paper presents a novel visual-LiDAR odometry and mapping method with low-drift characteristics. The proposed method is based on two popular approaches, ORB-SLAM and A-LOAM, with monocular scale correction and visual-bootstrapped LiDAR poses initialization modifications. The scale corrector calculates the proportion between the depth of image keypoints recovered by triangulation and that provided by LiDAR, using an outlier rejection process for accuracy improvement. Concerning LiDAR poses initialization, the visual odometry approach gives the initial guesses of LiDAR motions for better performance. This methodology is not only applicable to high-resolution LiDAR but can also adapt to low-resolution LiDAR. To evaluate the proposed SLAM system's robustness and accuracy, we conducted experiments on the KITTI Odometry and S3E datasets. Experimental results illustrate that our method significantly outperforms standalone ORB-SLAM2 and A-LOAM. Furthermore, regarding the accuracy of visual odometry with scale correction, our method performs similarly to the stereo-mode ORB-SLAM2.**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is an irreplaceable technique for mobile robots and autonomous vehicles, providing reliable surrounding environment information and real-time positions. According to the use of sensors, this technique can be divided into two categories: visual-based and LiDAR-based. Over the past two decades, visual SLAM has made significant strides, resulting in commercially available frameworks. Modern visual SLAM algorithms develop into two branches: feature-based and direct methods. Feature-based methods [1], [2] reduce the reprojection error of matched feature points (keypoints) through bundle adjustment (BA) [3]. On the other hand, direct methods normally optimize the photometric error of sparse keypoints without corresponding matchings [4], [5]. The advantage of visual SLAM is rich semantic information, low cost and small size, which is an indispensable part of the field of automatic driving and AR.

In most cases, LiDAR SLAM usually outperforms visual SLAM. Most recent pure LiDAR SLAM methods are developed based on LOAM [6], a milestone LiDAR SLAM framework combined with SCAN-to-SCAN and SCAN-to-MAP registration modes. These LOAM-based techniques yield superior performance compared to the baseline LOAM, with improvements in efficiency [7], robust registration [8], motion compensation [9] and local optimization [10]. In addition, LiDAR-based loop closure detection techniques [11], [12] have been widely employed for place recognition and graph optimization to reduce the accumulated error of LiDAR SLAM further.

Nonetheless, standalone visual or LiDAR SLAM either has intractable drawbacks. Visual SLAM systems are prone to localization failure [1] in fast motions. On the other hand, for LiDAR SLAM, motion distortion [6], [9] is still a tricky problem for spinning LiDAR, and its loop closure detection is more complicated and challenging due to lack of stable features [12]. It has been a noticeable trend to fuse visual and LiDAR SLAM to enhance the overall performance.

According to the fusion techniques, LiDAR-camera fused SLAM can be divided into three categories: LiDAR-assisted visual SLAM, vision-assisted LiDAR SLAM, and vision-LiDAR coupled SLAM. The first two means rely mainly on LiDAR or camera, and the other sensor takes the assistance role. Moreover, the last type generally utilizes both visual and LiDAR odometry in the system.

The first category tends to focus on image depth enhancement [13] or combines with direct methods without estimating the depth of feature points [14]. The second category has few related studies, and it often uses visual information to help LiDAR SLAM perform loop closure detection or render map texture [15]–[17]. Since this category is not the research content of this paper, we will not describe it in detail. The third category is the hot field of current research, which can be subdivided into loosely coupling and tightly coupling. Loosely coupling is to cascade the two or filter the results of the two [18], [19]. Tightly coupling [20] focuses on constructing a joint optimization problem, including vision and LiDAR factors for state estimation.

Our work is deeply related to depth enhancement. Whereas the error of depth enhancement is significant when the point cloud is sparse, and the feature points with enhanced depth may not be successfully tracked. Directly tracking projected points with high gradients is a solution, but such points cannot be tracked accurately and stably. In this study, we combine the powerful tracking ability of the feature-based method with optical flow and propose a novel scale correction method to address the monocular scale drift problem. Moreover, considering the LOAM algorithm depends on the constant velocity model, it is prone to failure in scenes with excessive acceleration or degradation. Using the results of the visual odometry to initialize the LiDAR odometry's pose can increase the LOAM performance.

The contributions of this paper are as follows:

*This work was supported by State Key Laboratory of Intelligent Control and Decision of Complex Systems

[1]Hanyu Cai, Ni Ou, and Junzheng Wang are with School of Automation, Beijing Institute of Technology, Beijing, China. Hanyu: caihanyu4258@163.com, Ni: 3120205431@bit.edu.cn

* Corresponding author: wangjz@bit.edu.cn

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

1) A visual-LiDAR loosely coupled odometry. Solve the problem that LOAM fails in degradative scenarios, and increase the performance.
2) A novel scale correction algorithm is proposed that does not need to enhance the depth of the visual feature point. It guarantees that the output of the visual odometry will not have a significant drift.
3) Implement our system on a large-scale dataset and verify its effectiveness.

This paper is organized as follows. Section II presents studies related to our work. Section III introduces the proposed loosely coupled system and our scale correction algorithm. Section IV shows the experimental datasets and results. Finally, Section V demonstrates our conclusion and possible extensions to our work.

II. Related Work

LiDAR-camera SLAM can be broadly classified into three categories: LiDAR-assisted visual SLAM, vision-assisted LiDAR SLAM, and vision-LiDAR coupled SLAM. Note that vision-assisted LiDAR SLAM systems [16] are not comprehensively reviewed in this paper because this system usually hinges on semantic information, which requires knowledge of image recognition that is out of our scope.

A. LiDAR-assisted Visual SLAM

LiDAR-assisted visual SLAM generally aims to utilize LiDAR's point cloud data to obtain more accurate depth information for image feature points. A typical method in this category is LIMO, where LiDAR data is directly applied to estimate the depth of feature points [13]. Yuewen et al. proposed CamVox, an RGBD SLAM system combined with Livox LiDAR [21]. The performance of outdoor RGBD cameras is improved by depth enhancement, and the depth of many enhanced feature points reaches 100 meters. Another approach to using LiDAR data in visual SLAM is by projecting point clouds onto images and performing the direct method on projected points [14]. Reproject the projected points to the next frame image and then minimize the photometric error to solve the pose. This method does not have the error caused by depth enhancement, but it requires accurate extrinsic parameters between the camera and LiDAR. LiDAR points are too sparse compared to image pixels, and the above methods can obtain the depth of a small number of points. In order to increase the number of pixels with depth, Varuna et al. used the Gaussian process regression on the projected points from LiDAR to the image to improve the depth estimation [22]. Within a local image patch, they use the enhanced depth pixels as a priori to predict the depth of the remaining pixels in the image patch. In addition to depth enhancement, LiDAR can also improve the robustness of visual SLAM to illumination, which is also reflected in CamVox [21]. Jiawei Mo et al. proposed a method that uses LiDAR's descriptor to address the issue that visual loop closure detection is heavily affected by illumination changes [23]. They calculate the LiDAR point cloud into three descriptors and store them. The stereo SLAM map is also calculated as three descriptors and matched with the LiDAR descriptors. This method only relies on three-dimensional points to complete visual loop closure detection.

To summarize, depth enhancement is the most popular technique in LiDAR-assisted visual SLAM. In this paper, we propose a novel approach that can apply to the low-resolution LiDAR case, where the density of LiDAR point clouds is much lower than that of the camera images.

B. Vision-LiDAR Coupled SLAM

In contrast to LiDAR-assisted visual SLAM, vision-LiDAR coupled SLAM integrates both visual and LiDAR odometry modules to enhance the system's accuracy. V-LOAM is a loosely coupled system that combines visual and LiDAR odometry modules [24]. In this study, visual odometry recovers the depth of feature points from surrounding projected LiDAR points, while LiDAR odometry leverages high-frequency camera poses to mitigate drift. However, V-LOAM still faces two significant issues: ineffective depth enhancement and non-negligible drift error on the z-axis (also remains in its baseline [6]). Zikang Yuan et al. proposed SDV-LOAM [19]. It tracks the high-gradient projected LiDAR points as visual odometry and employs an adaptive scan-to-map optimization method to constrain pose in all six dimensions well. By contrast, TVL-SLAM [20] does not enhance the visual odometry's depth estimation nor utilize the motion estimation from visual odometry as the LiDAR odometry's initial guess. Instead, it establishes a joint optimization problem of visual and LiDAR features, thereby establishing a tightly coupled system.

The advantage of loose coupling is that the system structure is simple and the precision is high, but the robustness is not strong due to the influence of each module. Tight coupling is generally more robust due to joint state estimation but requires more computation.

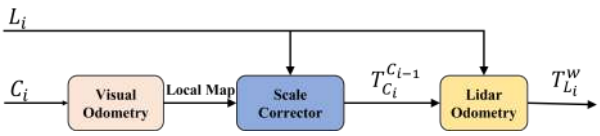


Fig. 1. Overview of our method.

III. Methodology

A. System Overview

The overview figure of our method is shown in Fig. 1, and the definitions of primary notations are present in Table I. Our system synchronizes the camera and LiDAR data at 10Hz. During the first stage, a local vision map is generated using the mono camera initialization or tracking. Subsequently, we utilize LiDAR data to estimate the monocular scale factor that represents the ratio between the corresponding vision local map and laser scan. However, due to the scale drift of the monocular odometry, we correct the scale factor periodically during the trajectory using the proposed scale corrector. Following scale correction, the LiDAR odometry

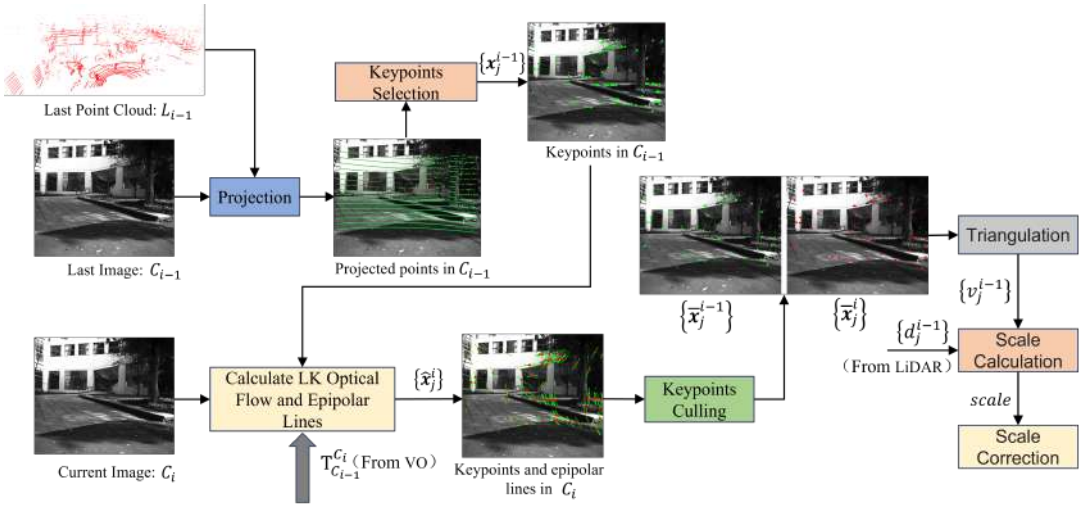


Fig. 2. Pipeline of the Scale Corrector.

TABLE I
NOMENCLATURE

Notations	Description
C_i	i^{th} frame of image keyframe
L_i	i^{th} frame of point cloud
$\{\boldsymbol{x}_j^i\}$	keypoints projected from L_i onto C_i
$\{\widehat{\boldsymbol{x}}_j^i\}$	reserved keypoints of $\{\boldsymbol{x}_j^i\}$ after optical flow tracking
$\{\overline{\boldsymbol{x}}_j^i\}$	reserved keypoints of$\{\widehat{\boldsymbol{x}}_j^i\}$ after keypoint culling
$\mathbf{T}_A^B$	transformation of A with respect to B
$\mathbf{P}_w^i$	coordinates of i^{th} map point with respect to the world
$\mathbf{K}$	camera intrinsic matrix
d_j^i	measured depth of the j^{th} projected point onto C_i
v_j^i	visual depth of the j^{th} projected point onto C_i
$\boldsymbol{p}_j^i$	LiDAR point corresponding to $\boldsymbol{x}_j^i$

generates the final pose with the initial guess from the visual odometry (we call it visual bootstrapping), resulting in a final localization frequency of 10 Hz. The visual odometry and LiDAR odometry are implemented based on ORB-SLAM2 [1] and A-LOAM [6], respectively, so we focus on performance comparison with the two baselines in our experiments part (Section IV).

The remaining parts (Section III-B and Section III-C) jointly introduce the implementation of the proposed scale corrector. The pipeline of our scale corrector is displayed in Fig. 2. To start with, we project the last frame of point cloud L_{i-1} onto the corresponding image C_{i-1} and select keypoints $\{\boldsymbol{x}_j^{i-1}\}$ among the projected points. Subsequently, the optical flow algorithm is employed to track each $\boldsymbol{x}_j^{i-1}$ in the current image C_i and thus get $\{\widehat{\boldsymbol{x}}_j^{i-1}\}$ and $\{\widehat{\boldsymbol{x}}_j^i\}$ simultaneously. Moreover, to guarantee the accuracy of keypoint correspondence, we also design two criteria for keypoints culling based on epipolar lines, which are further introduced in (3) and (4). Based on this keypoint matching, we can conduct triangulation between matched $\{\overline{\boldsymbol{x}}_j^{i-1}\}$ and $\{\overline{\boldsymbol{x}}_j^i\}$ to recover their depth in the local map. Finally, scale correction is performed between the local map and the corresponding laser scan periodically throughout the trajectory.

B. Scale Corrector: Keypoint Extraction

1) Projection and Matching: As outlined in Section III-A, the content of this section includes the projection, matching and culling steps of keypoints. For clarity, we did not take image distortion into account. Then, the process of projection between C_{i-1} and L_{i-1} can be formulated in (1).

$$\boldsymbol{x}_j^{i-1} = \frac{1}{d_j^{i-1}} \mathbf{K} \mathbf{T}_L^C \boldsymbol{p}_j^{i-1} \tag{1}$$

where $\boldsymbol{p}_j^{i-1}$ is the j^{th} point of L_{i-1}, $\mathbf{T}_L^C$ is the extrinsic parameter between camera and LiDAR. Imprecise extrinsic will cause a significant error, and the corresponding calibration method is shown in our previous work [25].

Further, the following criteria are applied to filter out distinctive $\{\boldsymbol{x}_j^{i-1}\}$ through neighbouring image information.

a) $\boldsymbol{x}_j^{i-1}$ should meet the requirements of the FAST-9 [26] corner.
b) The image gradient at $\boldsymbol{x}_j^{i-1}$ should be large enough.

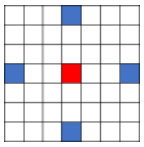
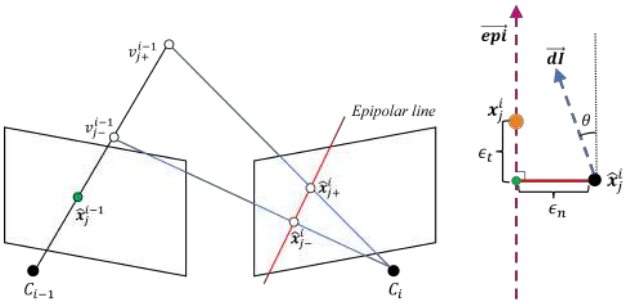


Fig. 3. FAST-12 pre-testing process. The red point is the keypoint to be tested. The blue points are domain image points.

However, the first criterion is not applicable to low-resolution LiDAR due to the scarcity of projected points. To resolve this issue, we lower the requirement to obtain $\{\boldsymbol{x}_j^{i-1}\}$ as shown in Fig. 3. We adopt the FAST-12 pre-testing process. Calculate the difference in the pixel values between the keypoint and the surrounding four points, and if more than three meet the threshold, our requirements are met. In

addition, we employ non-maximum suppression to ensure a uniform distribution of keypoints.

Regarding keypoint matching, we employ the Lucas-Kanade [27] optical flow with the input of $\{\boldsymbol{x}_j^{i-1}\}$ from the last image to track corresponding points $\{\widehat{\boldsymbol{x}}_j^i\}$ in the current image.



(a) Epipolar line (b) Error directions

Fig. 4. (a): The projection keypoint will theoretically lie on the epipolar line. (b): The error between the tracked keypoint(black) and the theoretical point(orange) is divided into tangential and normal errors.

2) Culling: Since many keypoints are not Fast corners, tracking these keypoints by optical flow will cause significant uncertainty. To further improve the accuracy of keypoint matching, we conduct keypoint culling based on the epipolar lines. Fig. 4(a) shows the conception of the epipolar line. According to epipolar geometry, the keypoint $\widehat{x}_j^i$ should be located on the epipolar line. For one, it should be discarded when its distance to the epipolar line is too large. For another, in some special cases, the keypoint still should be culled when its pixel gradient is perpendicular to the epipolar line even though it is near the epipolar line. The reason is that other points distributed along the pixel gradient direction are also likely to be extracted to match this epipolar line, thereby increasing the uncertainty of its distance to the epipolar line. To cull the keypoints under the above circumstances, we propose two errors indicated in Fig. 4(b), denoted as normal error ϵ_n and tangential error ϵ_t. We will formulate them in the following parts of this section.

According to the theory of epipolar line, $\widehat{\boldsymbol{x}}_j^{i-1}$ and $\widehat{\boldsymbol{x}}_j^i$ can theoretically be constrained by (2).

$$(\widehat{\boldsymbol{x}}_j^i)^T \mathbf{K}^{-T} (\mathbf{t}_{C_{i-1}}^{C_i})_\times \mathbf{R}_{C_{i-1}}^{C_i} \mathbf{K}^{-1} \widehat{\boldsymbol{x}}_j^{i-1} = 0 \tag{2}$$

where $\mathbf{R}_{C_{i-1}}^{C_i}$ and $\mathbf{t}_{C_{i-1}}^{C_i}$ are the rotation and translation parts of $\mathbf{T}_{C_{i-1}}^{C_i}$ respectively, and $(\cdot)_\times$ represents an antisymmetric matrix. More obviously, from the formula (2), the equation of the epipolar line can be obtained as $Ax + By + C = 0$.

Based on this definition, the quality of tracking points can be evaluated quantitatively. As displayed in Fig. 4(b), we propose two evaluation metrics of different directions. Intuitively, as formulated in (3), the **normal error** ϵ_n is evaluated through the distance between $\widehat{\boldsymbol{x}}_j^i$ and epipolar line.

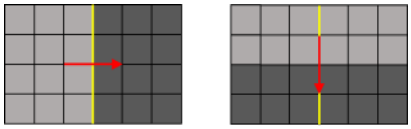


Fig. 5. Two extreme cases of pixel gradient and epipolar line directions. The yellow line is the epipolar line, and the red is the pixel gradient. **Left:** The two are perpendicular; many similar pixels are on the epipolar line. Thus, the matching uncertainty on the epipolar line is significant. **Right:** The two are parallel; the boundary pixels have a higher degree of discrimination than other pixels on the epipolar line. Thus, the matching uncertainty on the epipolar line is small.

We also set a threshold (0.5) to filter out fine points subject to this condition.

$$\epsilon_n = \frac{|A\widehat{\boldsymbol{x}}_{j.x}^i + B\widehat{\boldsymbol{x}}_{j.y}^i + C|}{\sqrt{A^2 + B^2}} < 0.5 \tag{3}$$

where $\widehat{\boldsymbol{x}}_{j.x}^i$ & $\widehat{\boldsymbol{x}}_{j.y}^i$ are the x & y coordinates of $\widehat{\boldsymbol{x}}_j^i$, respectively.

Before explaining the tangential error, it is necessary to introduce optical flow again. Optical flow relies on pixel gradient to track the keypoint, usually using an image patch around the keypoint to increase accuracy. The same trick is used in the epipolar search [4]. Therefore, we can refer to the epipolar search to give a qualitative description of the tangential error. Inspired by [28], the angle between the epipolar line direction and the pixel gradient can be used to describe the matching uncertainty along the epipolar tangential direction. Fig. 5 shows two extreme cases. The larger the angle between the pixel gradient and the epipolar line, the more considerable the uncertainty along the epipolar tangential direction.

Consequently, for a keypoint $\widehat{\boldsymbol{x}}_j^i$ tracked by optical flow, we denote $\overrightarrow{\boldsymbol{epi}}$ and $\overrightarrow{\boldsymbol{dI}}$ as the epipolar line direction vector and pixel gradient vector, respectively, as shown in Fig. 4(b). Then, we can define the $|\cos\theta|$ and its threshold in (4).

$$|\cos\theta| = \left| \frac{\overrightarrow{\boldsymbol{epi}} \cdot \overrightarrow{\boldsymbol{dI}}}{\|\overrightarrow{\boldsymbol{epi}}\| \cdot \|\overrightarrow{\boldsymbol{dI}}\|} \right| > 0.5 \tag{4}$$

Where θ is the angle between $\overrightarrow{\boldsymbol{epi}}$ and $\overrightarrow{\boldsymbol{dI}}$. The **tangential error** ϵ_t may be more significant if $|\cos\theta|$ is smaller than the threshold according to the matching uncertainty from the previous analysis.

At the end of keypoint culling, the points not subject to (3) and (4) are discarded, thereby reserving reliable matched points $\{\overline{\boldsymbol{x}}_j^i\}$ and $\{\overline{\boldsymbol{x}}_j^{i-1}\}$.

3) Scale Calculation: With matched keypoints $\{\overline{\boldsymbol{x}}_j^i\}$ and $\{\overline{\boldsymbol{x}}_j^{i-1}\}$, we can restore the depth of each point $\overline{\boldsymbol{x}}_j^{i-1}$ by triangulation and calculate the scale factor s_j^{i-1} through being dividing by the measured depth d_j^{i-1}, which is the distance of LiDAR point $\boldsymbol{p}_j^i$ previously projected to C_i in (1).

$$s_j^{i-1} = \frac{d_j^{i-1}}{v_j^{i-1}} \tag{5}$$

Note that there are probably a considerable proportion of outliers among $\{s_j^{i-1}\}$, so we introduce RANSAC [29] for outlier rejection and output the mean of inliers as the final scale factor.

C. Scale Corrector: Scale Correction

In this section, we detail how to apply scale correction to the whole SLAM system. As mentioned in Section III-A, our visual odometry is implemented based on ORB-SLAM2 [1]. We remove the loop closing thread and employ scale correction during local mapping process. Without loop detection and closure, the scale of local map is unstable, and thus we periodically correct the scale of local map throughout the trajectory.

At the first stage, denote $\{\mathbf{T}_w^{C_0}, \mathbf{T}_w^{C_1}, \mathbf{T}_w^{C_2} \dots \mathbf{T}_w^{C_m}\}$ as the poses of keyframes in the local map and $\{\mathbf{P}_w^0, \mathbf{P}_w^1, \mathbf{P}_w^2 \dots \mathbf{P}_w^n\}$ as the constituent map points of the local map. Note that these values are all with respect to the world coordinate system. Therefore, we transform poses and map points to reference frame C_0 using $(\mathbf{T}_w^{C_0})^{-1}$. Subsequently, in the local map coordinate system, we can correct the scale of the local map after local bundle adjustment. Finally, the local map is transformed into the world coordinate system again for the sake of compatibility with ORB-SLAM2.

Notably, we do not frequently correct the scale, as this can interfere with the local mapping thread and cause a loss of efficiency. Instead, the scale correction is only triggered when $|scale - 1| \geq 2\%$, where $scale$ is the final scale factor calculated by the scale corrector.

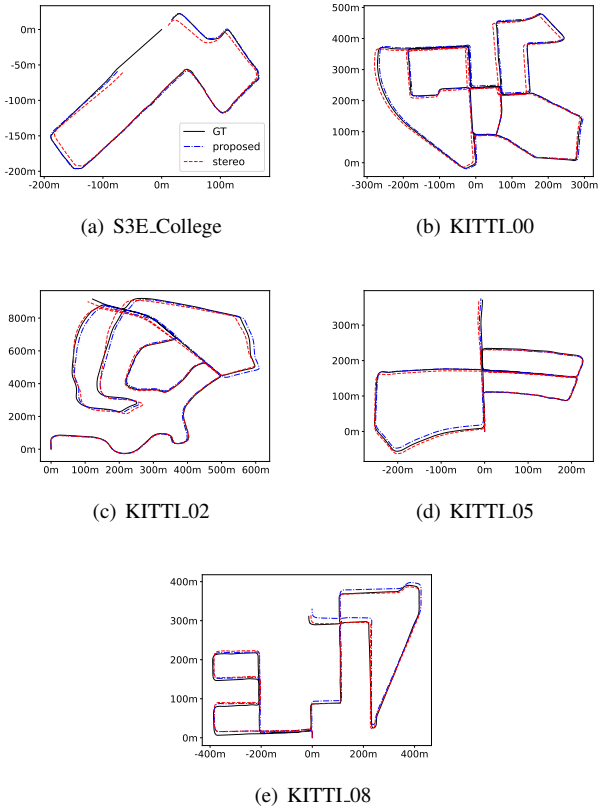
IV. Experiments

We evaluate the performance of the proposed system on KITTI Odometry and S3E datasets. They both incorporate data collected from visual and LiDAR sensors. Four challenging sequences with long distances are selected for evaluation. Regarding data setting, KITTI Odometry uses *HDL-64E* LiDAR and *FL2-14S3M-C* cameras, while S3E uses *VLP-16* LiDAR and *HikRobot MV-CS050-10GC* cameras, which is more challenging for scale correction due to the vertical sparsity of reprojected LiDAR points. Note that we have presented a solution to the sparsity issue in Section III-B.1.

Given that our method is developed based on ORB-SLAM2 [1] and A-LOAM [6], we focus on comparing the localization performance of our system to that of these two baselines. In addition, we also compared with SDV-LOAM [19], one of the state-of-the-art algorithms introduced in Section II-B. All SLAM systems are performed on a laptop with a single-core AMD 6800H @3.2GHz.

A. Effectiveness of Scale Corrector

To verify the effectiveness of the proposed scale corrector, we compare the absolute rotation and translation error (ATE & ARE) between our visual odometry and the stereo-mode ORB-SLAM2. The formulation and implementation of the two metrics can be found in *evo* [30] tool.



(a) S3E_College (b) KITTI_00

(c) KITTI_02 (d) KITTI_05

(e) KITTI_08

Fig. 6. Trajectories estimated by visual odometry. Other legends are consistent with (a). In S3E, the pose of some frames cannot be estimated due to monocular initialization.

It should be noted that the ground-truth poses of the S3E dataset are provided by RTK without orientation (ARE is not evaluated for S3E), which worked at a much lower rate than the camera. In addition, the extrinsic calibration between RTK and camera (left) is not given. To solve these problems, we interpolate the trajectory of the visual odometry using timestamps to synchronize the predicted poses to the ground truth values using *evo* and meanwhile employ Umeyama [31] alignment between the predicted and ground-truth trajectories. Quantitative results on five representative sequences are shown in Table II while corresponding qualitative results are drawn in Fig 6. When loop closure is banned for both, our visual odometry yields better performance than stereo-mode ORB-SLAM2 in most cases, indicating the effectiveness of our scale correction module. Regarding underlying reasoning, we assume that our method is more capable of correcting the depth of distant keypoints due to the assistance of scale corrector, which is challenging for stereo vision as the parallax is not sufficient enough in this case. Moreover, we change the reference coordinate system during local optimization to the earliest keyframe in the local map, which reduce the value during optimization compared to the original solution and bring a slight performance improvement.

TABLE II
TRAJECTORY ERRORS OF SLAM METHODS

Sequence / Length		Ours VO	ORB-SLAM2(Stereo)		Ours VLO	A-LOAM	SDV-LOAM
KITTI_00 / 3724m	ATE(m)	**5.631**	8.946	translationl RMSE(%)	1.182	1.655	**0.9836**
	ARE(deg)	**1.791**	1.920	rotational error(deg/m)	0.0061	0.0078	**0.0041**
KITTI_02 / 5067m	ATE(m)	**13.53**	17.20	translationl RMSE(%)	3.263	11.26	**0.8022**
	ARE(deg)	**1.821**	3.300	rotational error(deg/m)	0.0103	0.0307	**0.0024**
KITTI_05 / 2205m	ATE(m)	5.096	**4.460**	translationl RMSE(%)	1.4496	4.7189	**0.7036**
	ARE(deg)	**0.6319**	1.100	rotational error(deg/m)	0.0065	0.0155	**0.0030**
KITTI_08 / 3222m	ATE(m)	13.98	**12.47**	translationl RMSE(%)	1.895	5.100	**1.1031**
	ARE(deg)	**1.803**	1.824	rotational error(deg/m)	0.0075	0.0187	**0.0037**
[1]S3E_College / 920m	ATE(m)	**1.673**	5.374	ATE(m)	**3.097**	5.505	[2]Failed
	ARE(deg)	–	–	ARE(deg)	–	–	

[1] The ground truth of the S3E dataset has only the translation part, and the rotation part is the unit quaternion.
[2] SDV-LOAM fails on S3E_College.

(a) GT (b) A-LOAM (c) Ours

Fig. 7. Performance of degraded scenes. On a big detour with a degraded scene, A-LOAM makes wrong pose estimates, while ours works well.

B. Effectiveness of Visual Bootstrapping

As for the verification of the effectiveness of Visual Bootstrapping for the LiDAR odometry, we compare it with the baseline A-LOAM [6] and SDV-LOAM [19] on the same datasets shown in Section IV-A. However, there is a slight difference in evaluation. For the KITTI dataset, we replace the *evo* tool with the official KITTI evaluation tool [32] for localization evaluation since it better demonstrates the drift degree in a long distance. Table II illustrates that our system achieves significantly lower translation drift and slightly lower rotation drift than the A-LOAM. In the KITTI dataset, our performance is not as good as SDV-LOAM, but SDV-LOAM does not adapt to the *VLP-16* LiDAR and thus fails on the S3E dataset.

For qualitative results, we present a partial view of LiDAR map in Fig 7, which is part of a curved road with only trees around. In this case, A-LOAM suffers degradation while our LiDAR odometry works well. Therefore, both qualitatively and quantitatively, our method outperforms A-LOAM. As for the reasons, A-LOAM lacks constraints on the z-axis, and the loss function easily falls into a minimum value in a degraded scene. Using the results of visual odometry to compensate for the initial value of A-LOAM can reduce the number of iterations and avoid the problem that the loss function falls into a minimum value due to the significant difference between the initial value and the actual value.

V. CONCLUSION AND FUTURE WORK

In this study, we propose a loosely coupled monocular-LiDAR SLAM technique with a novel scale corrector. Its pose prediction derives from monocular odometry with scale correction and LiDAR odometry with visual bootstrapping. Concerning localization performance, our visual odometry achieves better performance than stereo-mode ORB-SLAM2 when loop closure for neither is available, while our LiDAR odometry significantly outperforms baseline A-LOAM [6]. It is illustrated by quantitative results that the whole system yields markedly lower translation drift and moderately lower rotation drift. Qualitative results also show that our system is more robust than A-LOAM [6] in degraded scenes. On the other hand, as for limitations, the proposed system relies heavily on the stability of visual odometry. In other words, a severe drift of visual odometry can cause a great loss of performance to our system, which deserves our deeper investigation.

In our future study, we are expected to refine the proposed framework, including enhancing the robustness of visual odometry through back-end optimization, adding trouble-detection and troubleshooting tragedies for visual odometry failure and involving LiDAR points in constructing visual map.

REFERENCES

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on*

robotics, vol. 31, no. 5, pp. 1147–1163, 2015.

[2] I. Cvišić, I. Marković, and I. Petrović, "Soft2: Stereo visual odometry for road vehicles based on a point-to-epipolar-line metric," *IEEE Transactions on Robotics*, 2022.

[3] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle adjustment in the large," in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II 11*. Springer, 2010, pp. 29–42.

[4] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[5] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*. Springer, 2014, pp. 834–849.

[6] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[7] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-loam: Fast lidar odometry and mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4390–4396.

[8] P. Zhou, X. Guo, X. Pei, and C. Chen, "T-loam: truncated least squares lidar-only odometry and mapping in real time," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–13, 2021.

[9] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "Ct-icp: Real-time elastic lidar odometry with loop closure," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5580–5586.

[10] Z. Liu and F. Zhang, "Balm: Bundle adjustment for lidar mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3184–3191, 2021.

[11] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4802–4809.

[12] G. Kim, B. Park, and A. Kim, "1-day learning, 1-year localization: Long-term lidar localization using scan context image," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1948–1955, 2019.

[13] J. Graeter, A. Wilczynski, and M. Lauer, "Limo: Lidar-monocular visual odometry," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 7872–7879.

[14] Y.-S. Shin, Y. S. Park, and A. Kim, "Direct visual slam using sparse depth for camera-lidar system," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5144–5151.

[15] Z. Zhu, S. Yang, H. Dai, and F. Li, "Loop detection and correction of 3d laser-based slam with visual information," in *Proceedings of the 31st International Conference on Computer Animation and Social Agents*, 2018, pp. 53–58.

[16] X. Liang, H. Chen, Y. Li, and Y. Liu, "Visual laser-slam in large-scale indoor environments," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2016, pp. 19–24.

[17] Z. Liu, Y. Hu, T. Fu, and M.-O. Pun, "Dense three-dimensional color reconstruction with data fusion and image-guided depth completion for large-scale outdoor scenes," in *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*, 2022, pp. 3468–3471.

[18] J. Zhang and S. Singh, "Laser–visual–inertial odometry and mapping with high robustness and low drift," *Journal of field robotics*, vol. 35, no. 8, pp. 1242–1264, 2018.

[19] Z. Yuan, Q. Wang, K. Cheng, T. Hao, and X. Yang, "Sdv-loam: Semi-direct visual-lidar odometry and mapping," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–18, 2023.

[20] C.-C. Chou and C.-F. Chou, "Efficient and accurate tightly-coupled visual-lidar slam," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14 509–14 523, 2021.

[21] Y. Zhu, C. Zheng, C. Yuan, X. Huang, and X. Hong, "Camvox: A low-cost and accurate lidar-assisted visual slam system," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5049–5055.

[22] V. De Silva, J. Roche, and A. Kondoz, "Fusion of lidar and camera sensor data for environment sensing in driverless vehicles," 2017.

[23] J. Mo and J. Sattar, "A fast and robust place recognition approach for stereo visual odometry using lidar descriptors," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5893–5900.

[24] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2174–2181.

[25] N. Ou, H. Cai, J. Yang, and J. Wang, "Targetless extrinsic calibration of camera and low-resolution 3-d lidar," *IEEE Sensors Journal*, vol. 23, no. 10, pp. 10 889–10 899, 2023.

[26] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. Springer, 2006, pp. 430–443.

[27] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, 1981, pp. 674–679.

[28] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1449–1456.

[29] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[30] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.

[31] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991.

[32] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

Difficulty-Aware Time-Bounded Planning under Uncertainty for Large-Scale Robot Missions

Michal Staniaszek, Lara Brudermüller, Raunak Bhattacharyya, Bruno Lacerda, Nick Hawes

***Abstract*— We consider planning problems where a robot must visit a large set of locations to complete a task at each one. Our focus is problems where the *difficulty* of each task, and thus its duration, can be predicted, but not fully known in advance. We propose a general Markov decision process (MDP) model for *difficulty-aware* problems, and propose variants on this model which allow adaptation to different robotics domains. Due to the intractability of the general problem, we propose simplifications to allow planning in large domains, the key being constraining navigation using a solution to the travelling salesperson problem (TSP). We build a set of variant models for two domains with different characteristics: UV disinfection, and cleaning, evaluating them on maps generated from real-world environments. We evaluate the effect of model variants and simplifications on performance, and show that our models outperform a rule-based baseline.**

I. Introduction

Many real-world mobile robot applications such as cleaning, visual inspection, and environmental monitoring involve missions where the robot must execute tasks to *service* each of a set of locations within a time bound.

In this paper, we identify and model a general class of such problems where the duration of actions required to make progress on a specific task is influenced by its *difficulty*. Because the factors which typically make a robotic task difficult, such as environment dynamics or state estimation uncertainty, can only be observed at execution time, we use probabilistic models to predict the difficulty at each location. This yields a problem which requires *planning under uncertainty*. To keep within the time bound, the system must take into account the fact that difficulty affects task duration. Time bounds may be imposed by various sources, such as operational requirements to complete tasks before a certain time, times when humans are in the environment, or weather. We refer to *difficulty-aware* planning problems as those where 1) difficulty of tasks can be modelled probabilistically, 2) task duration depends on difficulty and can also be modelled probabilistically, and 3) a location's task difficulty can be observed online when the robot reaches it.

The class of difficulty-aware problems naturally captures a wide range of current robotics applications: 3D reconstruction of human spaces, where difficulty is due to scene complexity and dynamics [1]; underwater asset inspection, due to uncertain communication and currents moving the vehicle as it captures data [2], [3]; and UV disinfection, due to localisation uncertainty [4], [5].

To solve a difficulty-aware planning problem, the planner has to jointly consider two problems: the order in which to visit locations (ordering), and how much time to spend servicing each location (time allocation). The ordering problem requires the planner to implicitly solve a TSP, so planning for difficulty-aware missions is computationally challenging even for small problems. As originally proposed by Lane and Kaelbling [6], we exploit the fact that these two problem aspects can be decoupled to apply a hierarchical planning approach. We solve the TSP to produce a tour of locations, following that tour while solving the difficulty-aware time allocation problem. Decoupling ordering in this way reduces both the action and state spaces of the MDP and allows our models to scale to much larger problem instances.

Our main contribution is a novel model for difficulty-aware time-bounded planning problems under uncertainty, which allows a wide range of mobile robot missions to be expressed as a reward maximisation problem in an acyclic MDP with a set of terminal states. We present variants of the model to indicate its adaptability to robotics problems, and use state space reduction methods to scale to large problem instances of over 250 tasks, showing that using a TSP to constrain ordering has the greatest effect. We validate our approach through a systematic evaluation of planning and simulated execution in two robotics domains.

II. Related Work

The core of our paper is a time-bounded planning problem under uncertainty. Such problems are commonly formulated as finite-horizon MDPs [7]. We consider a variation where we include time in the state and allow actions to take variable amounts of time. We are specifically interested in problems where reward-generating actions have stochastic duration, but the duration distribution for each action depends on the task difficulty, which is an observable state factor. Prior work has formulated related time-bounded planning problems using MDPs, such as an approach for speeding up the solution of time-bounded planning problems with multiple objectives, minimising time taken while maximising reward [8]. While we do not consider multiple objectives, our model could encode their example and handle larger problem instances. When action duration distributions are not known in advance, a Gaussian process (GP) can be used to maintain a belief over environmental features which influence the distributions, and sampled from when planning [9]. As we assume difficulty is

All authors are with the Oxford Robotics Institute, University of Oxford, Oxford, UK. {michal, larab, raunakbh, bruno, nickh}@robots.ox.ac.uk. This work was supported by the EPSRC Programme Grant "From Sensing to Collaboration" (EP/V000748/1), the UKAEA/EPSRC Fusion Grant (EP/W006839/1), and a gift from Amazon Web Services. 979-8-3503-0704-7/23/$31.00

a discrete set of values which do not influence each other, a GP is overly complex for our needs.

Including uncertainty as a state factor is a way of avoiding formulating problems as a more complex and harder to solve partially observable Markov decision process (POMDP), instead creating an *Augmented MDP* [10], [11]. Those works consider localisation uncertainty, and we represent difficulty a similar way in our UV disinfection example domain. We discretise a distribution on the location variance and use it as a state factor representing different degrees of uncertainty.

Visiting a set of locations under a time-bound has been investigated in the literature on the orienteering problem (OP) [12]. In the OP, the goal is to visit a subset of a given set of locations that provides the maximum reward given a time bound. Probabilistic extensions to the OP have also been investigated where the associated reward with each location is stochastic [13], [14]. Both robotic exploration [15] and persistent monitoring problems [16] have been approached using solutions to the OP. However, our problem differs in that the reward is not obtained simply by visiting each location. It depends on the actions performed, and consequently the time spent, at a location.

Prior work has used a TSP to simplify practical robotics navigation problems by forcing an ordering of tasks [6]. They use a TSP to enforce ordering on a set of navigation macro-actions, considering uncertainty only at the macro-action level, and do not consider time as a factor in the model. We show that the same technique can be useful without the macro action decomposition, and in MDPs which must consider uncertainty in time coming from multiple sources. More formal hierarchical planning methods can be used to reduce the state space of a large MDP by breaking it into sub-problems. Some methods use new algorithms to convert and solve a problem in a hierarchical manner [17], [18], while others use standard MDP solution methods on hand-designed or learned hierarchies [19]–[21]. Both approaches impose an additional burden on the model designer, requiring either implementation of non-standard solution algorithms, or manual design and building of hierarchies. Our models are standard MDPs, with domain-specific simplification approaches.

III. Preliminaries

A *Markov decision process (MDP)* is a tuple $\mathcal{M} = \langle S, \iota, A, T, R, \gamma \rangle$, where S is a finite set of states; ι is a probability distribution over the initial state; A is a finite set of actions; $T : S \times A \times S \to [0, 1]$ is a probabilistic transition function returning the probability of arriving at state s' after taking action a in state s; $R : S \times A \to \mathbb{R}$ a function returning a reward for performing a in state s; and $\gamma \in (0, 1]$ is a discount factor. The aim for an MDP is to find an optimal *policy* $\pi^* : S \to A$ that maximises the expected cumulative discounted reward:

$$\pi^* = \text{argmax}_\pi \mathbb{E}^\pi_{\mathcal{M},\iota}\big[\sum_{i=0}^{\infty} \gamma^i R(s_i, a_i)\big] \tag{1}$$

The optimisation objective may diverge when $\gamma = 1$, but if all runs of the MDP reach a terminal state from which no more reward can be gathered, the objective converges to a finite value regardless. All runs of our model reach a terminal state as mission duration is bounded, so we consider $\gamma = 1$.

IV. Problem Formulation

We consider a mobile robot that navigates in a discrete *topological map*. A topological map is a tuple $\mathcal{T} = \langle V, E, dur_{nav} \rangle$, where $V = \{v_1, \ldots, v_n\}$ is a set of locations in the environment represented by poses of the form (x, y, z, θ) in a global frame; $E \subseteq V \times V$ encodes a set of directed edges the robot can traverse; and $dur_{nav} : E \to \mathbb{N}$ is a function which maps edges to travel durations. The goal is for the robot to navigate around the locations and *service* them by executing an action (e.g. clean or take a sensor reading). The action set available to the robot is to service its current location v, or traverse an edge $(v_c, v') \in E$.

Service actions increase the *service level* at the robot's current location. Service levels are denoted by $l \in \mathbb{N}_L$, where $L \in \mathbb{N}$ and $\mathbb{N}_L = \{0, \ldots, L\}$. These actions have a *difficulty*, modelled as a function $diff : V \to \mathbb{N}_D$, where $D \in \mathbb{N}$, $\mathbb{N}_D = \{0, \ldots, D\}$, which can vary according to the location. This function is unknown a priori, but we have access to a discrete distribution over the difficulty level at each location. For $v \in V$ and $d \in D$, $P(diff(v) = d)$ is the probability of the difficulty at node v being d. There is also a *utility function* $U : V \times \mathbb{N}_L \times \mathbb{N}_L \times \mathbb{N}_D \to \mathbb{R}_{\geq 0}$ such that $U(v, l, l', d)$, known a priori, is the utility of moving the service level at v from level l to level l', $l' > l$, given that the difficulty at v is d.

The difficulty level at a location impacts the *service duration*. We assume a discrete set $\Lambda = \{\lambda_1, \cdots, \lambda_{|\Lambda|}\} \subset \mathbb{N}$ of $|\Lambda|$ representative durations, obtained by discretising a continuous distribution over possible durations. The exact mapping between difficulty and duration is unknown, but we assume that we have access to the probability of the robot taking duration λ to change the service level at v from l to l' under difficulty d, where $v \in V$, $l, l' \in \mathbb{N}_L$ with $l' > l$, $d \in \mathbb{N}_D$ and $\lambda \in \Lambda$. We denote this as $P(dur_{serv}(v, l, l', d) = \lambda)$.

Note that we make two core assumptions, neither of which reduce applicability to practical robotics problems. 1) The deterministic duration set Λ requires that possible durations be known in advance. 2) Difficulty distributions are independent, so difficulty at one location cannot influence that at another.

Mission Goal: Given an initial node $\overline{v} \in V$ and time bound $B \in \mathbb{N}$, find a policy which decides the navigation and servicing actions to execute to maximise the expected sum of gathered utility U, while ensuring the robot returns to $\overline{v}$ within B units of time.

V. Modelling

A. General MDP Model

We start by defining a general MDP model of the time-bounded mission described above. We define $n = |V|$ and, for set X, denote the Cartesian product of X with itself n times as X^n. Given a set of discrete durations Λ and a time

bound B, we define Λ_B^+ as the set of all sums of elements of Λ that are less than or equal to B.

States: The state space is of the form $S = V \times \mathbb{N}_L^n \times \mathbb{N}_D \times \Lambda_B^+$. A state $s = (v_i, l_1, \cdots, l_n, d_i, \tau)$ means that robot is at location v_i, for some $1 \leq i \leq n$; the service level at each location v_j is l_j, for each $1 \leq j \leq n$; the difficulty level at the current location v_i is d_i; and the elapsed time since the start of the mission is τ. The probability of the initial state being $(\bar{v}, 0, \cdots, 0, d, 0)$ is $P(\mathit{diff}(\bar{v}) = d)$, i.e. the robot starts at initial location $\bar{v}$, the service level at every location is initialised as 0 and the mission starts at time 0; the difficulty at $\bar{v}$ is defined according to $P(\mathit{diff}(\bar{v}) = d)$.

Actions and Transitions: In state s, the robot can move between locations using a topological map edge with action a_e. Taking a_e to traverse $e = (v_i, v') \in E$, the current location changes to v', and the time component of the state changes to $\tau' = \tau + dur_{nav}(e)$. The difficulty changes stochastically to d', according to $P(\mathit{diff}(v') = d')$.

The robot can increase the service level at its current location to some $l' > l_i$ with action $a_{l_i,l'}$. Taking service action $a_{l_i,l'}$ with target service level l', the current service level l_i changes to l', and the time component is updated to $\tau' = \tau + \lambda$ stochastically, according to $P(dur_{serv}(v, l_i, l', d) = \lambda)$. Service actions can only increment the service level, such that for all $a_{l_i,l'} \in A, l' = l_i + 1$.

To ensure the robot returns to $\bar{v}$ within time bound B, we disallow actions that have some probability of transitioning to a state from which it is not possible to return to $\bar{v}$ in time. For some action $a \in A$, let $\lambda^a_{\max}$ denote the maximum possible time increment according to the duration model of a; and let $\lambda(v^a, \bar{v})$ be the time to travel to $\bar{v}$ from the location the robot will be at after executing a, according to the shortest path in $\mathcal{T}$. If $\tau + \lambda^a_{\max} + \lambda(v^a, \bar{v}) > B$, then a is not enabled in s. States with no action enabled have a *return to start* action which moves the robot from v_i to $\bar{v}$ by the shortest path in $\mathcal{T}$, and puts the model into a terminal state from which no more reward can be gathered.

Reward Function: The reward function returns a reward based on the utility U, known a priori, of service actions, which may have different utility depending on the difficulty, service level, or location. Other actions do not give a reward.

$$R((v_i, l_1, \ldots, l_n, d_i, \tau), a) = \begin{cases} U(v_i, l_i, l', d_i) & \text{if } a = a_{l_i,l'} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Optimisation Objective: Optimise the expected cumulative reward on the proposed MDP according to Equation 1. We construct the MDP to ensure all policies return to the start node within the time-bound B, after which no more reward can be gathered. Thus, maximising the cumulative reward matches the mission goal stated in the problem formulation.

B. Variants

Our model makes assumptions about how difficulty should be tracked and how the time component is incremented. We now discuss some variations on those assumptions which can be used to model different types of problems.

Endogenous/Exogenous Difficulty: The difficulty of servicing a task can be internal or external to the robot. For example, the difficulty of tasks based on accurate localisation depends on the robot's quantification of its localisation uncertainty, while the difficulty of cleaning an area can be related to the amount of dirt detected there. The former is an example of *endogenous* difficulty and the latter of *exogenous* difficulty. Assuming a static world, once the robot observes a level of exogenous difficulty at a given location, that level remains the same until the end of the mission. Difficulty in endogenous processes changes whenever the robot observes the value of that internal process again. The model in Section V-A assumes endogenous difficulty, so the state only tracks the difficulty value at the current node. Exogenous difficulty can be specified by adding a difficulty state feature for each location, replacing d_i by a set of state factors $\{d_1, \cdots, d_n\}$. This changes the state factor for difficulty from $\mathbb{N}_D$ to $\mathbb{N}_D^n$. The difficulty for each location is initialised to 0, indicating unknown difficulty. When the robot visits the location for the first time, the difficulty is set to the value retrieved from $\mathit{diff}(v)$. After the difficulty for a location is set, it does not change.

Informative Action Durations: In our general model, we assume that the execution of a service action at a node v does not provide extra information on the duration of further service actions on v. However, there are situations where the duration of a service action in a node informs the duration of further actions there. For example, when applying UV radiation to a surface, how long it takes to apply a certain dose depends on the pose of the robot and is stochastic. Once the time taken is observed, the time to apply the dose again in the same pose does not change. In these cases, the value of $P(dur_{serv}(v, l, l', d) = \lambda)$ depends on the duration values previously observed when servicing v. We consider a special case of this, where the observed duration λ of an initial service action determines the duration for subsequent service actions at a location. One additional state factor is required, which starts as 0 to indicate an unknown duration, and is set to λ once a service action is executed and its duration observed. If the uncertainty over the duration is exogenous, it is necessary to keep a factor for each location. The value of the factor is the deterministic duration for further service actions there. In the endogenous case this value is reset when revisiting a location. In the exogenous case, its value can be fixed once it is observed as it is a feature of the environment.

VI. Example Domains

We define two domains to show how our proposed models can be adapted to robotics problems with different properties.

UV disinfection: In the UV disinfection problem, the aim is to use a robot to disinfect a series of locations by irradiating surfaces with UVC light in the 100-280nm range to apply a dose which will achieve log reductions in microbial activation [4]. A 1-log reduction indicates 90% inactivation of a microbial colony, 2-log 99%, and so on.

We assign four service levels, 0 indicating the location has not been cleaned, the other levels corresponding to 1,

2, and 3-log reductions in activation. The dose applied to a surface is proportional to the inverse square of distance to the UV source, which means that small variations in the robot's position can have a significant effect. As such, we define the difficulty as the robot's metric localisation uncertainty, discretised into three levels representing high (0), medium (1), and low (2) confidence in its location.

To generate difficulty distributions $\mathit{diff}(v)$ we use empirical localisation uncertainty data from navigation of a Scitos X3 robot in Oxfordshire County Library (the library map in Fig. 1). We cluster the data using a Gaussian mixture model for each confidence level. A location's distribution is based on how often the localisation uncertainty there is matched by each mixture model. This data is used over all maps by randomly sampling from the set of location distributions.

The service duration distribution dur_{serv} is generated by sampling poses from the GMM and using an irradiation simulation to determine the duration required to apply the dose required to reach service level 1 for each difficulty. We fit a categorical distribution to these samples with a discretisation of 5s. As the service levels are log-linear, we can multiply the durations for $dur_{serv}(v, 0, 1, d)$ by two to get the distribution for level 2, and multiply by 4 to get the durations for level 3. This domain has rewards with diminishing returns, to encourage higher coverage. 100 reward is given for service level 1, 50 for 2, and 25 for 3. For further details, see the supplementary material[1] or our previous work [22].

Cleaning: In the cleaning domain, the robot must fully clean floors in various locations, which may have different sizes and dirtiness levels. There are only two service levels, as a location is either clean or not clean. A location's dirtiness corresponds to the difficulty, affecting how long the robot takes to clean it. We track the difficulty for each location, as it is exogenous. A location can have no dirt (0), or low (1), medium (2) or heavy (3) dirt. The robot only knows how dirty a location is after visiting it, and could be determined using computer vision or human input. During model building $\mathit{diff}(v)$ for each location is sampled from a set of 3 artificially generated distributions by uniform random selection. Each location is randomly designated as a small, medium, or large room. dur_{serv} is deterministic, found by multiplying the duration for the difficulty by the location's size. For servicing a location with difficulty d, the system receives $100d$ reward.

VII. State Space Reduction Strategies

As our model has state factors which depend on the number of locations, the resulting MDP is impractical to solve for anything more than trivial problems. Combining simple strategies can greatly reduce the state space.

A. Fixed Navigation

The most drastic state space reduction comes from limiting navigation on the topological map to a tour generated by solving a TSP [6]. Let $V_{next} : V \to V$ be the mapping from a location to the next location to visit. It is defined by a TSP tour constrained to the topological graph, generated by an optimal TSP solver [23], which visits all locations. This prunes possible edge transition actions a_e at location v_i by making the only valid edge $e = (v_i, V_{next}(v_i))$. It can be the case that $(v_i, V_{next}(v_i)) \notin E$, as to get from v_i to $V_{next}(v_i)$ requires traversing through an intermediate location in the topological map. We redefine $dur_{nav} : V \times V \to \mathbb{N}$ to give the shortest duration path between any two locations. This simulates a fully connected map and means that a_e can be used to jump directly from one location to another without having to traverse individual edges along the shortest path. The tour takes time B_{TSP}, based on the sum of dur_{nav} for the traversed edges. The remaining budget to service all locations is $B_{MDP} = B - B_{TSP}$. By pruning actions according to V_{next}, it is impossible for the model to return to any previous location, so the state only needs to track state factors relevant for the current location. The service level state factors for all nodes go from $\mathbb{N}_L^n$ to $\mathbb{N}_L$. The state $s = (v_i, l_1, \cdots, l_n, d_i, \tau)$ becomes $s = (v_i, l_i, d_i, \tau)$, greatly reducing the number of dimensions in the Cartesian product.

B. Single Service Action

We change the action $a_{l,l'}$ such that l must be 0. This compresses actions for incrementally increasing the service level at a location into a single action to go from service level 0 to any service level l'.

The probability distributions $P(dur_{serv}(v, l, l', d))$ for incremental actions must be collapsed to represent a transition from service level 0 to l' without reaching intermediate service levels. This can be done by taking the product of the probability distributions for each incremental service level, i.e. $P(dur_{serv}(v, 0, l', d)) = \prod_{l=0}^{l'-1} P(dur_{serv}(v, l, l+1, d))$. This leads to a reduction of choices available at each location. With incremental actions there is a choice at each service level increment to either continue servicing or leave. With single actions, it is only necessary to choose which (if any) service level to achieve, without having to make the same decision again at each increment. This is limiting, as the policy cannot choose to abort servicing early if the service time observed from $dur_{serv}(v, l, l', d))$ is unfavourable, and if service times are favourable it cannot exploit the extra time to return to locations for further servicing.

The state factor for the service level at each location l_i, which contributes $\mathbb{N}_L^n$ to the state space, can become Boolean $\mathbb{N}_L = \{0, 1\}$ instead of an integer value $\mathbb{N}_L = \{0, 1, \ldots, L\}$, greatly reducing the model's state space. The outcome of any action $a_{0,l'}$ is that $l_i = 1$. The simplification relies on implicit encoding of the service level l' in the action. We maintain the reward by summing values of U for each increment.

VIII. Experimental Evaluation

We will refer to models based on which variants and reduction strategies they use: incremental (I) or single (S) actions; stochastic (ST), informative action duration (AD), or deterministic (D) service duration distributions (in the deterministic case, for a given level of difficulty, $dur_{serv}(v, l, l', d)$ has a single outcome $\lambda \in \Lambda$ with probability 1); and free (FN)

[1] robots.ox.ac.uk/~michal/papers/difficulty-aware-supp.pdf

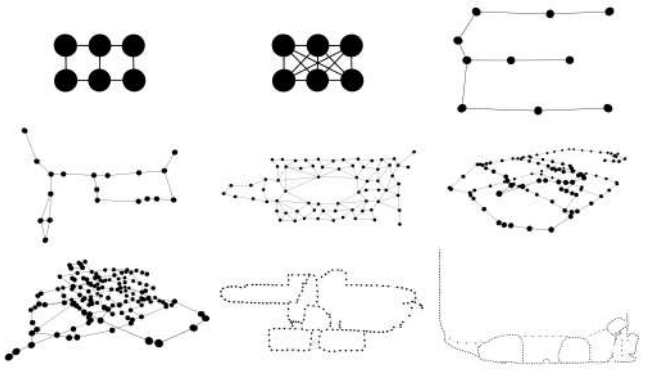


Fig. 1. Topological maps used in experiments. From left to right and top to bottom: tiny (6 locations), tiny_fc (6), warehouse (10), foyer (19), library (70), hall (107), oilrig (138), plant (172), fence (257). See supplementary material for examples of environment pointclouds.

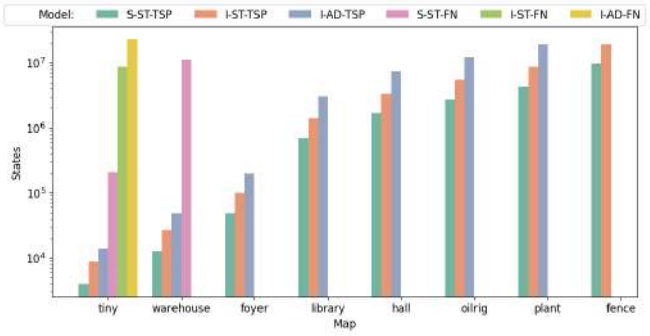


Fig. 2. Number of states in MDPs for the UV domain across topological maps of different sizes. Models which we are unable to solve due to memory constraints are omitted. B for each map is defined by $\alpha_2 = \alpha_3 = 0.5$. tiny_fc is omitted as it is identical to tiny, except I-AD-FN is unsolvable.

or fixed (TSP) navigation. For example, the model described in Section V-A has incremental service actions, stochastic action transitions, and free navigation, so is written as I-ST-FN. For the UV domain there are six models with endogenous difficulty. Single action and informative action duration models are mutually exclusive, so we have three free (I-AD-FN, I-ST-FN, S-ST-FN), and three fixed navigation models (I-AD-TSP, I-ST-TSP, S-ST-TSP). The cleaning domain has only a single service level, and its service duration distribution is deterministic, so it has two models, S-D-FN and S-D-TSP, with exogenous difficulty. Models are evaluated by solving their MDP with the PRISM model checker [24], which generates an optimal policy using value iteration. All models were solved using 64GB RAM, 16GB of swap space, and an Intel Core i7-8700 CPU at 3.20GHz. Models are evaluated on the maps in Fig. 1, with all except the tiny map generated from visits to real environments.

A. Domain Baselines

We compare our models to a rule-based *baseline behaviour* to quantify performance and highlight the benefits of planning.

1) UV Baseline: Allocates uniform service duration $\beta = B_{MDP}/|V|$ to all locations in the map, and follows the TSP tour. With no stochasticity in difficulty or time, constant service time per node should always reach the same service level. In practice this is not the case and the level reached at different locations will vary.

2) Cleaning Baseline: Follows the TSP tour, greedily cleaning each location for the deterministic service time computed from its size and dirtiness, while ensuring it can return to the start location within the time bound.

B. Time Bound Selection

Models are evaluated with three different time bounds $B = B_{TSP} + B_{MDP}$. B_{MDP} is defined by α_1, α_2, and α_3, where $\sum_i \alpha_i = 1$, as described below. The purpose of the α values is to generate time bounds proportional to the number of nodes in a map, and to use knowledge of the service levels and expected times to define how constraining the bound is on the service level and number of nodes that can be serviced.

1) UV Time Bound: Values of α_s correspond to what proportion of locations should be serviced to level s in the ideal case, e.g. if $\alpha_1 = 0.5$, and $\alpha_2 = 0.5$ we expect half of nodes should be serviced to level 1 and half to level 2. t_s is the time to reach level s at all locations assuming a deterministic setting. The bound is then $B_{MDP} = \sum_{s=1}^{3} \alpha_l t_l$.

2) Cleaning Time Bound: The time to service a location depends on the difficulty. We compute the expected number of locations for each difficulty level, and the expected time to clean a location with a specific difficulty regardless of the size of the location. The dot product of the resulting vectors gives the total expected time $\mathbb{E}(t_s)$ to clean the expected number of nodes at each difficulty. The bound is then $B_{MDP} = \sum_{s=1}^{3} \alpha_s \mathbb{E}(t_s)$, with values of α controlling the bound according to the proportion of locations each difficulty we might expect to be serviced, e.g. setting $\alpha_1 = 0.5$ and $\alpha_2 = 0.5$, the model should have enough time to clean half of locations with low dirt, and half with medium dirt.

C. Results

1) State Space and Solution Time: Figure 2 shows the states for models across all maps in the UV domain. The number of states required to represent the I-AD-FN on the smallest map is larger than that required for S-ST-TSP on the largest map. We were unable to generate solutions for any free navigation models for maps larger than 19 nodes due to memory limits. In the cleaning domain, with the largest time bound on the tiny map, S-D-FN has around 1.02×10^7, while S-D-TSP has 457. Even in the 257 node map with the largest bound S-D-TSP has only 1.17×10^6 states. The growth of the state space for free navigation models appears approximately log-linear in the number of nodes, but also depends on the average degree of the topological map.

TABLE I

SOLUTION TIME IN SECONDS FOR FIXED NAVIGATION MODELS ACROSS SELECTED MAPS WITH HORIZON $\alpha_2 = \alpha_3 = 0.5$.

Model	foyer	library	hall	oilrig	plant	fence
I-AD-TSP	14	773	2767	6378	16431	N/A
I-ST-TSP	4	125	555	974	1956	7540
S-ST-TSP	2	74	269	570	1181	4074

Table I shows solution time across selected maps. The number of states is closely related to solution time of the MDP. In the map with 6 locations, for the longest horizon PRISM required 30m to solve I-AD-FN, the most complex model, but less than 0.2s to solve S-ST-TSP, the simplest. The disparity between the fixed navigation models becomes clearer on the library map, where the horizon is around 50 minutes. For the largest map of 257 locations, with the time bound defining 3 hours of robot operation, solution time for S-ST-TSP is a little over an hour. Computation time on the tiny_fc map for free navigation models is over twice that of the same models on the tiny map, as a result of increased choices and transitions in the MDP.

2) Service levels: We evaluate our models by executing the policies they generate in a discrete event simulator which evolves the state according to the dynamics of a model without simplifications. Any variation in reward should show the effects of the simplifications which do not directly map to the behaviour of the world represented by the unsimplified model. To represent the world, we use the *world model* I-AD-FN for the UV domain, and S-D-FN for the cleaning domain, as they are closest to how we expect the real world to evolve. The simulator does not require solution of the MDP of the world model, so we are able to run simulations of even the largest maps. We run policies generated from other models on the simulator. We can only compare all models on the tiny map as we are unable to solve the full set of free navigation models on larger maps due to memory constraints.

Fig. 3 shows the difference in service levels achieved between the baseline and models. In the UV domain, models always achieve a higher average service level across all nodes, and are much closer to achieving the service levels we would expect based on α. Compared to the rule-based baseline, planning always achieves higher average service levels across all nodes in the UV domain. In a deterministic setting, with time bound from $\alpha_1 = 1$, the baseline should always reach service level 1, but the majority of locations are not serviced as the time allocated is insufficient. Planning achieves cleaning level 1 in almost all locations.

The cleaning domain has only two service levels, so we show the average number of nodes at each dirtiness level after the policy has been executed (Fig. 4). The baseline and models have access to the same service duration information, but after policy execution the S-D-TSP model receives more reward on average for shorter time bounds. As the time bound increases, there is less difference between planning and the baseline, as slack in the budget allows the baseline to gather reward at every location regardless of uncertainty.

3) Rewards: Table II shows the mean reward for various models on the library map for the UV and cleaning domains over 1000 simulations. Our models always outperform the baselines, even in the cleaning domain with deterministic service action duration. On the 6 node map, mean rewards across all models are within approximately 1.5% of each other. This is also the case for the subsets of models we can solve for larger maps. This indicates that for small maps the state reduction strategies enable scalability while retaining

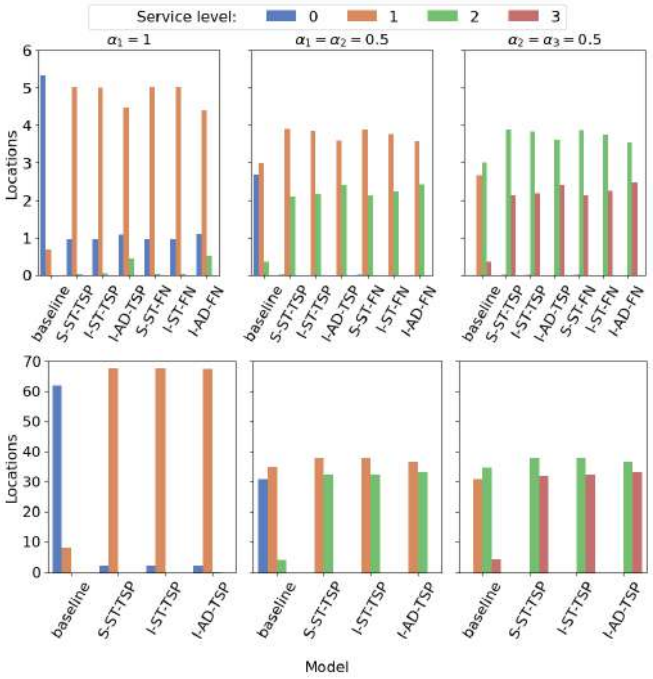


Fig. 3. Average number of locations at specific service levels achieved in the UV domain on the tiny (top) and library (bottom) maps. Other maps have similar results. 1000 policy executions per model on each time bound as specified by α values. Standard error is less than 1% in all cases.

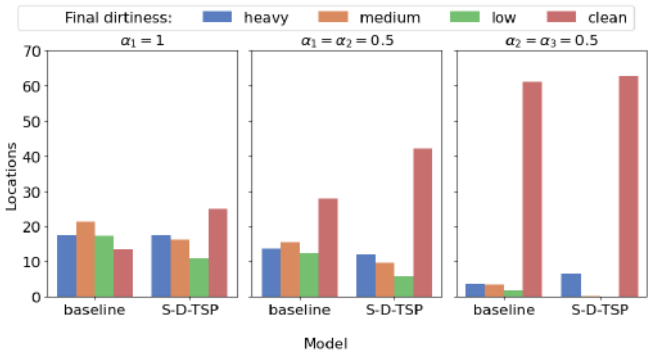


Fig. 4. Average number of locations with specific dirtiness after policy execution in the cleaning domain by baseline and S-D-TSP model on the library map. 1000 policy executions per model for each time bound specified by α values. Standard error is less than 1% in all cases.

performance. We performed this evaluation with uniform and non-uniform probability distributions to test whether more uncertainty has a large effect. Uniform distributions show a slight increase in variation, but all models are still within approximately 2% of each other. The result of simulating policies gives very similar reward values to the expected policy value from PRISM. Over 1000 policy executions, the mean reward for all models in simulation is within 1% of the expected value of the policy, with the largest standard deviation equivalent to a difference of fully servicing at most two locations. On the tiny map, the only one where we can run all models, we see very little difference in reward values for fixed and free navigation variations. This is likely due to the simplicity of the map, where there is little advantage to free navigation models in making choices about navigation.

TABLE II
TOTAL REWARD ON THE LIBRARY MAP OVER 1000 POLICY EXECUTIONS.

UV	$\alpha_1 = 1$	$\alpha_1 = \alpha_2 = 0.5$	$\alpha_2 = \alpha_3 = 0.5$
Baseline	804 ± 261	4104 ± 211	9058 ± 107
S-ST-TSP	6781 ± 78	8608 ± 65	11302 ± 32
I-ST-TSP	6783 ± 75	8609 ± 63	11305 ± 34
I-AD-TSP	6784 ± 79	8662 ± 67	11329 ± 30

Cleaning	$\alpha_1 = 1$	$\alpha_1 = \alpha_2 = 0.5$	$\alpha_2 = \alpha_3 = 0.5$
Baseline	2666 ± 169	5534 ± 298	12094 ± 511
S-D-TSP	4615 ± 117	8092 ± 188	12151 ± 302

On the warehouse map, the mean reward for the S-ST-FN model over 1000 simulations for the shortest time bound was 1088, while the three fixed navigation models were between 905 and 912. For the foyer map, solving the MDP for the shortest time bound took 32 hours. The expected reward value was 1873, as opposed to the fixed navigation models having reward of approximately 1790. Both of these differences are significant according to a two-sided Kolmogorov-Smirnov test, which further indicates that models with free navigation may be able to get more reward on larger maps. Comparing results on the tiny maps, connectivity also affects reward distribution, with free navigation models performing up to 2% better for some models. We hypothesise that this difference may be clearer on larger maps, but we are unable to test this due to memory constraints on the free navigation models.

IX. CONCLUSIONS

We propose a general MDP for robot task planning under uncertainty which considers task difficulty, and apply simplifications to solve it for large topological maps in two representative robotics domains. We greatly reduce the state space of the MDP by constraining the navigation with a TSP, decoupling the ordering and time allocation problems. On small maps, this simplification results in similar expected reward to policies generated by unsimplified models. We show that our models always outperform a rule-based baseline. Unconstrained navigation with the general model should outperform simplified models on larger maps, but even with our simplifications, evaluating this will require different solution approaches.

Future work will investigate the use of approximate techniques such as Monte Carlo tree search [25] and labeled real-time dynamic programming [26], or further simplification through hierarchical planning methods to find solutions. We aim to implement our models on a physical robot, where the time bound might be set using expected battery life. A potential difficulty is that real world task execution times may not easily map onto our assumption of a fixed set of possible task durations.

REFERENCES

[1] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb, "State of the Art on 3D Reconstruction with RGB-D Cameras," *Computer Graphics Forum*, May 2018.

[2] M. Budd, G. Salavasidis, I. Karnarudzaman, C. A. Harris, A. B. Phillips, P. Duckworth, N. Hawes, and B. Lacerda, "Probabilistic Planning for AUV Data Harvesting from Smart Underwater Sensor Networks," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[3] G. A. Hollinger, A. A. Pereira, J. Binney, T. Somers, and G. S. Sukhatme, "Learning Uncertainty in Ocean Current Predictions for Safe and Reliable Navigation of Underwater Vehicles: Learning Uncertainty in Ocean Current Predictions," *Journal of Field Robotics*, 2016.

[4] A. Pierson, J. W. Romanishin, H. Hansen, L. Z. Ya, and D. Rus, "Designing and Deploying a Mobile UVC Disinfection Robot," in *IROS*, 2021, p. 2.

[5] L. Brudermüller, R. Bhattacharyya, B. Lacerda, and N. Hawes, "Time-bounded large-scale mission planning under uncertainty for uv disinfection," in *ICAPS '22 PlanRob Workshop*, 2022.

[6] T. Lane and L. P. Kaelbling, "Approaches to macro decompositions of large markov decision process planning problems," in *Mobile Robots XVI*, 2002.

[7] Mausam and A. Kolobov, *Planning with Markov Decision Processes: An AI Perspective*, 2012.

[8] B. Lacerda, D. Parker, and N. Hawes, "Multi-objective policy generation for mobile robots under probabilistic time-bounded guarantees," in *ICAPS*, 2017.

[9] P. Duckworth, B. Lacerda, and N. Hawes, "Time-Bounded Mission Planning in Time-Varying Domains with Semi-MDPs and Gaussian Processes," Tech. Rep., 2020.

[10] N. Roy and S. Thrun, "Coastal navigation with mobile robots," in *NeurIPS*, 2000.

[11] L. Nardi and C. Stachniss, "Uncertainty-aware path planning for navigation on road networks using augmented MDPs," in *ICRA*, 2019.

[12] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering Problem: A survey of recent variants, solution approaches and applications," 2016.

[13] E. Angelelli, C. Archetti, C. Filippi, and M. Vindigni, "The probabilistic orienteering problem," *Computers & Operations Research*, 2017.

[14] ——, "A dynamic and probabilistic orienteering problem," *Computers & Operations Research*, 2021.

[15] O. Peltzer, A. Bouman, S.-K. Kim, R. Senanayake, J. Ott, H. Delecki, M. Sobue, M. J. Kochenderfer, M. Schwager, J. Burdick *et al.*, "Fig-op: Exploring large-scale unknown environments on a fixed time budget," in *IROS*, 2022.

[16] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to persistent monitoring tasks," *IEEE Transactions on Robotics*, 2016.

[17] J. L. Barry, L. P. Kaelbling, and T. Lozano-Perez, "Deth*: Approximate hierarchical solution of large markov decision processes," in *ICJAI*, 2011.

[18] B. Bakker, Z. Zivkovic, and B. Kröse, "Hierarchical dynamic programming for robot path planning," in *IROS*. IEEE, 2005, pp. 2756–2761.

[19] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean, and C. Boutilier, "Hierarchical solution of markov decision processes using macro-actions," in *Uncertainty in Artificial Intelligence*, 1998.

[20] T. G. Dietterich, "Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition," *JAIR*, 2000.

[21] N. Gopalan, M. L. Littman, J. MacGlashan, S. Squire, S. Tellex, J. Winder, L. L. Wong *et al.*, "Planning with abstract markov decision processes," in *ICAPS*, 2017.

[22] L. Brudermüller, R. Bhattacharyya, B. Lacerda, and N. Hawes, "Time-bounded large-scale mission planning under uncertainty for uv disinfection," in *ICAPS 2022 Workshop on Planning and Robotics (PlanRob)*, Jun. 2022.

[23] D. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, "Concorde tsp solver," https://www.math.uwaterloo.ca/tsp/concorde/index.html.

[24] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *International Conference on Computer Aided Verification*, 2011.

[25] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Computers and Games*, 2007.

[26] B. Bonet and H. Geffner, "Labeled rtdp: Improving the convergence of real-time dynamic programming." in *ICAPS*, 2003.

Robust Multi-Agent Pickup and Delivery with Delays

Giacomo Lodigiani[1], Nicola Basilico[2], and Francesco Amigoni[1]

***Abstract*— Multi–Agent Pickup and Delivery (MAPD) is the problem of computing collision-free paths for a group of agents such that they can safely reach delivery locations from pickup ones. These locations are provided at runtime, making MAPD a combination between classical Multi–Agent Path Finding (MAPF) and online task assignment. Current algorithms for MAPD do not consider many of the practical issues encountered in real applications: real agents often do not follow the planned paths perfectly, and may be subject to delays and failures. In this paper, we study the problem of MAPD with *delays*, and we present two solution approaches that provide robustness guarantees by planning paths that limit the effects of imperfect execution. In particular, we introduce two algorithms, k–TP and p–TP, both based on a decentralized algorithm typically used to solve MAPD, Token Passing (TP), which offer deterministic and probabilistic guarantees, respectively. Experimentally, we compare our algorithms against a version of TP enriched with online replanning. k–TP and p–TP provide robust solutions, significantly reducing the number of replans caused by delays, with little or no increase in solution cost and running time.**

I. Introduction

In Multi–Agent Pickup and Delivery (MAPD) [1], a set of agents must jointly plan collision–free paths to serve pickup–delivery tasks that are submitted at runtime. MAPD combines a task-assignment problem, where agents must be assigned to pickup–delivery pairs of locations, with Multi–Agent Path Finding (MAPF) [2], where collision–free paths for completing the assigned tasks must be computed. A particularly challenging feature of MAPD problems is that they are meant to be cast into dynamic environments for long operational times. In such settings, tasks appear at any time in an online fashion.

Despite studied only recently, MAPD has a great relevance for a number of real–world application domains. Automated warehouses, where robots continuously fulfill new orders, arguably represent the most significant industrial deployments [3]. Beyond logistics, MAPD applications include also the coordination of teams of service robots [4] or fleets of autonomous cars, and the automated control of non–player characters in video games [5].

Recently, the MAPF community has focused on resolution approaches that can deal with real–world–induced relaxations of some idealistic assumptions usually made when defining the problem. A typical example is represented by the assumption that the planned paths are executed without errors. In reality, execution of paths might be affected by delays and other issues that can hinder some of their expected properties (e.g., the absence of collisions). One approach is to add online adaptation to offline planning, in order to cope with situations where the path execution incurs in errors [6]. Despite being reasonable, this approach is not always desirable in real robotic applications. Indeed, replanning can be costly in those situations where additional activities in the environment are conditioned to the plans the agents initially committed to. In other situations, replanning cannot even be possible: think, as an example, to a centralized setting where robots are no more connected to the base station when they follow their computed paths. This background motivated the study of *robustness* [1], [7], [8], generally understood as the capacity, guaranteed at planning time, of agents' paths to withstand unexpected runtime events. In our work, we focus on robustness in the long–term setting of MAPD, where it has not been yet consistently studied.

Specifically, in this paper, we study the robustness of MAPD to the occurrence of *delays*. To do so, we introduce a variant of the problem that we call *MAPD with delays* (*MAPD–d* for short). In this variant, like in standard MAPD, agents must be assigned to tasks (pickup–delivery locations pairs), which may continuously appear at any time step, and collision–free paths to accomplish those tasks must be planned. However, during path execution, delays can occur at arbitrary times causing one or more agents to halt at some time steps, thus slowing down the execution of their planned paths. We devise a set of algorithms to compute robust solutions for MAPD–d. The first one is a baseline built from a decentralized MAPD algorithm, Token Passing (TP), to which we added a mechanism that replans in case collisions caused by delays are detected when following planned paths. TP is able to solve well–formed MAPD problem instances [9], and we show that, under some assumptions, the introduction of delays in MAPD–d does not affect well–formedness. We then propose two new algorithms, k–TP and p–TP, which adopt the approach of robust planning, computing paths that limit the risk of collisions caused by potential delays. k–TP returns solutions with deterministic guarantees about robustness in face of delays (k–robustness), while solutions returned by p–TP have probabilistic robustness guarantees (p–robustness). We compare the proposed algorithms by running experiments in simulated environments and we evaluate the trade–offs offered by different levels and types of robustness.

In summary, the main contributions of this paper are: the introduction of the MAPD–d problem and the study of some of its properties (Section III), the definition of two algorithms (k–TP and p–TP) for solving MAPD–d

[1]Giacomo Lodigiani and Francesco Amigoni are with the Department of Electronics, Information, and Bioengineering (DEIB), Politecnico di Milano, Milan, Italy giacomo.lodigiani@mail.polimi.it, francesco.amigoni@polimi.it

[2]Nicola Basilico is with the Department of Computer Science, University of Milan, Milan, Italy nicola.basilico@unimi.it

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

problems with robustness guarantees (Section IV), and their experimental evaluation that provides insights about how robustness and solution cost can be balanced (Section V).

II. Preliminaries and Related Work

In this section, we discuss the relevant literature related to our work and we introduce the formal concepts we build upon in the following sections.

A basic MAPF problem assigns a start–goal pair of vertices on a graph $G = (V, E)$ to each agent from a set $A = \{a_1, a_2, \ldots, a_\ell\}$ and is solved by a minimum–cost discrete–time set of paths allowing each agent to reach its goal without collisions [2]. We shall define agent a_i's *path* as $\pi_i = \langle \pi_{i,t}, \pi_{i,t+1}, \ldots, \pi_{i,t+n} \rangle$, namely a finite sequence of vertices $\pi_{i,h} \in V$ starting at some time t and ending at $t+n$. Following π_i, the agent must either move to an adjacent vertex ($(\pi_{i,t}, \pi_{i,t+1}) \in E$) or not move ($\pi_{i,t+1} = \pi_{i,t}$).

MAPD extends the above one–shot setting to a time–extended setting by introducing tasks $\tau_j \in \mathcal{T}$, each specifying a pickup and a delivery vertex denoted as s_j and g_j, respectively. A task has to be assigned to an agent that must execute it following a collision–free path from its initial location to s_j and then from s_j to g_j. A peculiar characteristic of this problem is that the set $\mathcal{T}$ is filled at runtime: a task can be added to the system at any (finite) time and from the moment it is added it becomes assignable to any agent. An agent is *free* when it is currently not executing any task and *occupied* when it is assigned to a task. If an agent is free, it can be assigned to any task $\tau_j \in \mathcal{T}$, with the constraint that a task can be assigned to only one agent. When this happens, the task is removed from $\mathcal{T}$ and, when the agent completes its task eventually arriving at g_j, it returns free. A *plan* is a set of paths, which are required to be *collision–free*, namely any two agents cannot be in the same vertex or traverse the same edge at the same time. Each action (movement to an adjacent vertex or wait) lasts one time step. Solving MAPD means finding a minimum–cost plan to complete all the tasks in $\mathcal{T}$. Cost usually takes one of two possible definitions. The *service time* is the average number of time steps needed to complete each task τ_j, measured as the time elapsed from τ_j's arrival to the time an agent reaches g_j. The *makespan*, instead, is the earliest time step at which all the tasks are completed. Being MAPD a generalization of MAPF, it is NP–hard to solve optimally with any of the previous cost functions [10], [11].

Recent research focused on how to compute solutions of the above problems which are robust to delays, namely to runtime events blocking agents at their current vertices for one or more time steps, thus slowing down the paths execution. The MAPF literature provides two notions of robustness, which we exploit in this paper. The first one is that of k–robustness [8], [12]. A plan is k–robust iff it is collision–free and remains so when at most k delays for each agent occur. To create k–robust plans, an algorithm should ensure that, when an agent leaves a vertex, that vertex is not occupied by another agent for at least k time steps. In this way, even if the first agent delays k times, no collision can occur. The second one is called p–robustness [7]. Assume that a fixed probability p_d of any agent being delayed at any time step is given and that delays are independent of each other. Then, a plan is p–robust iff the probability that it will be executed without a collision is at least p. Differently from k–robustness, this notion provides a probabilistic guarantee.

Robustness for MAPD problems has been less studied. One notion proposed in [9] and called *long–term robustness* is actually a *feasibility* property that guarantees that a finite number of tasks will be completed in a finite time. Authors show how a sufficient condition to have long–term robustness is to ensure that a MAPD instance is *well–formed*. This amounts to require that (i) the number of tasks is finite; (ii) there are as much non-task endpoints as agents, where non-task endpoints are vertices designated as rest locations at which agents might not interfere with any other moving agent; (iii) for any two (task or non-task) endpoints, there exists a path between them that traverses no other endpoints.

In this work, we leverage the above concepts to extend k– and p–robustness to long–term MAPD settings. To do so, we focus on a current state–of–the–art algorithm for MAPD, Token Passing (TP) [9]. This algorithm follows an online and decentralized approach that, with respect to the centralized counterparts, trades off optimality to achieve an affordable computational cost in real–time long–term settings. We report it in Algorithm 1. The *token* is a shared block of memory containing the current agents' paths $\{\pi_i\}$, the current task set $\mathcal{T}$, and the current assignment of tasks to the agents. The token is initialized with paths in which each agent a_i rests at its initial location $loc(a_i)$ (line 1). At each time step, new tasks might be added to $\mathcal{T}$ (line 3). When an agent has reached the end of its path in the token, it becomes free and requests the token (at most once per time step). The token is sent in turn to each requesting agent (line 5) and the agent with the token assigns itself (line 9) to the task τ in $\mathcal{T}$ whose pickup vertex is closest to its current location (line 8), provided that no other path already planned (and stored in the token) ends at the pickup or delivery vertex of such task (line 6). The distance between the current location $loc(a_i)$ of agent a_i and the pickup location s_j of a task is calculated using a (possibly approximated) function h (for the grid environments of our experiments we use the Manhattan distance). The agent then computes a collision–free path from its current position to the pickup vertex, then from there to the delivery vertex, and finally it eventually rests at the delivery vertex (line 11). Finally, the agent releases the token (line 17) and everybody moves one step on its path (line 19). If a_i cannot find a feasible path it stays where it is (line 13) or it calls the function *Idle* to compute a path to a non-task endpoint in order to ensure long–term robustness (line 15).

Note that other dynamic and online settings, different from ours, have been considered for MAPF and MAPD. For example, [13] introduces a setting in which the set of agents is not fixed, but agents can enter and leave the system, [14] proposes an insightful comparison of online algorithms that can be applied to the aforementioned setting, and [15] studies

Algorithm 1: Token Passing

initialize *token* with path $\langle loc(a_i)\rangle$ for each agent a_i ($loc(a_i)$ is a_i's current (eventually initial) location);
while *true* **do**
  add new tasks, if any, to the task set $\mathcal{T}$;
  while *agent* a_i *exists that requests token* **do**
    /* *token* assigned to a_i and a_i executes now */;
    $\mathcal{T}' \leftarrow \{\tau_j \in \mathcal{T} \mid$ no path in *token* ends in s_j or $g_j\}$;
    if $\mathcal{T}' \neq \{\}$ **then**
      $\tau \leftarrow \arg\min_{\tau_j \in \mathcal{T}'} h(loc(a_i), s_j)$;
      assign a_i to τ;
      remove τ from $\mathcal{T}$;
      update a_i's path in *token* with the path returned by *PathPlanner*$(a_i, \tau, token)$;
    else if *no task* $\tau_j \in \mathcal{T}$ *exists with* $g_j = loc(a_i)$ **then**
      update a_i's path in *token* with the path $\langle loc(a_i)\rangle$;
    else
      update a_i's path in *token* with *Idle*$(a_i, token)$;
    end
    /* a_i returns *token* to system */;
  end
  agents move on their paths in *token* for one time step;
end

a related problem where the actions have uncertain costs.

III. MAPD with Delays

Delays are typical problems in real applications of MAPF and MAPD and may have multiple causes. For example, robots can slow down due to some errors occurring in the sensors used for localization and coordination [16]. Moreover, real robots are subject to physical constraints, like minimum turning radius, maximum velocity, and maximum acceleration, and, although algorithms exists to convert time–discrete MAPD plans into plans executable by real robots [17], small differences between models and actual agents may still cause delays. Another source of delays is represented by anomalies happening during path execution and caused, for example, by partial or temporary failures of some agent [18].

We define the problem of *MAPD with delays* (*MAPD–d*) as a MAPD problem (see Section II) where the execution of the computed paths π_i can be affected, at any time step t, by delays represented by a time–varying set $\mathcal{D}(t) \subseteq A$. Given a time step t, $\mathcal{D}(t)$ specifies the subset of agents that will delay the execution of their paths, lingering at their currently occupied vertices at time step t. An agent could be delayed for several consecutive time steps, but not for indefinitely long in order to preserve well–formedness (see next section). The temporal realization of $\mathcal{D}(t)$ is unknown when planning paths, so a MAPD–d instance is formulated as a MAPD one: no other information is available at planning time. The difference lies in how the solution is built: in MAPD–d we compute solutions accounting for robustness to delays that might happen at runtime.

More formally, delays affect each agent's execution trace. Agent a_i's *execution trace* $e_i = \langle e_{i,0}, e_{i,1}, ..., e_{i,m}\rangle$[1] for a given path $\pi_i = \langle \pi_{i,0}, \pi_{i,1}, \ldots, \pi_{i,n}\rangle$ corresponds to the actual sequence of m ($m \geq n$) vertices traversed by a_i while following π_i and accounting for possible delays. Let us call $idx(e_{i,t})$ the index of $e_{i,t}$ (the vertex occupied by a_i at time step t) in π_i. Given that $e_{i,0} = \pi_{i,0}$, the execution trace is defined, for $t > 0$, as:

$$e_{i,t} = \begin{cases} e_{i,t-1} & \text{if } a_i \in \mathcal{D}(t) \\ \pi_{i,h} \mid h = idx(e_{i,t-1}) + 1 & \text{otherwise} \end{cases}.$$

An execution trace terminates when $e_{i,m} = \pi_{i,n}$ for some m.

Notice that, if no delays are present (that is, $\mathcal{D}(t) = \{\}$ for all t) then the execution trace e_i exactly mirrors the path π_i and, in case this is guaranteed in advance, the MAPD–d problem becomes *de facto* a regular MAPD problem. In general, such a guarantee is not given and solving a MAPD–d problem opens the issue of computing collision–free tasks–fulfilling MAPD paths (optimizing service time or makespan) characterized by some level of robustness to delays.

The MAPD–d problem reduces to the MAPD problem as a special case, so the MAPD–d problem is NP-hard.

A. Well-formedness of MAPD–d

In principle, if a problem instance is well–formed, delays will not affect its feasibility (this property is also called long–term robustness, namely the guarantee that a finite number of tasks will be completed in a finite time, see Section II). Indeed, well–formedness is given by specific topological properties of the environment and delays, by their definition, are not such a type of feature. There is, however, an exception to this argument corresponding to a case where a delay does cause a modification of the environment, eventually resulting in the loss of well–formedness and, in turn, of feasibility. This is the case where an agent is delayed indefinitely and cannot move anymore (namely when the agent is in $\mathcal{D}(t)$ for all $t \geq T$ for a given time step T). In such a situation, the agent becomes a new obstacle, potentially blocking a path critical for preserving the well–formedness. The assumption that an agent cannot be delayed indefinitely made in the previous section ensures the well-formedness of MAPD–d instances. More precisely, a MAPD–d instance is well–formed when, in addition to requirements (i)–(iii) from Section II, it satisfies also: (iv) any agent cannot be in $\mathcal{D}(t)$ forever (i.e., for all $t \geq T$ for a given T).

In a real context, condition (iv) amounts to removing or repairing the blocked agents. For instance, if an agent experiences a permanent fail, it will be removed (in this case its incomplete task will return in the task set and at least one agent must survive in the system) or repaired after a finite number of time steps. This guarantees that the well–formedness of a problem instance is preserved (or, more precisely, that it is restored after a finite time).

[1]For simplicity and w.l.o.g., we consider a path and a corresponding execution trace starting from time step 0.

Algorithm 2: TP with replanning

```
1  initialize token with the (trivial) path ⟨loc(a_i)⟩ for each
     agent a_i;
2  while true do
3      add new tasks, if any, to the task set 𝒯;
4      ℛ ← CheckCollisions(token);
5      foreach agent a_i in ℛ do
6          retrieve task τ assigned to a_i;
7          π_i ← PathPlanner(a_i, τ, token);
8          if π_i is not null then
9              update a_i's path in token with π_i;
10         else
11             recovery from deadlocks;
12         end
13     end
14     while agent a_i exists that requests token do
15         proceed like in Algorithm 1 (lines 5-17);
16     end
17     agents move along their paths in token for one time
         step (or stay at their current position if delayed);
18 end
```

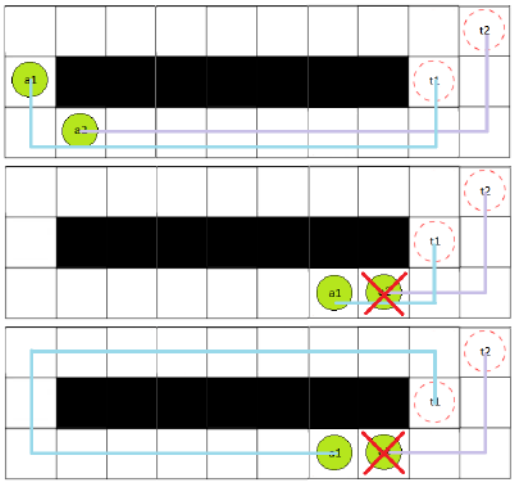


Fig. 1. An example of TP with replanning. The figure shows a grid environment with two agents and two tasks at different time steps. At time step 0 (top), the agents plan their paths without collisions. At time steps 6 and 7 (middle) a_2 is delayed and at time step 7 a collision is detected in the token. Then, a_1 regains the token and replans (bottom).

B. A MAPD–d baseline: TP with replanning

Algorithms able to solve well–formed MAPD problems, like TP, are in principle able to solve well–formed MAPD–d problems as well. The only issue is that these algorithms would return paths that do not consider possible delays occurring during execution. Delays cause paths to possibly collide, although they did not at planning time. (Note that, according to our assumptions, when an agent is delayed at time step t, there is no way to know for how long it will be delayed.)

In order to have a baseline to compare against the algorithms we propose in the next section, we introduce an adaptation of TP allowing it to work also in presence of delays. Specifically, we add to TP a replanning mechanism that works as follows: when a collision is detected between agents following their paths, the token is assigned to one of the colliding agents to allow replanning of a new collision–free path. This is a modification of the original TP mechanism where the token can be assigned only to free agents that have reached the end of their paths (see Algorithm 1). To do this, we require the token to include also the current execution traces of the agents.

Algorithm 2 reports the pseudo–code for this baseline method that we call TP with replanning. At the current time step a collision is checked using the function *CheckCollisions* (line 4): a collision occurs at time step t if an agent a_i wants to move to the same vertex to which another agent a_j wants to move or if a_i and a_j want to swap their locations on adjacent vertices. For example, this happens when a_j is delayed at t or when one of the two agents has been delayed at an earlier time step. The function returns the set $\mathcal{R}$ of non–delayed colliding agents that will try to plan new collision–free paths (line 7). The *PathPlanner* function considers a set of constraints to avoid conflicts with the current paths of other agents in the token. A problem may happen when multiple delays occur at the same time; in particular situations, two or more agents may prevent each other to follow the only paths available to complete their tasks. In this case, the algorithm recognizes the situation and implements a deadlock recovery behavior. In particular, although with our assumptions agents cannot be delayed forever, we plan short collision–free random walks for the involved agents in order to speedup the deadlock resolution (line 11). An example of execution of TP with replanning is depicted in Figure 1.

IV. Algorithms for MAPD with delays

In this section we present two algorithms, k–TP and p–TP, able to plan paths that solve MAPD–d problem instances with some guaranteed degree of robustness in face of delays. In particular, k–TP provides a deterministic degree of robustness, while p–TP provides a probabilistic degree of robustness. For developing these two algorithms, we took inspiration from the corresponding concepts of k– and p–robustness for MAPF that we outlined in Section II.

A. k–TP Algorithm

A *k–robust* solution for MAPD–d is a plan which is guaranteed to avoid collisions due to at most k consecutive delays for each agent, not only considering the paths already planned but also those planned in the future. (By the way, this is one of the main differences between our approach and the robustness for MAPF.) As we have discussed in Section III, TP with replanning (Algorithm 2) can just react to the occurrence of delays once they have been detected. The k–TP algorithm we propose, instead, plans in advance considering that delays may occur, in the attempt of avoiding replanning at runtime. The algorithm is defined as an extension of TP with replanning, so it is able to solve all well–formed MAPD–d problem instances. A core difference is an additional set of constraints enforced during path planning.

Algorithm 3: k-TP

```
1  initialize token with the (trivial) path ⟨loc(a_i)⟩ for each
     agent a_i;
2  while true do
3      add new tasks, if any, to the task set 𝒯;
4      ℛ ← CheckCollisions(token);
5      foreach agent a_i in ℛ do
6          proceed like in Algorithm 2 (lines 6-11);
7      end
8      while agent a_i exists that requests token do
9          /* token is assigned to a_i and a_i executes now */;
10         𝒯' ← {τ_j ∈ 𝒯 | no path in token ends in s_j or in
             g_j};
11         if 𝒯' ≠ {} then
12             τ ← arg min_{τ_j ∈ 𝒯'} h(loc(a_i), s_j);
13             assign a_i to τ;
14             remove τ from 𝒯;
15             π_i ← PathPlanner(a_i, τ, token);
16             if π_i is not null then
17                 update token with k-extension(π_i, k);
18         else if no task τ_j ∈ 𝒯 exists with g_j = loc(a_i)
             then
19             update a_i's path in token with the path
                 ⟨loc(a_i)⟩;
20         else
21             π_i ← Idle(a_i, token);
22             if π_i is not null then
23                 update token with k-extension(π_i, k);
24         end
25         /* a_i returns token to system */;
26     end
27     agents move along their paths in token for one time
         step (or stay at their current position if delayed);
28 end
```

The formal steps are reported in Algorithm 3. A new path π_i, before being added to the token, is used to generate the constraints (the k–extension of the path, also added to the token, lines 17 and 23) representing that, at any time step t, any vertex in

$$\{\pi_{i,t-k}, \ldots, \pi_{i,t-1}, \pi_{i,t}, \pi_{i,t+1}, \ldots, \pi_{i,t+k}\}$$

should be considered as an obstacle (at time step t) by agents planning later. In this way, even if agent a_i or agent a_j planning later are delayed up to k times, no collision will occur. For example, if $\pi_i = \langle v_1, v_2, v_3 \rangle$, the 1-extension constraints will forbid any other agent to be in $\{v_1, v_2\}$ at the first time step, in $\{v_1, v_2, v_3\}$ at the second time step, in $\{v_2, v_3\}$ at the third time step, and in $\{v_3\}$ at the fourth time step.

The path of an agent added to the token ends at the delivery vertex of the assigned task, so the space requested in the token to store the path and the corresponding k–extension constraints is finite, for finite k. Note that, especially for large values of k, it may happen that a sufficiently robust path for an agent a_i cannot be found at some time step; in this case, a_i simply returns the token and tries to replan at the next time step. The idea is that, as other agents advance along their paths, the setting becomes less constrained and a path can be found more easily. Clearly, since delays that affect the execution are not known beforehand, replanning is still necessary in those cases where an agent gets delayed for more than k consecutive time steps.

B. p–TP Algorithm

The idea of k–robustness considers a fixed value k for the guarantee, which could be hard to set: if k is too low, plans may not be robust enough and the number of (possibly costly) replans could be high, while if k is too high, it will increase the total cost of the solution with no extra benefit (see Section V for numerical data supporting these claims).

An alternative approach is to resort to the concept of p–robustness. A *p–robust* plan guarantees to keep collision probability below a certain threshold p ($0 \leq p \leq 1$). In a MAPD setting, where tasks are not known in advance, a plan could quickly reach the threshold with just few paths planned, so that no other path can be added to it until the current paths have been executed. Our solution to avoid this problem is to impose that only the collision probability of *individual* paths should remain below the threshold p, not of the whole plan. As discussed in [19], this might also be a method to ensure a notion of fairness among agents.

We thus need a way to calculate the collision probability for a given path. We adopt a model based on Markov chains [20]. Assuming that the probability that any agent is delayed at any time step is fixed and equal to p_d, we model agent a_i's execution trace e_i (corresponding to a path π_i) with a Markov chain, where the transition matrix P is such that with probability p_d the agent remains at the current vertex and with probability $1-p_d$ advances along π_i. We also assume that transitions along chains of different agents are independent. (This simplification avoids that delays for one agent propagate to other agents, which could be problematic for the model [19], while still providing an useful proxy for robustness.)

This model is leveraged by our p–TP algorithm reported as Algorithm 4. The approach is again an extension of TP with replanning, so also in this case we are able to solve any well–formed MAPD instance. Here, one difference with the basic algorithms is that before inserting a new path π_i in the token, the Markov chain model is used to calculate the collision probability $cprob_{\pi_i}$ between π_i and the paths already in the token (lines 18 and 30). Specifically, the probability distribution for the vertex occupied by an agent a_i at the beginning of a path $\pi_i = \langle \pi_{i,t}, \pi_{i,t+1}, \ldots, \pi_{i,t+n} \rangle$ is given by a (row) vector s_0 with length n that has every element set to 0 except that corresponding to the vertex $\pi_{i,t}$, which is 1. The probability distribution for the location of an agent at time step $t+j$ is given by $s_0 P^j$ (where P is the transition matrix defined above). For example, in a situation with 3 agents and 4 vertices (v_1, v_2, v_3, v_4), the probability distributions at a given time step t for the locations of agents a_1, a_2, and a_3 could be $\langle 0.6, 0.2, 0.1, 0.1 \rangle$, $\langle 0.3, 0.2, 0.2, 0.3 \rangle$, and $\langle 0.5, 0.1, 0.3, 0.1 \rangle$, respectively. Then, for any vertex traversed by the path π_i, we calculate its collision probability as 1 minus the probability that all the other agents are not at that vertex at that time step multiplied by the probability

Algorithm 4: p-TP

```
1  initialize token with path ⟨loc(a_i)⟩ for each agent a_i;
2  while true do
3      add new tasks, if any, to the task set T;
4      R ← CheckCollisions(token);
5      foreach agent a_i in R do
6          proceed like in Algorithm 2 (lines 7 - 13);
7      end
8      while agent a_i exists that requests token do
9          /* token assigned to a_i and a_i executes now */;
10         T' ← {τ_j ∈ T | no path in token ends in s_j or in
             g_j};
11         if T' ≠ {} then
12             τ ← arg min_{τ_j ∈ T'} h(loc(a_i), s_j);
13             assign a_i to τ;
14             remove τ from T;
15             j ← 0;
16             while j < itermax do
17                 π_i ← PathPlanner(a_i, τ, token);
18                 cprob_{π_i} ← MarkovChain(π_i, token);
19                 if cprob_{π_i} < p then
20                     update a_i's path in token with π_i;
21                     break
22                 j ← j + 1;
23             end
24         else if no task τ_j ∈ T exists with g_j = loc(a_i)
             then
25             update a_i's path in token with the path
                 ⟨loc(a_i)⟩;
26         else
27             j ← 0;
28             while j < itermax do
29                 π_i ← Idle(a_i, token);
30                 cprob_{π_i} ← MarkovChain(π_i, token);
31                 if cprob_{π_i} < p then
32                     update a_i's path in token with π_i;
33                     break
34                 j ← j + 1;
35             end
36         end
37         /* a_i returns token and system executes now */;
38     end
39     agents move along their paths in token for one time
         step (or stay at their current position if delayed);
40 end
```

that the agent is actually at that vertex at the given time step. Following the above example, the collision probability in v_1 for agent a_1 at t (i.e., the probability that at least one of the other agents is at v_1 at t) is calculated as $[1 - (1 - 0.3) \cdot (1 - 0.5)] \cdot 0.6 = 0.39$. The collision probabilities of all the vertices along the path are summed to obtain the collision probability $cprob_{\pi_i}$ for the path π_i. If this probability is above the threshold p (lines 19 and 31), the path is rejected and a new one is calculated. If an enough robust path is not found after a fixed number of rejections *itermax*, the token is returned to the system and the agent will try to replan at the next time step (as other agents advance along their paths, chances of collisions could decrease).

Also for p–TP, since the delays are not known beforehand, replanning is still necessary. Moreover, we need to set the value of p_d, with which we build the probabilistic guarantee according to the specific application setting. We deal with this in the next section.

V. Experimental Results

A. Setting

Our experiments are conducted on a 3.2 GHz Intel Core i7 8700H laptop with 16 GB of RAM. We tested our algorithms in two warehouse 4–connected grid environments where the effects of delays can be significant: a small one, 15×13 units, with 4 and 8 agents, and a large one, 25×17, with 12 and 24 agents (Figure 2). (Environments of similar size have been used in [9].) At the beginning, the agents are located at the non-task endpoints. We create a sequence of 50 tasks choosing the pickup and delivery vertices uniformly at random among a set of predefined vertices. The arrival time of each task is determined according to a Poisson distribution [21]. We test 3 different arrival frequencies λ for the tasks: 0.5, 1, and 3 (since, as discussed later, the impact of λ on robustness is not relevant, we do not show results for all values of λ). During each run, 10 delays per agent are randomly inserted and the simulation ends when all the tasks have been completed.

We evaluate k–TP and p–TP against the baseline TP with replanning (to the best of our knowledge, we are not aware of any other algorithm for finding robust solutions to MAPD–d). For p–TP we use two different values for the parameter p_d, 0.02 and 0.1, modeling a low and a higher probability of delay, respectively. (Note that this is the expected delay probability used to calculate the robustness of a path and could not match with the delays actually observed.) For planning paths of individual agents (*PathPlanner* in the algorithms), we use an A* path planner with Manhattan distance as heuristic.

Solutions are evaluated according to the makespan (i.e., the earliest time step at which all tasks are completed, see Section II). (Results for the service time are qualitatively similar and are not reported here.) We also consider the number of replans performed during execution and the total time required by each simulation (including time for both planning and execution). The reported results are averages over 100 randomly restarted runs. All algorithms are implemented in Python and the code is publicly available at an online repository[2].

B. Results

Results relative to small warehouse are shown in Tables I and II and those relative to large warehouse are shown in Tables III and IV. For the sake of readability, we do not report the standard deviation in tables. Standard deviation values do not present any evident oddity and support the conclusions about the trends reported below.

The baseline algorithm, TP with replanning, appears twice in each table: as k–TP with $k = 0$ (that is the basic implementation as in Algorithm 2) and as p–TP with $p_d =$

[2] https://github.com/Lodz97/Multi-Agent_Pickup_and_Delivery

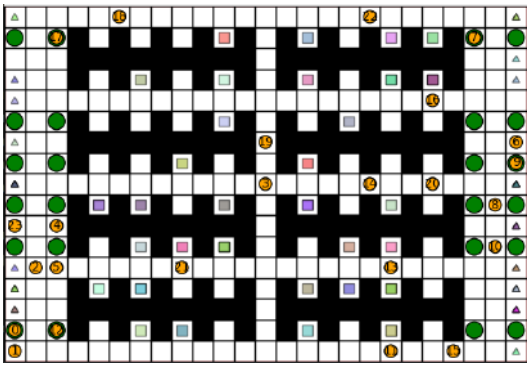


Fig. 2. Large warehouse with 24 agents, obstacles (black), pickup (colored squares) and delivery (triangles) vertices, and endpoints (green circles)

TABLE I

SMALL WAREHOUSE, $\lambda = 0.5$, AND 10 DELAYS PER AGENT

k or p		$\ell = 4$			$\ell = 8$		
		makespan	# replans	runtime [s]	makespan	#replans	runtime [s]
k-TP	0	**364.88**	7.26	**0.85**	**234.59**	16.04	**2.11**
	1	374.48	1.4	0.91	240.69	3.85	2.27
	2	390.82	0.1	1.16	241.14	0.73	2.15
	3	411.09	0.01	1.59	259.38	0.09	3.12
	4	436.12	**0.0**	2.0	278.33	**0.04**	4.49
p-TP, $p_d = .1$	1	**364.88**	7.26	1.14	**234.59**	16.04	2.63
	0.5	369.5	6.29	1.81	237.27	12.59	5.0
	0.25	395.07	4.29	2.88	255.21	5.63	6.11
	0.1	409.17	2.9	3.16	268.99	3.23	6.32
	0.05	428.64	2.93	3.42	279.26	2.76	6.48
p-TP, $p_d = .02$	0.5	366.72	7.34	1.29	238.83	12.81	3.87
	0.25	378.42	6.8	1.57	236.21	10.21	4.38
	0.1	391.63	4.53	2.37	250.39	6.73	5.57
	0.05	405.53	3.51	2.66	256.24	4.25	5.34

TABLE II

SMALL WAREHOUSE, $\lambda = 3$, AND 10 DELAYS PER AGENT

k or p		$\ell = 4$			$\ell = 8$		
		makespan	# replans	runtime [s]	makespan	# replans	runtime [s]
k-TP	0	**354.77**	8.3	**0.6**	**217.79**	14.67	1.93
	1	363.22	1.47	0.77	219.87	4.01	**1.81**
	2	383.59	0.2	0.95	226.75	0.58	1.89
	3	400.77	0.01	1.33	250.23	0.12	3.02
	4	429.12	**0.0**	1.68	263.47	**0.01**	4.32
p-TP, $p_d = .1$	1	**354.77**	8.3	0.86	**217.79**	14.67	2.53
	0.5	360.29	6.7	1.45	224.31	11.06	4.93
	0.25	381.98	5.12	2.3	245.24	6.46	5.83
	0.1	404.92	2.93	2.81	251.42	3.55	5.66
	0.05	417.04	2.65	3.05	262.73	3.65	6.11
p-TP, $p_d = .02$	0.5	358.14	8.05	1.25	219.58	13.19	3.61
	0.25	372.92	7.02	1.57	228.25	10.93	3.77
	0.1	380.31	4.41	2.12	233.97	6.89	4.65
	0.05	393.55	3.45	2.5	244.62	4.81	4.98

0.1 and $p = 1$ (which accepts all paths). The two versions of the baseline return the same results in terms of makespan and number of replans (we use the same random seed initialization for runs with different algorithms), but the total runtime is larger in the case of p–TP, due to the overhead of calculating the Markov chains and the collision probability for each path.

TABLE III

LARGE WAREHOUSE, $\lambda = 0.5$, AND 10 DELAYS PER AGENT

k or p		$\ell = 12$			$\ell = 24$		
		makespan	# replans	runtime [s]	makespan	# replans	runtime [s]
k-TP	0	283.62	17.18	**2.8**	269.25	20.71	8.32
	1	**276.7**	3.88	3.27	**264.96**	5.37	**5.78**
	2	285.32	1.18	4.89	275.48	1.62	9.54
	3	304.05	0.24	7.54	300.55	0.4	15.55
	4	310.59	**0.01**	10.9	300.45	**0.1**	22.11
p-TP, $p_d = .1$	1	283.62	17.18	4.12	269.25	20.71	11.2
	0.5	286.95	10.02	11.3	291.78	17.09	38.61
	0.25	305.13	5.38	17.26	313.63	9.59	58.95
	0.1	330.58	4.51	19.6	322.26	4.51	54.92
	0.05	337.33	3.56	20.27	348.89	3.89	57.24
p-TP, $p_d = .02$	0.5	289.86	14.51	7.41	290.05	20.3	28.74
	0.25	287.72	9.92	10.19	286.77	14.15	39.47
	0.1	311	6.53	13.76	304.24	8.94	49.04
	0.05	313.38	6.41	14.91	308.1	7.02	49.96

TABLE IV

LARGE WAREHOUSE, $\lambda = 3$, AND 10 DELAYS PER AGENT

k or p		$\ell = 12$			$\ell = 24$		
		makespan	# replans	runtime [s]	makespan	# replans	runtime [s]
k-TP	0	265.23	18.96	**2.91**	258.49	30.83	**8.12**
	1	269.78	4.22	3.28	254.56	8.98	9.81
	2	274.78	1.19	4.75	261.3	1.71	12.03
	3	279.02	0.18	7.31	273.56	0.59	19.43
	4	290.59	**0.04**	10.76	282.07	**0.17**	30.91
p-TP, $p_d = .1$	1	265.23	18.96	4.16	258.49	30.83	10.78
	0.5	268.74	11.31	9.04	257.64	17.21	36.74
	0.25	298.01	7.39	14.58	287.75	9.96	48.14
	0.1	318.37	5.3	16.33	310.46	6.32	47.11
	0.05	331.1	3.83	16.83	334.06	4.42	47.62
p-TP, $p_d = .02$	0.5	**259.64**	12.47	7.22	**247.76**	20.47	26.21
	0.25	289.75	12.05	9.23	264.6	15.72	39.68
	0.1	280.07	6.78	11.59	290.65	9.88	42.76
	0.05	298.34	6.21	12.98	293.68	8.81	42.23

Looking at robustness, which is the goal of our algorithms, we can see that, in all settings, both k–TP and p–TP significantly reduce the number of replans with respect to the baseline. For k–TP, increasing k leads to increasingly more robust solutions with less replans, and the same happens for p–TP when the threshold probability p is reduced. However, increasing k shows a more evident effect on the number of replans than reducing p. More robust solutions, as expected, tend to have a larger makespan, but the first levels of robustness ($k = 1$, $p = 0.5$) manage to reduce significantly the number of replans with a small or no increase in makespan. For instance, in Table IV, k–TP with $k = 1$ decreases the number of replans of more than 75% with an increase in makespan of less than 2%, with respect to the baseline. Pushing towards higher degrees of robustness (i.e., increasing k or decreasing p) tends to increase makespan significantly with diminishing returns in terms of number of replans, especially for k–TP.

Comparing k–TP and p–TP, it is clear that solutions produced by k–TP tend to be more robust at similar makespan (e.g., see k–TP with $k = 1$ and p–TP with $p_d = .1$ and $p = 0.5$ in Table I), and decreasing p may sometimes lead to relevant increases in makespan. This suggests that our implementation of p–TP has margins for improvement: if the computed path exceeds the threshold p we wait the next time step to replan, without storing any collision information extracted from the Markov chains; finding ways to exploit this information may lead to an enhanced version of p–TP (this investigation is left as future work). It is also interesting to notice the effect of p_d in p–TP: a higher p_d (which, in our experiments, amounts to overestimating the actual delay probability that, considering that runs last on average about 300 time steps and there are 10 delays per agent, is equal to $\frac{10}{300} = 0.03$) leads to solutions requiring less replans, but with a noticeable increase in makespan.

Considering runtimes, k–TP and p–TP are quite different. For k–TP, we see a trend similar to that observed for makespan: a low value of k ($k = 1$) often corresponds to a slight increase in runtime with respect to the baseline (sometimes even a decrease), while for larger values of k the runtime may be much longer than the baseline. Instead, p–TP shows a big increase in runtime with respect to the baseline, that does not change too much with the values of p, at least for low values of p ($p = 0.1$, $p = 0.05$). Finally, we can see how different task frequencies λ have no significant impact on our algorithms, but higher frequencies have the global effect of reducing makespan tasks (which are always 50 per run) are available earlier.

Finally, we run simulations in a even larger warehouse 4–connected grid environment of size 25×37, with 50 agents, $\lambda = 1$, 100 tasks, and 10 delays per agent. The same qualitative trends discussed above are observed also in this case. For example, k–TP with $k = 2$ reduces the number of replans of 93% with an increase of makespan of 5% with respect to the baseline. The runtime of p–TP grows to hundreds of seconds, also with large values of p, suggesting that some improvements are needed. Full results are not reported here due to space constraints.

VI. Conclusion

In this paper, we introduced a variation of the Multi-Agent Pickup and Delivery (MAPD) problem, called MAPD with delays (MAPD–d), which considers an important practical issue encountered in real applications: delays in execution. In a MAPD–d problem, agents must complete a set of incoming tasks (by moving to the pickup vertex of each task and then to the corresponding delivery vertex) even if they are affected by an unknown but finite number of delays during execution. We proposed two algorithms to solve MAPD–d, k–TP and p–TP, that are able to solve well–formed MAPD–d problem instances and provide deterministic and probabilistic robustness guarantees, respectively. Experimentally, we compared them against a baseline algorithm that reactively deals with delays during execution. Both k–TP and p–TP plan robust solutions, greatly reducing the number of replans needed with a small increase in solution makespan. k–TP showed the best results in terms of robustness–cost trade–off, but p–TP still offers great opportunities for future improvements.

Future work will address the enhancement of p–TP according to what we outlined in Section V-B and the experimental testing of our algorithms in real–world settings.

References

[1] H. Ma, "Target assignment and path planning for navigation tasks with teams of agents," Ph.D. dissertation, University of Southern California, Department of Computer Science, Los Angeles, USA, 2020.

[2] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, R. Barták, and E. Boyarski, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proc. SoCS*, 2019, pp. 151–159.

[3] P. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," in *Proc. IAAI*, 2007, pp. 1752–1759.

[4] M. Veloso, J. Biswas, B. Coltin, and S. Rosenthal, "Cobots: Robust symbiotic autonomous mobile service robots," in *Proc. IJCAI*, 2015, pp. 4423–4429.

[5] H. Ma, J. Yang, L. Cohen, T. Kumar, and S. Koenig, "Feasibility study: Moving non-homogeneous teams in congested video game environments," *Proc. AIIDE*, pp. 270–272, 2017.

[6] H. Ma, T. Kumar, and S. Koenig, "Multi-agent path finding with delay probabilities," in *Proc. AAAI*, 2017, pp. 3605–3612.

[7] D. Atzmon, R. Stern, A. Felner, N. Sturtevant, and S. Koenig, "Probabilistic robust multi-agent path finding," in *Proc. ICAPS*, 2020, pp. 29–37.

[8] D. Atzmon, R. Stern, A. Felner, G. Wagner, R. Barták, and N.-F. Zhou, "Robust multi-agent path finding and executing," *J Artif Intell Res*, vol. 67, pp. 549–579, 2020.

[9] H. Ma, J. Li, T. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," in *Proc. AAMAS*, 2017, pp. 837–845.

[10] J. Yu and S. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proc. AAAI*, 2013, pp. 1443–1449.

[11] P. Surynek, "An optimization variant of multi-robot path planning is intractable," in *Proc. AAAI*, 2010, pp. 1261–1263.

[12] Z. Chen, D. Harabor, J. Li, and P. Stuckey, "Symmetry breaking for k-robust multi-agent path finding," in *Proc. AAII*, 2021, pp. 12 267–12 274.

[13] J. Svancara, M. Vlk, R. Stern, D. Atzmon, and R. R. Barták, "Online multi-agent pathfinding," in *Proc. AAAI*, 2019, pp. 7732–7739.

[14] H. Ma, "A competitive analysis of online multi-agent path finding," in *Proc. ICAPS*, 2021, pp. 234–242.

[15] T. Shahar, S. Shekhar, D. Atzmon, A. Saffidine, B. Juba, and R. Stern, "Safe multi-agent pathfinding with time uncertainty," *J Artif Intell Res*, vol. 70, pp. 923–954, 2021.

[16] E. Khalastchi and M. Kalech, "Fault detection and diagnosis in multi-robot systems: A survey," *Sensors*, vol. 19, no. 18, pp. 1–19, 2019.

[17] H. Ma, W. Hönig, T. Kumar, N. Ayanian, and S. Koenig, "Lifelong path planning with kinematic constraints for multi-agent pickup and delivery," in *Proc. AAAI*, 2019, pp. 7651–7658.

[18] P. Guo, H. Kim, N. Virani, J. Xu, M. Zhu, and P. Liu, "Roboads: Anomaly detection against sensor and actuator misbehaviors in mobile robots." in *Proc. DSN*, 2018, pp. 574–585.

[19] G. Wagner and H. Choset, "Path planning for multiple agents under uncertainty," in *Proc. ICAPS*, 2017, pp. 577–585.

[20] D. Levin and Y. Peres, *Markov chains and mixing times*. American Mathematical Society, 2017, vol. 107.

[21] K.-K. Tse, "Some applications of the Poisson process," *Appl Math*, vol. 05, pp. 3011–3017, 2014.

On Improvement Heuristic to Solutions of the Close Enough Traveling Salesman Problem in Environments with Obstacles

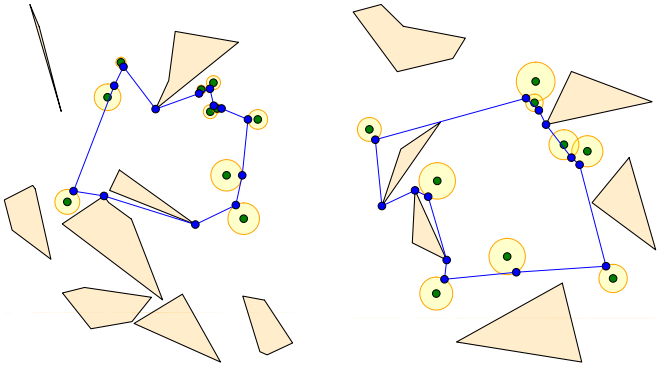
Jindřiška Deckerová Kristýna Kučerová Jan Faigl

***Abstract*— In this paper, we present a novel improvement heuristic to address the Close Enough Traveling Salesman Problem in environments with obstacles (CETSP$_{\text{obs}}$). The CETSP$_{\text{obs}}$ is a variant of the Traveling Salesman Problem (TSP), where the goal is to find a sequence of visits to given disk-shaped regions together with the points of visits to the regions. We address challenging instances in a polygonal domain with polygonal obstacles, where the final path connecting the regions must be collision-free. We propose a novel `Post-Optimization` procedure using Mixed Integer Non-Linear Programming (MINLP) to improve existing heuristic solutions to the CETSP$_{\text{obs}}$. We deploy the method with existing heuristic solvers and based on the presented evaluation results, the proposed `Post-Optimization` significantly improves the heuristic solutions of all examined solvers and makes them competitive regarding the solution quality. The statistical evaluation reveals that the sequence found using relatively sparse sampling of the disk regions yields the best solutions among the evaluated solvers. The results support the benefit of the proposed MINLP-based solution to the continuous optimization of the CETSP$_{\text{obs}}$.**

I. Introduction

The studied problem is motivated by *multi-goal path planning* [1] that is a robotic variant of the well-known combinatorial *Traveling Salesman Problem* (TSP) [2], where paths connecting the given set of locations are collision-free among possible obstacles in the environment. In the TSP, we search for a cost-efficient closed-loop tour visiting the locations, and we thus determine an optimal sequence of visits to the locations. Hence, the TSP represents a suitable problem formulation for various robotic sequencing tasks [3]. Furthermore, in remote data collection missions [4], [5], it is sufficient to visit a close region around the particular location and thus save the travel cost. In such scenarios, the TSP becomes the *TSP with Neighborhoods* (TSPN), where we need to determine the optimal sequence to visit the regions and also the optimal point of the visit to each region.

The neighborhoods in the TSPN can be represented as continuous regions [4], [6], [7], [8], or as clusters of regions [9], [10], [11], [12], or as clusters of locations [13]. In general, the TSPN is an APX-hard [14], and many heuristic approaches [15], [16], [17], [18], and approximation algorithms [19], [20], [14], [21] have been proposed. Further, the TSPN with disk-shaped neighborhoods has been introduced as the *Close Enough TSP* (CETSP) in [4]. Although exact methods have been proposed to solve the CETSP [22], [23], they do not account for possible obstacles, and connections between the regions are only straight line segments with the length determined as the Euclidean distance between points of visits to the regions.



(a) `t10_005` (b) `t9_005`

Fig. 1. Instances of the CETSP$_{\text{obs}}$ with found solutions depicted in blue.

In this paper, we address the robotic variant of the CETSP in the polygonal domain with polygonal obstacles, further referred to as the CETSP$_{\text{obs}}$; see examples of instances in Fig. 1. When compared to the CETSP, the main challenge of the CETSP$_{\text{obs}}$ is determining collision-free paths between points of visits to the regions that can be arbitrarily located in the regions while finding the optimal sequence. Thus, the shortest paths connecting the regions need to be determined quickly, as many queries can be expected during the optimization of the sequence and points of visits to the regions. For the regular TSP with point locations or sampled regions to a discrete set of points, visibility graph can be constructed [24], [25] for shortest path queries; however, it is not the case of the CETSP$_{\text{obs}}$ with continuous regions.

Only two approaches explicitly address the CETSP$_{\text{obs}}$ (to the best of the authors' knowledge). The first is based on the shortest-path approximation employed in an unsupervised learning-based solution of the TSP in the polygonal domain [26], [27]. The second is the GLNSC [28] based on the decomposition of the CETSP$_{\text{obs}}$ to the continuous optimization of the CETSP and the point-to-point optimization using Delaunay triangulation. Besides, the discretized variant of the CETSP$_{\text{obs}}$ can be solved as the *Generalized TSP* (GTSP) [13] using pre-computed shortest paths among the obstacles and each sampled location of the regions. However, the optimal solution of such a discretized instance would be only the approximate solution of the original CETSP$_{\text{obs}}$

The authors are with the Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University, Prague, Czech Republic. {deckejin|kucerkr1|faiglj}@fel.cvut.cz

The presented work has been supported by the Czech Science Foundation (GAČR) under the research project No. 22-05762S.

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

depending on the sampling density.

We propose to address approximations of existing solutions to the CETSP$_{\text{obs}}$ by the `Post-Optimization` procedure that improves any existing heuristic solution. The procedure is based on formulating the problem as *Mixed Integer Non-Linear Programming* (MINLP) and exploits the given sequence of visits to the regions. We employed the procedure to existing solvers GLNSC [28] and unsupervised learning of the *Self-Organizing Map* (SOM) [27]. In addition, we adopted GTSP-based approach [18] to the *Generalized TSPN* (GTSPN), which first determines the sequence of visits to the regions' centers and then computes the points of visits using the local iterative optimization. Based on the empirical evaluation, the proposed `Post-Optimization` procedure improves solutions found by the existing solvers and makes the sampling-based GTSP the best-performing solver.

The rest of the paper is organized as follows. The CETSP$_{\text{obs}}$ is formally defined in Section II. The examined SOM and GTSP-based solvers are briefly described in Section III. The proposed `Post-Optimization` procedure is presented in Section IV. The results of the empirical evaluation are summarized in Section V, and the paper is concluded in Section VI.

II. Problem Statement

The studied CETSP$_{\text{obs}}$ is to find the shortest multi-point path that visits each of the n disk-shaped regions $\mathcal{S} = \{S_1, \ldots, S_n\}$ while avoiding m polygonal obstacles $\mathcal{O} = \{O_1, \ldots, O_m\}$. Each region $S_i \in \mathcal{S}$ is defined by its center $\boldsymbol{c}_i \in \mathbb{R}^2$, radius $\delta_i \geq 0$, and it is entirely inside the free space of the polygonal domain. A polygon obstacle $O_j \in \mathcal{O}$ is defined by a sequence of l_j vertices represented as points in $\mathbb{R}^2$, $O_j = (\boldsymbol{o}_j^1, \ldots, \boldsymbol{o}_j^{l_j})$, $\boldsymbol{o}_j^t \in \mathbb{R}^2$, for $1 \leq t \leq l_j$.

A solution of the CETSP$_{\text{obs}}$ is defined by a sequence Σ of visits to regions together with the points of visits $\mathcal{P}$ further referred to as *waypoints*. The final multi-point path is formed by a sequence of (shortest) paths among obstacles connecting $\mathcal{P}$ according to Σ. Hence, for the purpose of finding a path among obstacles connecting two waypoints, we consider a set of obstacles' points $\mathcal{Q}$ denoting the vertices of the obstacles' borders. Thus, the multi-point path is denoted $(\Sigma, \mathcal{P}, \mathcal{Q})$, where the terms can be defined as follows.

- Σ – *Sequence of visits* defining the order of visits to the regions: $\Sigma = (\sigma_1, \ldots, \sigma_n)$, $\sigma_i \neq \sigma_j$ for $i \neq j$.
- $\mathcal{P}$ – *Waypoints* are the points of visits to the regions: $\mathcal{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$, $\boldsymbol{p}_i \in \mathbb{R}^2$. For each waypoint $\boldsymbol{p}_i$, it holds $\|\boldsymbol{c}_i - \boldsymbol{p}_i\| \leq \delta_i$.
- $\mathcal{Q}$ – *Obstacles' points* forming the final path connecting $\mathcal{P}$ according to Σ, $\mathcal{Q} = \cup_{i=1}^{n} \{\boldsymbol{q}_i^0, \ldots, \boldsymbol{q}_i^{k_i}\}$, where $k_i \geq 0$ denotes the number of obstacles' points of the path connecting consecutive waypoints $\boldsymbol{p}_{\sigma_i}$ and $\boldsymbol{p}_{\sigma_{i+1}}$. Note that for a closed multi-point path, $\boldsymbol{p}_{\sigma_1}$ is the consecutive waypoint of $\boldsymbol{p}_{\sigma_n}$.

The length $\mathcal{L}^*$ of the path between two waypoints $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ can be defined according to the number of obstacles' points k_i. If the straight line connection of the waypoints is collision-free, the length is directly the Euclidean distance

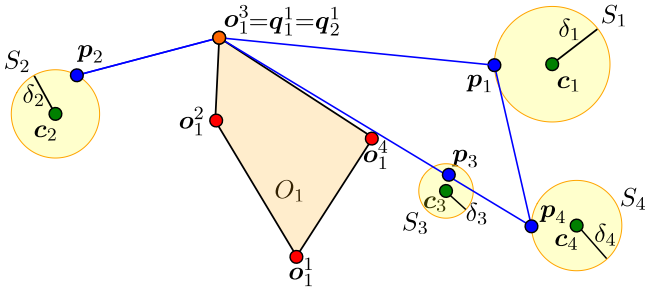


Fig. 2. A solution of the CETSP$_{\text{obs}}$ instance with $n = 4$ regions and one obstacle, $m = 1$. Regions' centers are small green disks. Vertices of the obstacles are small red disks, while obstacles' points $\mathcal{Q}$ are in orange. The determined waypoints $\mathcal{P}$ are visualized as small blue disks.

$\mathcal{L}^*(\boldsymbol{p}_i, \boldsymbol{p}_j) = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|$ and $k_i = 0$; for $k_i = 1$, it is $\mathcal{L}^*(\boldsymbol{p}_i, \boldsymbol{p}_j) = \|\boldsymbol{p}_i - \boldsymbol{q}_i^1\| + \|\boldsymbol{q}_i^1 - \boldsymbol{p}_j\|$; otherwise

$$\mathcal{L}^*(\boldsymbol{p}_i, \boldsymbol{p}_j) = \|\boldsymbol{p}_i - \boldsymbol{q}_i^1\| + \sum_{l=1}^{k_i - 1} \|\boldsymbol{q}_i^l - \boldsymbol{q}_i^{l+1}\| + \|\boldsymbol{q}_i^{k_i} - \boldsymbol{p}_j\|. \tag{1}$$

The used notation is visualized in an example of the solution instance in Fig. 2. The CETSP$_{\text{obs}}$ is formulated as the optimization problem in Problem 1.

Problem 1 (CETSP with polygonal domain (CETSP$_{\text{obs}}$)):

$$\mathcal{L}^* = \min_{\Sigma, \mathcal{P}, \mathcal{Q}} \mathcal{L}^*(\boldsymbol{p}_{\sigma_n}, \boldsymbol{p}_{\sigma_1}) + \sum_{i=1}^{n-1} \mathcal{L}^*(\boldsymbol{p}_{\sigma_i}, \boldsymbol{p}_{\sigma_{i+1}}) \tag{2}$$

s.t.

$$\Sigma = (\sigma_1, \ldots, \sigma_n), \sigma_i \neq \sigma_j \text{ if } i \neq j, 1 \leq \sigma_i \leq n, \tag{3}$$

$$\mathcal{P} = \{\boldsymbol{p}_{\sigma_1}, \ldots, \boldsymbol{p}_{\sigma_n}\}, \quad \boldsymbol{p}_i \in \mathbb{R}^2, \tag{4}$$

$$\|\boldsymbol{p}_{\sigma_i} - \boldsymbol{c}_{\sigma_i}\| \leq \delta_{\sigma_i} \quad \forall i \in \{1, \ldots, n\}. \tag{5}$$

III. Background

The studied CETSP$_{\text{obs}}$ is solved using existing heuristics and applying the proposed `Post-Optimization` procedure to their provided solution. In addition to the GLNSC [28] that directly solves the addressed CETSP$_{\text{obs}}$, an unsupervised learning approach has been proposed to solve the CETSP$_{\text{obs}}$ with polygonal regions in [27]. Besides, the GTSP-based approach [18] can be utilized to solve a discretized variant of the CETSP$_{\text{obs}}$. Therefore, the two additional methods are briefly overviewed with the relatively straightforward modifications for the CETSP$_{\text{obs}}$ to make the paper self-contained.

A. SOM-based Unsupervised Learning for the CETSP$_{\text{obs}}$

The unsupervised learning approach presented in [27] is based on the SOM for the TSP [29] and has been deployed in the polygonal domain in [26] using an approximation of the shortest path based on the underlying convex partitioning of the polygonal domain. Although there are multiple improvements of the SOM-based unsupervised learning for various routing problems, such as [30], [31], [32] and its

generalization *Growing Self-Organizing Array* (GSOA) [17] deployed in [18], we directly utilize the available solver [27].

The unsupervised learning [27] is an iterative procedure in which a ring of $2n$ nodes (representing the multi-point path) is adapted to the regions during learning epochs. For each region, the closest node of the ring is determined as a winner node. Then, the winner node is adapted (moved) toward the region together with its neighboring nodes with the decreasing power of adaptation based on the neighboring function. The adaptation's power is controlled by the learning gain decreased every learning epoch to converge the ring to a stable solution. Note that the adaptation (movement) is along the shortest paths (or their approximation) among obstacles. Besides, the regions are examined in a random order in each epoch to avoid local minima [26].

After a finite number of epochs, the ring represents a multi-point path as each region has a unique winner node because of inhibition of the winners for each epoch [27]. Since the ring is represented as an array of nodes, the sequence of visits to the regions can be retrieved by traversing the ring. Besides, the winner node is associated with the point of the visit to the polygonal region.

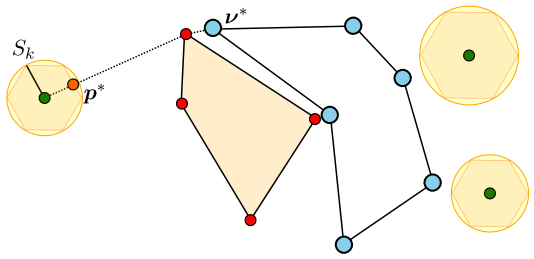


Fig. 3. Illustration of the winner node ν^* for the region S_k and its point of visit to the region $\boldsymbol{p}^*$ determined in the SOM solver [27]. The ring of nodes is represented as connected small blue disks.

The main modification of [27] for the herein addressed $\text{CETSP}_{\text{obs}}$ with disk-shaped regions is to represent each disk as the polygonal region with l vertices. However, unsupervised learning can still benefit from continuous regions. It is because the point of the visit to the region $\boldsymbol{p}^*$ is determined as the point on the region's boundary that intersects the shortest path between a node ν^* and disk's center, see Fig. 3. If the winner node is already inside the region, which can be caused by the adaptation of other nodes, its position is used as $\boldsymbol{p}^*$. The final multi-point path is retrieved by traversing the ring and connecting the associated points to the winner nodes. The reader is referred to [27] or [26] for further details on the utilized unsupervised learning.

B. GTSP-based Solver to the $\text{CETSP}_{\text{obs}}$

The GTSP-based solver [18] has been proposed to solve a continuous variant of the GTSPN by discretization to the GTSP using regions' centers and deploying the heuristic GTSP solver [33] to determine the sequence of visits. Deploying the GTSP-based solver to the $\text{CETSP}_{\text{obs}}$ is straightforward. The disk-shaped regions are discretized into a finite set of samples on the disks' boundaries. Then, the visibility graph [34] is employed to determine the shortest paths between the samples; see Fig. 4. The following four steps summarize the usage of the GTSP-based solver [18].

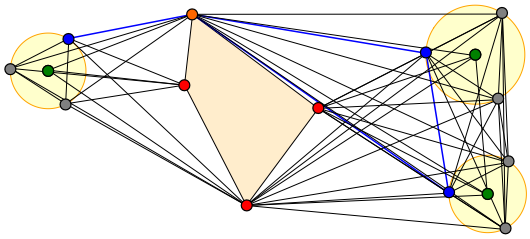


Fig. 4. An example of the GTSP-based solution of the $\text{CETSP}_{\text{obs}}$ using visibility graph. The found solution is depicted in blue.

- *Step 1.* Sample each region S_i into l samples Ξ on the region's border.
- *Step 2.* Construct visibility graph $\mathcal{G}$ in the polygonal domain for the samples Ξ.
- *Step 3.* Create an instance of the GTSP for the GTSP solver [33] using samples Ξ as a set of locations and the shortest paths between samples determined with $\mathcal{G}$ as the lengths between sets.
- *Step 4.* Use the GLKH solver [33] to find a sequence of visits and $\mathcal{G}$ to determine the solution $(\Sigma, \mathcal{P}, \mathcal{Q})$.

IV. Proposed Post-Optimization Procedure

The proposed `Post-Optimization` procedure is based on the MINLP mathematical model to find locally optimal solutions of the studied problem using the given sequence of visits Σ from some feasible solution $(\Sigma, \mathcal{P}, \mathcal{Q})$. The optimization idea is to minimize the path connecting the waypoints; however, we need to account for the obstacles' points through which a path among obstacles connects the waypoints. Therefore, in the MINLP model, we have two types of waypoints. The first waypoints are denoted $\mathcal{P}$, further also called the *disks' waypoints*, and are being optimized according to the problem statement in Section II. The second type of waypoints are the obstacles' points $\mathcal{Q}$, further referred to as the *obstacles' waypoints*.

We do not need to include all obstacles' points in the model, but only those connected with a region's waypoint by a straight line segment in the multi-point path. A connection between two consecutive obstacles' points (vertices) is guaranteed to be collision-free (e.g., using a visibility graph), and we do not change the topology of the multi-point path. Thus, depending on the number of obstacles' vertices of the path connecting two consecutive waypoints p_i and p_j, we add zero, one q_i^1 or two obstacles' waypoints q_i^1 and q_i^2 as defined in (1). Furthermore, if an obstacle's point (vertex) is included in two (or multiple) paths, such as the (orange) vertex in Fig. 4, the point is added to the model as the obstacle waypoint multiple times. Thus, the number of waypoints n' in the model can be $n' \geq n$.

Since all the waypoints have disk-shaped regions in the MINLP formulation, we consider zero disk's radius for obstacles' waypoints, and we get a sequence of regions $\mathcal{S}'$. Hence, a position of the waypoint with $\delta_i = 0$ is not effectively optimized in the MINLP solution. The model is summarized in Model 1 with the following variables.

- *Decision variables* $\boldsymbol{x} \in \mathbb{R}^{n' \times 2}$ represent the optimized waypoints $\mathcal{P}_{\text{opt}} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{n'})$.
- *Auxiliary variables* $\boldsymbol{f} \in \mathbb{R}^{n'}$ and $\boldsymbol{w} \in \mathbb{R}^{n' \times 2}$ are used to minimize the squared difference of two consecutive waypoints (6–8).
- Further *auxiliary variables* $\boldsymbol{v} \in \mathbb{R}^{n' \times 2}$ are used to ensure that each waypoint $\boldsymbol{p}_i$ is within δ_i distance from the particular region's center $\boldsymbol{c}_i$ (9–10).

Model 1 (MINLP model):

$$\min_{\boldsymbol{x} \in \mathbb{R}^{n' \times 2}} \sum_{i=1}^{n'} f_i \tag{6}$$

s.t.

$$f_i^2 \geq \boldsymbol{w}_i^T \, \boldsymbol{w}_i, \qquad \forall i \in \{1, \ldots, n'\} \tag{7}$$

$$\boldsymbol{w}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i, \qquad \forall i \in \{1, \ldots, n'-1\} \tag{8}$$

$$\delta_i^2 \geq \boldsymbol{v}_i^T \boldsymbol{v}_i, \qquad \forall i \in \{1, \ldots, n'\} \tag{9}$$

$$\boldsymbol{v}_i = \boldsymbol{x}_i - \boldsymbol{c}_i, \qquad \forall i \in \{1, \ldots, n'\} \tag{10}$$

In solving the created Model 1, we aim to optimize the position of the disks' waypoints within the particular disk. However, the optimized position might yield a collision of the straight line segment connecting two consecutive waypoints (regions of $\mathcal{S}'$) and an obstacle. Therefore, three constraints are added if and only if there is an obstacle O_j between two consecutive regions of $\mathcal{S}'$ as follows.

The first constraint

$$\boldsymbol{d}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i \tag{11}$$

uses *auxiliary variables* $\boldsymbol{d}_i \in \mathbb{R}^{n' \times 2}$ to express a straight line segment of two consecutive waypoints as the difference in coordinates. The second and third constraints are for l_j obstacle's vertices

$$-d_{i,2}\, o^l_{j,1} + d_{i,1}\, o^l_{j,2} + d_{i,2}\, x_{i,1} - d_{i,1}\, x_{i,2} \leq M\, y_{i,j} \tag{12}$$

and

$$-d_{i,2}\, o^l_{j,1} + d_{i,1}\, o^l_{j,2} + d_{i,2}\, x_{i,1} - d_{i,1}\, x_{i,2} \geq -M\,(1 - y_{i,j}) \tag{13}$$

for $1 \leq l \leq l_j$ to ensure that the straight line segment expressed as $\boldsymbol{d}_i$ does not intersects the obstacle O_j represented by a sequence of points $O_j = (\boldsymbol{o}_j^1, \ldots, \boldsymbol{o}_j^{l_j})$. The constraints express that two waypoints are on the same half-plane. Therefore, only one of the constraints (12) or (13) is activated in the model. That is achieved by using the *Big-M* method (we use $M = 100\,000$), and *binary variables* $\boldsymbol{y} \in \{0,1\}^{n'}$ are used to activate the particular constraints.

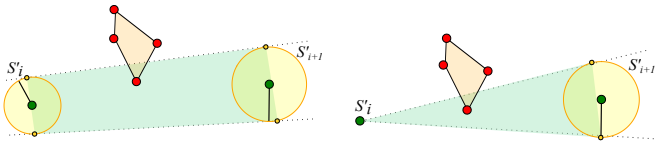
The proposed `Post-Optimization` is based on the MINLP model's construction, summarized in Algorithm 1. Adding constraints (11–13) corresponds to Lines 9, 11 and 12 of Algorithm 1, respectively. Note that the implementation of isObstacleBetween() depends on the type of the regions as a determination of an obstacle between two disk regions or between a disk and a point (disk region with zero radius for the obstacle's waypoint), as depicted in Fig. 5.

The proposed improvement procedure is a relatively straightforward adjustment of the waypoints within the disk-shaped regions. The procedure has been applied to the existing solutions of the $\text{CETSP}_{\text{obs}}$ and examined empirically. The results are reported in the following section.

Algorithm 1: `Post-Optimization` of the given $\text{CETSP}_{\text{obs}}$ solution $(\Sigma, \mathcal{P}, \mathcal{Q})$

Input: $\mathcal{S} = \{S_1, \ldots, S_n\}$ – a set of the regions.
Input: $\mathcal{O} = \{O_1, \ldots, O_m\}$ – a set of the obstacles.
Input: $(\Sigma, \mathcal{P}, \mathcal{Q})$ – Σ is a sequence of visits to $\mathcal{S}$ with the corresponding waypoints $\mathcal{P}$ and obstacles' points $\mathcal{Q}$.
Output: $(\Sigma, \mathcal{P}_{\text{opt}}, \mathcal{Q})$ – optimized solution.

1 $\mathcal{S}' \leftarrow ()$ `// Regions ordered by` Σ
2 **for** σ_i `in` Σ **do**
3 $\mathcal{S}' \leftarrow \text{insert}(\mathcal{S}', S_{\sigma_i})$
4 **for** l `in` $0 : k_{\sigma_i}$ **do**
5 $\mathcal{S}' \leftarrow \text{insert}(\mathcal{S}', S(\boldsymbol{c} = \boldsymbol{q}^l_{\sigma_i}; \delta = 0))$ `// Insert obstacle's point as a new region with zero radius.`
6 $\mathcal{M} \leftarrow \text{createModel}(\mathcal{S}')$ `// According to Model 1 with decision variables` $\boldsymbol{x}$
7 **forall** consecutive regions $(S_i, S_{i+1}) \in \mathcal{S}'$ `and` $O_j \in \mathcal{O}$ **do**
8 **if** isObstacleBetween$((S_i, S_{i+1}), O_j)$ **then**
9 $\mathcal{M} \leftarrow \text{addConstraint}(\mathcal{M}, \boldsymbol{d}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i)$
10 **for** l `in` $1 : l_j$ **do**
11 $\mathcal{M} \leftarrow \text{addConstraint}(\mathcal{M}, -d_{i,2}\, o^l_{j,1} + d_{i,1}\, o^l_{j,2} + d_{i,2}\, x_{i,1} - d_{i,1}\, x_{i,2} \leq M\, y_{i,j}$
12 $\mathcal{M} \leftarrow \text{addConstraint}(\mathcal{M}, -d_{i,2}\, o^l_{j,1} + d_{i,1}\, o^l_{j,2} + d_{i,2}\, x_{i,1} - d_{i,1}\, x_{i,2} \geq -M\,(1 - y_{i,j}))$
13 $\mathcal{P}_{\text{opt}} \leftarrow \text{solveMINLP}(\mathcal{M})$ `// Extract the optimized disks' waypoints`
14 **return** $(\Sigma, \mathcal{P}_{\text{opt}}, \mathcal{Q})$



(a) Two disk regions. (b) Disk with zero radius of the obstacle's waypoint and disk region.

Fig. 5. Visualization of the detecting obstacles between two consecutive regions. For two disk regions (left), the tangents are determined from the connection of the disks' centers. The disk has zero radius for the obstacle waypoint; thus, tangents are determined from the cone. Each obstacle's vertice must be on one side of the tangents, ensuring no obstacle between the regions.

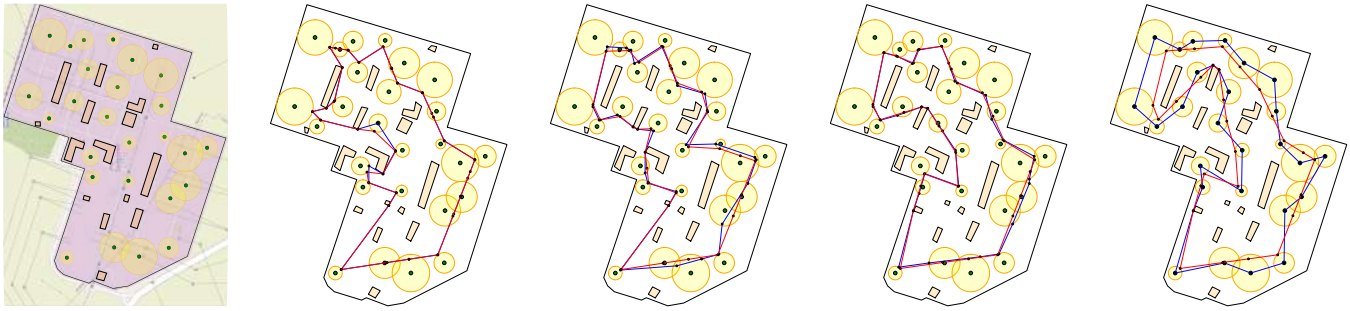
V. Empirical Evaluation

The proposed `Post-Optimization` procedure has been evaluated with the existing GLNSC [28], SOM [27], and GTSP [18] adjusted as described in Section III. All the methods are examined with and without the `Post-Optimization` procedure. The optimized solutions are denoted as GLNSC^+, SOM^+, and GTSP^+. The evaluation has been performed for a set of 32 randomly generated instances of the $\text{CETSP}_{\text{obs}}$, and one instance based on a real data collection scenario using a wheeled vehicle in an electrical substation depicted in Fig. 6. Each instance is named `t`n`_`x, where $n \in \{5, 6, 7, 8, 9, 10, 26\}$ denotes the number of regions, and x denotes the instance label, where $x \in \{1, \ldots, 6\}$ for $n = 5$ and $n = 8$, and $x \in \{1, \ldots, 5\}$ for each $n \in \{6, 7, 9, 10\}$. The SOM-based and GTSP-based solvers depend on the discretization l, selected to $l = 6$ providing the best trade-off between the computational requirements

TABLE I

PERFORMANCE INDICATORS OF THE EXAMINED $\text{CETSP}_{\text{obs}}$ SOLVERS INSTANCES AGGREGATED BY THE NUMBER OF NODES n.

Instance	GLNSC [28]		GLNSC$^+$		SOM ($l=6$) [27]		SOM$^+$ ($l=6$)		GTSP ($l=6$)		GTSP$^+$ ($l=6$)		GTSP ($l=1$)		GTSP$^+$ ($l=1$)	
	%PDB	%PDM	%PDB	%PDM	%PDB	%PDM	%PDB	%PDM	%PDB	%PDM	%PDB	%PDM	%PDB	%PDM	%PDB	%PDM
t5	**0.00**	32.75	**0.00**	**31.98**	1.66	35.49	0.08	32.91	1.37	33.98	**0.00**	32.67	15.38	51.96	**0.00**	33.13
t6	**0.00**	38.35	**0.00**	**38.33**	0.72	40.97	**0.00**	38.76	0.40	38.94	**0.00**	**38.33**	19.59	55.69	**0.00**	38.99
t7	**0.00**	48.91	**0.00**	48.91	0.44	51.91	**0.00**	49.51	0.25	49.39	**0.00**	**48.60**	15.99	66.04	0.16	48.91
t8	**0.00**	27.20	**0.00**	27.20	0.70	25.60	**0.00**	23.98	0.58	22.94	0.08	**22.53**	16.07	38.23	**0.00**	23.59
t9	**0.00**	15.74	**0.00**	15.74	1.67	17.73	**0.00**	16.15	0.96	16.08	**0.00**	**15.45**	18.11	30.40	1.15	16.84
t10	0.56	20.68	**0.00**	20.14	1.69	20.70	0.56	18.93	1.65	19.40	0.56	**18.48**	17.12	40.11	0.56	21.41
t26	**0.00**	1.58	**0.00**	**1.09**	1.95	4.46	**0.00**	2.74	2.46	2.46	1.89	1.89	30.89	30.89	8.47	21.92



(a) Map of an electrical substation with buildings as obstacles.

(b) GLNSC, $\mathcal{L} = 1507.44$, $\mathcal{L}^+ = 1480.38$.

(c) SOM ($l=6$), $\mathcal{L} = 1517.5$, $\mathcal{L}^+ = 1465.32$.

(d) GTSP ($l=6$), $\mathcal{L} = 1470.69$, $\mathcal{L}^+ = 1462.57$.

(e) GTSP ($l=1$), $\mathcal{L} = 1878.87$, $\mathcal{L}^+ = 1557.07$.

Fig. 6. A map of an electrical substation utilized as an real-world $\text{CETSP}_{\text{obs}}$ instance `t26_001`. Solutions of this instance $\text{CETSP}_{\text{obs}}$ instance found by the particular solver are depicted in blue, and the optimized solutions by the `Post-Optimization` procedure are in red.

and solution quality among $l \in \{6, 12, 24, 64, 128\}$. Besides, we include the GTSP solver with $l = 1$ using the disks' centers in the evaluation to highlight the benefit of the proposed `Post-Optimization` to improve the solution quality even for sparse sampling.

The proposed `Post-Optimization` procedure and the GTSP solver are implemented in `Julia v1.7.` using `JuMP` and the MINLP solver `Juniper` [35]. The GLNSC and SOM are implemented in `C++`, and the GLNSC uses SOM-based initialization in a fast mode [28]. Each solver was executed for 20 trials on the `Intel i7-9700` processor running at 3 GHz, and two performance indicators are used for the evaluation. The *solution quality* %PDB for each instance is measured as the percentage deviation from the best overall solution $\mathcal{L}^*_{\text{best}}$ of the best solution $\mathcal{L}^*$ among all performed trials of the particular method $\%\text{PDB} = (\mathcal{L}^* - \mathcal{L}^*_{\text{best}})/\mathcal{L}^*_{\text{best}}\,100\%$. The *solution robustness* %PDM for each instance is measured as the percentage deviation from the best overall solution $\mathcal{L}^*_{\text{best}}$ of the mean solution value $\bar{\mathcal{L}}^*$ among all performed trials of the particular method $\%\text{PDM} = (\bar{\mathcal{L}}^* - \mathcal{L}^*_{\text{best}})/\mathcal{L}^*_{\text{best}}\,100\%$. Besides, we report the computational times T in milliseconds.

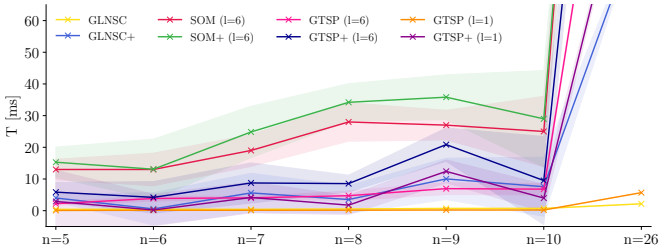


Fig. 7. Median computational times T aggregated from instances with the size n with standard deviation visualized as area around the medians.

The aggregated results are reported in Table I. The results support the expected improvement of the $\text{CETSP}_{\text{obs}}$ solutions and make the examined solvers competitive regarding the solution quality. Although the robustness varies and there is no clear winner, regarding the %PDB, the best-performing method is GLNSC, which can be considered the most complex algorithm. On the other hand, solutions of the very straightforward GTSP with $l = 1$, which can be solved as an instance of the TSP, are significantly improved by the proposed `Post-Optimization`.

The computational requirements of all solvers are exponential with n; see Fig. 7. Here, it is worth noting that the GLNSC method requires preprocessing the input instances by creating supporting structures, which is not included in the presented results, and similarly for the SOM-based solver. However, in both cases, the preprocessing time is competitive with the reported times, but the GLNSC becomes very demanding for larger instances.

TABLE II

STATISTICAL EVALUATION RESULTS OF THE $\text{CETSP}_{\text{obs}}$ SOLVERS.

a_1: GLNSC$^+$ a_2: GLNSC [28]	=	a_1: GLNSC [28] a_2: SOM ($l=6$) [27]	+	a_1: GLNSC [28] a_2: GTSP ($l=6$)	-	a_1: GLNSC [28] a_2: GTSP ($l=1$)	+
a_1: SOM$^+$ ($l=6$) a_2: GLNSC$^+$	=	a_1: SOM ($l=6$) [27] a_2: SOM$^+$ ($l=6$)	+	a_1: SOM ($l=6$) [27] a_2: GTSP ($l=6$)	-	a_1: SOM ($l=6$) [27] a_2: GTSP ($l=1$)	+
a_1: GTSP$^+$ ($l=6$) a_2: GLNSC$^+$	=	a_1: GTSP$^+$ ($l=6$) a_2: SOM$^+$ ($l=6$)	+	a_1: GTSP$^+$ ($l=6$) a_2: GTSP ($l=6$)	+	a_1: GTSP ($l=6$) a_2: GTSP ($l=1$)	+
a_1: GTSP$^+$ ($l=1$) a_2: GLNSC$^+$	-	a_1: GTSP$^+$ ($l=1$) a_2: SOM$^+$ ($l=6$)	-	a_1: GTSP$^+$ ($l=1$) a_2: GTSP$^+$ ($l=6$)	-	a_1: GTSP$^+$ ($l=1$) a_2: GTSP ($l=1$)	+

Symbols $+$, $-$, and $=$ denote the method a_1 provides statistically better, worse, or similar results than the method a_2, respectively.

We further report a statistical comparison of the solvers using the *Wilcoxon Signed Rank Test* [36], where the null hypothesis H_0 is that the solvers a_1 and a_2 provide solutions with statistically similar costs. H_0 is rejected if the obtained p-values are less than 0.001. In the statistical evaluation depicted in Table II, the symbol $=$ denotes a_1 performs similarly to a_2, or $+$ and $-$ if it performs better and worse, respectively, depending on the average solution cost. The results further support the statistically significant improvement of the solutions by the proposed `Post-Optimization` procedure. The GTSP-based solver with $l = 6$ performs best, and SOM is competitive with the GLNSC.

VI. Conclusion

We propose the `Post-Optimization` procedure to improve the heuristic solutions of the $\text{CETSP}_{\text{obs}}$. The procedure is based on the MINLP model to optimize the waypoints, and additional constraints are added to account for the polygonal obstacles. The procedure is employed with three solvers, the GLNSC, currently the only direct method to the $\text{CETSP}_{\text{obs}}$ with disk-shaped regions, and two existing heuristics straightforwardly modified for the disk-shaped regions. Based on the evaluation results, the proposed procedure improves all the found solutions and makes the methods competitive. Furthermore, based on a statistical comparison of the found solutions, the best-performing method is the GTSP with just six samples per each disk region. The sequence of visits to the disks is thus found on the discretized instance of the $\text{CETSP}_{\text{obs}}$, and the MINLP model enables finding the optimal solution of the continuous optimization part of the $\text{CETSP}_{\text{obs}}$ for that sequence. Hence, the proposed `Post-Optimization` represents a groundwork toward an optimal solution for finding the sequence using a branch-and-bound method, similar to the developed solvers to the CETSP without obstacles.

References

[1] S. Alatartsev, S. Stellmacher, and F. Ortmeier, "Robotic Task Sequencing Problem: A Survey," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 2, pp. 279–298, 2015.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton, NJ, USA: Princeton University Press, 2007.

[3] F. Suárez-Ruiz, T. S. Lembono, and Q.-C. Pham, "RoboTSP – A Fast Solution to the Robotic Task Sequencing Problem," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1611–1616.

[4] D. J. Gulczynski, J. W. Heath, and C. C. Price, "The close enough traveling salesman problem: A discussion of several heuristics," in *Perspectives in operations research*. Springer, 2006, pp. 271–283.

[5] V. Krátký, P. Petráček, V. Spurný, and M. Saska, "Autonomous reflectance transformation imaging by a team of unmanned aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2302–2309, 2020.

[6] B. Yuan, M. Orlowska, and S. Sadiq, "On the Optimal Robot Routing Problem in Wireless Sensor Networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 9, pp. 1252–1261, 2007.

[7] O. Tekdas, V. Isler, J. H. Lim, and A. Terzis, "Using mobile robots to harvest data from sensor fields," *IEEE Wireless Communications*, vol. 16, no. 1, pp. 22–28, 2009.

[8] M. Dunbabin and L. Marques, "Robots for Environmental Monitoring: Significant Advancements and Applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.

[9] I. Gentilini, F. Margot, and K. Shimada, "The travelling salesman problem with neighbourhoods: MINLP solution," *Optimization Methods and Software*, vol. 28, no. 2, pp. 364–378, 2013.

[10] K. Vicencio, B. Davis, and I. Gentilini, "Multi-goal path planning based on the generalized Traveling Salesman Problem with neighborhoods," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 2985–2990.

[11] A. Kovács, "Integrated task sequencing and path planning for robotic remote laser welding," *International Journal of Production Research*, vol. 54, no. 4, pp. 1210–1224, 2016.

[12] J. Deckerová, "Generalized routing problems with continuous neighborhoods," Master's thesis, České vysoké učení technické v Praze, 2021.

[13] G. Laporte, A. Asef-Vaziri, and C. Sriskandarajah, "Some Applications of the Generalized Travelling Salesman Problem," *Journal of the Operational Research Society*, vol. 47, no. 12, pp. 1461–1467, 1996.

[14] M. de Berg, J. Gudmundsson, M. J. Katz, C. Levcopoulos, M. H. Overmars, and A. F. van der Stappen, "TSP with neighborhoods of varying size," *Journal of Algorithms*, vol. 57, no. 1, pp. 22–36, 2005.

[15] W. K. Mennell, "Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehicle routing problem, sequence-dependent team orienteering problem," Ph.D. dissertation, University of Maryland, 2009.

[16] S. Alatartsev, M. Augustine, and F. Ortmeier, "Constricting Insertion Heuristic for Traveling Salesman Problem with Neighborhoods," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2013, pp. 2–10.

[17] J. Faigl, "GSOA: Growing Self-Organizing Array - Unsupervised learning for the Close-Enough Traveling Salesman Problem and other routing problems," *Neurocomputing*, vol. 312, pp. 120–134, 2018.

[18] J. Faigl, P. Váňa, and J. Deckerová, "Fast Heuristics for the 3-D Multi-Goal Path Planning Based on the Generalized Traveling Salesman Problem With Neighborhoods," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2439–2446, 2019.

[19] A. Dumitrescu and J. S. B. Mitchell, "Approximation algorithms for TSP with neighborhoods in the plane," *Journal of Algorithms*, vol. 48, no. 1, pp. 135–159, 2003.

[20] E. M. Arkin and R. Hassin, "Approximation algorithms for the geometric covering salesman problem," *Discrete Applied Mathematics*, vol. 55, no. 3, pp. 197–218, 1994.

[21] K. Elbassioni, A. V. Fishkin, and R. Sitters, "Approximation algorithms for the Euclidean traveling salesman problem with discrete and continuous neighborhoods," *International Journal of Computational Geometry & Applications*, vol. 19, no. 2, pp. 173–193, 2009.

[22] B. Behdani and J. C. Smith, "An integer-programming-based approach to the close-enough traveling salesman problem," *INFORMS Journal on Computing*, vol. 26, no. 3, pp. 415–432, 2014.

[23] W. P. Coutinho, R. Q. d. Nascimento, A. A. Pessoa, and A. Subramanian, "A branch-and-bound algorithm for the close-enough traveling salesman problem," *INFORMS Journal on Computing*, vol. 28, no. 4, pp. 752–765, 2016.

[24] H. Huang and A. V. Savkin, "Viable path planning for data collection robots in a sensing field with obstacles," *Computer Communications*, vol. 111, pp. 84–96, 2017.

[25] B. Banyassady, M.-K. Chiu, M. Korman, W. Mulzer, A. Van Renssen, M. Roeloffzen, P. Seiferth, Y. Stein, B. Vogtenhuber, and M. Willert, "Routing in polygonal domains," *Computational Geometry*, vol. 87, p. 101593, 2020.

[26] J. Faigl, M. Kulich, V. Vonásek, and L. Přeučil, "An application of the self-organizing map in the non-Euclidean Traveling Salesman Problem," *Neurocomputing*, vol. 74, no. 5, pp. 671–679, 2011.

[27] J. Faigl, V. Vonásek, and L. Přeučil, "Visiting convex regions in a polygonal map," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1070–1083, 2013.

[28] L. Fanta, "The Close Enough Travelling Salesman Problem in the polygonal domain," Master's thesis, České vysoké učení technické v Praze, 2021.

[29] S. Somhom, A. Modares, and T. Enkawa, "A self-organising model for the travelling salesman problem," *Journal of the Operational Research Society*, pp. 919–928, 1997.

[30] J. Faigl, "On the performance of self-organizing maps for the non-Euclidean Traveling Salesman Problem in the polygonal domain," *Information Sciences*, vol. 181, pp. 4214–4229, 2011.

[31] J. Faigl and G. A. Hollinger, "Autonomous Data Collection Using a Self-Organizing Map," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1703–1715, 2018.

[32] J. Faigl, "Data collection path planning with spatially correlated measurements using growing self-organizing array," *Applied Soft Computing*, vol. 75, pp. 130–147, 2019.

[33] K. Helsgaun, "GLKH," 2013, [cited 29 Jun 2022]. [Online]. Available: http://akira.ruc.dk/~keld/research/GLKH/

[34] M. H. Overmars and E. Welzl, "New methods for computing visibility graphs," in *Symposium on Computational Geometry*. ACM, 1988, pp. 164–171.

[35] O. Kröger, C. Coffrin, H. Hijazi, and H. Nagarajan, "Juniper: An Open-Source Nonlinear Branch-and-Bound Solver in Julia," in *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer International Publishing, 2018, pp. 377–386.

[36] A. Arcuri and L. Briand, "A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering," in *International Conference on Software Engineering*, 2011, pp. 1–10.

Context-Conditional Navigation with a Learning-Based Terrain- and Robot-Aware Dynamics Model

Suresh Guttikonda [*,1,2], Jan Achterhold [*,1], Haolong Li [1], Joschka Boedecker [2], and Joerg Stueckler [1]

***Abstract*— In autonomous navigation settings, several quantities can be subject to variations. Terrain properties such as friction coefficients may vary over time depending on the location of the robot. Also, the dynamics of the robot may change due to, e.g., different payloads, changing the system's mass, or wear and tear, changing actuator gains or joint friction. An autonomous agent should thus be able to adapt to such variations. In this paper, we develop a novel probabilistic, terrain- and robot-aware forward dynamics model, termed TRADYN, which is able to adapt to the above-mentioned variations. It builds on recent advances in meta-learning forward dynamics models based on Neural Processes. We evaluate our method in a simulated 2D navigation setting with a unicycle-like robot and different terrain layouts with spatially varying friction coefficients. In our experiments, the proposed model exhibits lower prediction error for the task of long-horizon trajectory prediction, compared to non-adaptive ablation models. We also evaluate our model on the downstream task of navigation planning, which demonstrates improved performance in planning control-efficient paths by taking robot and terrain properties into account.**

I. INTRODUCTION

Autonomous mobile robot navigation– the robot's ability to reach a specific goal location – has been an attractive research field over several decades, with applications ranging from self-driving cars, warehouse and service robots, to space robotics. In certain situations, e.g. weeding in agricultural robotics or search and rescue operations, robots operate in harsh and unstructured outdoor environments with limited or no human supervision to complete their task. During such missions, the robot needs to navigate over a wide variety of terrains with changing types, such as grass, gravel, or mud with varying slope, friction, and other characteristics. These properties are often hard to fully and accurately model beforehand [1]. Moreover, properties of the robot itself can change during operation due to battery consumption, weight changes, or wear and tear of the robot. Thus, the robot needs to be able to adapt to both changes in robot-specific and terrain-specific properties.

In this work, we develop a novel context-conditional learning approach which captures robot-specific and terrain-specific properties from interaction experience and environment maps. The idea for adaptability to varying robot-specific properties is to learn a deep forward dynamics model which is conditioned on a latent context variable. The context variable is inferred online from observed state transitions. The terrain features are extracted from an environment map and additionally included as conditional variable for the dynamics model.

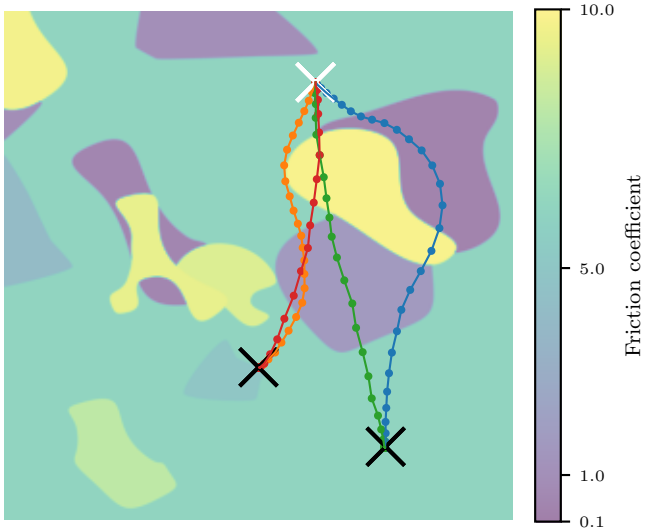


Fig. 1. **Terrain- and robot-aware control-efficient navigation**. We propose a method for control-cost optimal navigation with learned dynamics models. Our method can adapt to varying, unobserved properties of the robot, such as the mass, and spatially varying properties of the terrain, such as the friction coefficient. In the above example of navigating from a single starting point (white cross) to two different goals (black cross), as a result, our method circumvents areas of high friction coefficient and favors areas of low-friction coefficient. As the dissipated energy also depends on the mass of the robot, a heavy robot ($m = 4\,\text{kg}$, blue, orange) is allowed to take longer detours to the goal than a light robot ($m = 1\,\text{kg}$, green, red).

We develop and evaluate our approach in a 2D simulation of a mobile robot modeled as a point mass with unicycle driving dynamics that depend on a couple of robot-specific and terrain-specific parameters. Terrains are defined by regions in the map with varying properties. We demonstrate that our context-aware dynamics model learning approach can capture the varying robot and terrain properties well, while a dynamics model without context-awareness achieves less accurate prediction and planning performance.

In summary, in this paper, we contribute the following:

1) We propose a probabilistic deep forward dynamics model which can adapt to robot- and terrain-specific properties that influence the mobile robot's dynamics.
2) We demonstrate in a 2D simulation environment that these adaptation capabilities are crucial for the predictive performance of the dynamics model.

This work has been supported by Max Planck Society and Cyber Valley. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Jan Achterhold and Haolong Li.
[1]Embodied Vision Group, Max Planck Institute for Intelligent Systems, Tuebingen, Germany, [2]University of Freiburg, Germany
*equal contribution
979-8-3503-0704-7/23/$31.00 ©2023 IEEE

3) The learned context-aware dynamics model is used for robot navigation using model-predictive control. This way, efficient paths can be planned that take robot and terrain properties into account (see Fig. 1).

II. RELATED WORK

Some approaches to terrain-aware navigation use semantic segmentation for determining the category of terrain and use this information to only navigate on segments of traversable terrain [2], [3]. Zhu et al. [4] propose to use inverse reinforcement learning to learn the control costs associated with traversing terrain from human expert demonstrations. These methods, however, do not learn the dynamical properties of the robot on the terrain classes explicitly like our methods.

In BADGR [5] a predictive model is learned of future events based on the current RGB image and control actions, which can be used for planning navigation trajectories. The predicted events are collision, bumpiness, and position. The model is trained from sample trajectories in which the events are automatically labelled. Grigorescu et al. [6] learn a vision-based dynamics model which encodes camera images into a state observation for model-predictive control. Different to our approach, however, the method does not learn a model that can capture a variety of terrain- and robot-specific properties jointly. Siva et al. [7] learn an offset model from the predicted to the actual behavior of the robot from multimodal terrain features determined from camera, LiDAR, and IMU measurements. In Xiao et al. [8] a method for learning an inverse kinodynamics model from inertial measurements is proposed to handle high-speed motion planning on unstructured terrain. Sikand et al. [9] use contrastive learning to embed visual features of terrain with similar traversability properties close in the feature space. The terrain features are used for learning preference-aware path planning. Different to our study, the above approaches do not distinguish terrain- and robot-specific properties and model them concurrently.

Several approaches for learning action-conditional dynamics models have been proposed in the machine learning and robotics literature in recent years. In the seminal work PILCO [10], Gaussian processes are used to learn to predict subsequent states, conditioned on actions. The approach is demonstrated for balancing and swinging up a cart-pole. Several approaches learn latent embeddings of images and predict future latent states conditioned on actions using recurrent neural networks [11], [12], [13], [14]. The models are used in several of these works for model-predictive control and planning. Learning-based dynamics models are also popular in model-based reinforcement learning (see e.g. [15]). Shaj et al. [16] propose action-conditional recurrent Kalman networks which implement observation and action-conditional state-transition models in a Kalman filter with neural networks. While these approaches can model context from past observations in the latent state of the recurrent neural network, some approaches allow for incorporating an arbitrary set of context observations to infer a context variable [17] or a probability distribution thereon [18]. In this paper, we base our approach on the context-conditional dynamics model learning approach in [18] to infer the distribution of a context variable of robot-specific parameters using Neural Processes [19].

III. BACKGROUND

We build our approach on the context-conditional probabilistic neural dynamics model of *Explore the Context* (EtC [18]). In EtC, the basic assumption is that the dynamical system can be formulated by a Markovian discrete-time state-space model

$$\boldsymbol{x}_{n+1} = f(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{\alpha}) + \boldsymbol{\epsilon}_n,\ \boldsymbol{\epsilon}_n \sim \mathcal{N}(0, \boldsymbol{Q}_n), \tag{1}$$

where $\boldsymbol{x}_n$ is the state at timestep n, $\boldsymbol{u}_n$ is the control input, and $\boldsymbol{\alpha}$ is a latent, *unobserved* variable which modulates the dynamics, e.g., robot or terrain parameters. Gaussian additive noise is modeled by $\boldsymbol{\epsilon}_n$, having a diagonal covariance matrix $\boldsymbol{Q}_n$. Not only $\boldsymbol{\alpha}$ is assumed to be unknown, but also the function f itself. To model the system dynamics, EtC thus introduces an approximate forward dynamics model q_{fwd}. To capture the environment-specific properties $\boldsymbol{\alpha}$, the learned dynamics model is conditioned on a latent context variable $\boldsymbol{\beta} \in \mathbb{R}^B$. A probability density on $\boldsymbol{\beta}$ is inferred from interaction experience on the environment, represented by K transitions $(\boldsymbol{x}_+ \leftarrow \boldsymbol{x}, \boldsymbol{u})$ following Eq. (1) and collected in a *context* set $\mathcal{C}^\alpha = \{(\boldsymbol{x}^{(k)}, \boldsymbol{u}^{(k)}, \boldsymbol{x}_+^{(k)})\}_{k=1}^K$. A learned *context encoder* $q_{\text{ctx}}(\boldsymbol{\beta} \mid \mathcal{C}^\alpha)$ infers the density on $\boldsymbol{\beta}$. The target rollout $\mathrm{D}^\alpha = [\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{x}_1, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N-1}, \boldsymbol{x}_N]$ is a trajectory on the environment. Both context set and target rollout are generated on the same environment instance $\boldsymbol{\alpha}$. For a pair of target rollout and context set, the learning objective is to maximize the marginal log-likelihood

$$\log p(\mathrm{D}^\alpha \mid \mathcal{C}^\alpha) = \log \int p(\mathrm{D}^\alpha \mid \boldsymbol{\beta})\, p(\boldsymbol{\beta} \mid \mathcal{C}^\alpha)\, d\boldsymbol{\beta}. \tag{2}$$

Overall, we aim to maximize $\log p(\mathrm{D}^\alpha \mid \mathcal{C}^\alpha)$ in expectation over the distribution of environments Ω_α, and a distribution of pairs of target rollouts and context sets $\Omega_{\mathrm{D}^\alpha, \mathcal{C}^\alpha}$, i.e.

$$\mathbb{E}_{\boldsymbol{\alpha} \sim \Omega_\alpha, (\mathrm{D}^\alpha, \mathcal{C}^\alpha) \sim \Omega_{\mathrm{D}^\alpha, \mathcal{C}^\alpha}} \left[\log p(\mathrm{D}^\alpha \mid \mathcal{C}^\alpha)\right]. \tag{3}$$

The term $p(\mathrm{D}^\alpha \mid \boldsymbol{\beta})$ is modeled by single-step and multi-step prediction factors and reconstruction factors, all implemented by the approximate dynamics model $q_{\text{fwd}}\left(\boldsymbol{x}_n \mid \boldsymbol{x}_0, \boldsymbol{u}_{0:n-1}, \boldsymbol{\beta}\right)$, while $p(\boldsymbol{\beta} \mid \mathcal{C}^\alpha)$ is approximated by $q_{\text{ctx}}(\boldsymbol{\beta} \mid \mathcal{C}^\alpha)$.

Technically, the forward dynamics model is implemented with gated recurrent units (GRU, [20]) in a latent space. The initial state $\boldsymbol{x}_0$ is encoded into a hidden state $\boldsymbol{z}_0$. The control input $\boldsymbol{u}$ and context variable $\boldsymbol{\beta}$ are encoded into feature vectors and passed as inputs to the GRU

$$\boldsymbol{z}_0 = e_x(\boldsymbol{x}_0) \tag{4}$$

$$\mathbf{z}_{n+1} = \mathrm{GRU}\left(\boldsymbol{z}_n, [e_u(\boldsymbol{u}_n), e_\beta(\boldsymbol{\beta})]\right) \tag{5}$$

where e_x, e_u, and e_β are neural network encoders. The (predicted) latent state $\mathbf{z}_n$ is decoded into a Gaussian distribution in the state space

$$\boldsymbol{x}_n \sim \mathcal{N}\left(d_{x,\mu}(\mathbf{z}_n), d_{x,\sigma^2}(\mathbf{z}_n)\right) \tag{6}$$

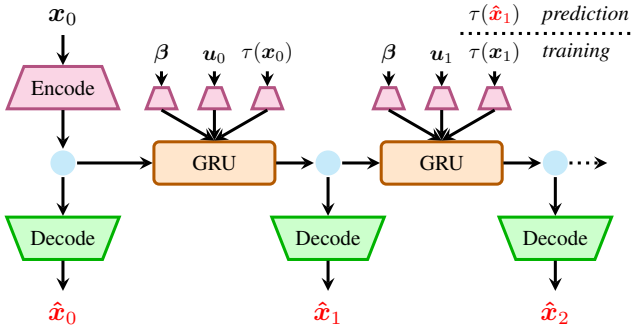


Fig. 2. Architecture of our proposed terrain- and robot-aware forward dynamics model (TRADYN). The initial state of the robot $\boldsymbol{x}_0$ is embedded as hidden state of a gated recurrent unit (GRU) cell. The GRU makes a single-step forward prediction in the latent space using embeddings of the context variable $\boldsymbol{\beta}$, action $\boldsymbol{u}$ and terrain observation $\boldsymbol{\tau}$ as additional inputs. Latent states are mapped to Gaussian distributions on the robot's observation space for decoding. While during training the actual terrain observation $\tau(\boldsymbol{x}_n)$ is used, during prediction, the map τ is queried at predicted robot locations $\tau(\hat{\boldsymbol{x}}_n)$. See Section IV for details.

using neural networks $d_{x,\mu}$, d_{x,σ^2}.

The context encoder gets as input a set of state-action-state transitions $\mathcal{C}^\alpha$ with flexible size K. The context encoder is implemented by first encoding each transition in the context set independently using a transition encoder e_{trans}, and, for permutation invariance, aggregating the encodings using a dimension-wise max operation. This yields the aggregated latent variable $\boldsymbol{z}_\beta$. Lastly, a Gaussian density over the context variable $\boldsymbol{\beta}$ is predicted from the aggregated encodings

$$q_{\text{ctx}}(\boldsymbol{\beta} \mid \mathcal{C}^\alpha) = \mathcal{N}\left(\boldsymbol{\beta}; d_{\beta,\mu}(\boldsymbol{z}_\beta), \operatorname{diag}(d_{\beta,\sigma^2}(\boldsymbol{z}_\beta))\right) \tag{7}$$

with neural network decoders $d_{\beta,\mu}$, d_{β,σ^2}. The network d_{β,σ^2} is designed so that the predicted variance is positive and decreases monotonically when adding context observations.

To form a tractable loss, the marginal log likelihood in Eq. (2) is (approximately, see [21]) bounded using the evidence lower bound

$$\begin{aligned}\log p(\mathrm{D}^\alpha \mid \mathcal{C}^\alpha) \gtrapprox\ & \mathbb{E}_{\boldsymbol{\beta}\sim q_{\text{ctx}}(\boldsymbol{\beta}\mid \mathrm{D}^\alpha \cup \mathcal{C}^\alpha)}\left[\log p(\mathrm{D}^\alpha \mid \boldsymbol{\beta})\right] \\ & - \lambda_{KL}\, \mathrm{KL}\left(q_{\text{ctx}}(\boldsymbol{\beta} \mid \mathrm{D}^\alpha \cup \mathcal{C}^\alpha) \,\|\, q_{\text{ctx}}(\boldsymbol{\beta} \mid \mathcal{C}^\alpha)\right).\end{aligned} \tag{8}$$

similar to Neural Processes [19]. For training the dynamics model and context encoder, the approximate bound in Eq. (8) is maximized by stochastic gradient ascent on empirical samples for target rollouts and context sets. Samples are drawn from trajectories generated on a training set of environments.

By collecting context observations at test time, and inferring $\boldsymbol{\beta}$ using $q_{\text{ctx}}(\boldsymbol{\beta} \mid \mathcal{C}^\alpha)$, the dynamics model $q_{\text{fwd}}(\boldsymbol{x}_n \mid \boldsymbol{x}_0, \boldsymbol{u}_{0:N-1}, \boldsymbol{\beta})$ can adapt to a particular environment instance $\boldsymbol{\alpha}$ (called *calibration*).

IV. METHOD

In the modeling assumption of EtC, changes in the dynamics among different instances of environments are captured in a global latent variable $\boldsymbol{\alpha}$ (see Eq. (1)) which is unobserved. In terrain-aware robot navigation, among different environments, the terrain varies (with the terrain layout captured by α_{terrain}), in addition to robot-specific parameters such as actuator gains (captured by $\boldsymbol{\alpha}_{\text{robot}}$). In principle, both effects can be absorbed into a single latent variable $\boldsymbol{\alpha} = (\alpha_{\text{terrain}}, \boldsymbol{\alpha}_{\text{robot}})$. Here, we make more specific assumptions, and assume the terrain-specific properties to be captured in a state-dependent function $\alpha_{\text{terrain}}(\boldsymbol{x}_n)$.

A. Terrain- and Robot-Aware Dynamics Model

Conclusively, we assume the following environment dynamics

$$\boldsymbol{x}_{n+1} = f(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{\alpha}_{\text{robot}}, \alpha_{\text{terrain}}(\boldsymbol{x}_n)) + \boldsymbol{\epsilon}_n \tag{9}$$

with $\boldsymbol{\epsilon}_n \sim \mathcal{N}(0, \boldsymbol{Q}_n)$ as in Eq. (1). In our case of terrain-aware robot navigation, $\boldsymbol{x}_n$ refers to the robot state at timestep n, $\boldsymbol{u}_n$ are the control inputs, $\boldsymbol{\alpha}_{\text{robot}}$ captures (unobserved) properties of the robot (mass, actuator gains), and $\alpha_{\text{terrain}}(\boldsymbol{x}_n)$ captures the spatially dependent terrain properties (e.g., friction). While we assume α_{terrain} to be unobserved, we assume the existence of a *known map* of *terrain features* $\tau_{\text{terrain}}(\boldsymbol{x}_n)$, which can be queried at any $\boldsymbol{x}_n$ to estimate the value of $\alpha_{\text{terrain}}(\boldsymbol{x}_n)$. Exemplarily, τ_{terrain} may yield visual terrain observations, which relate to friction coefficients.

As we retain the assumption of EtC that $\boldsymbol{\alpha}_{\text{robot}}$ is not directly observable, we condition the multi-step forward dynamics model on the latent variable $\boldsymbol{\beta}$. In addition, we condition on observed terrain features $\boldsymbol{\tau}_{0:n-1}$, i.e.,

$$\hat{\boldsymbol{x}}_n \sim q_{\text{fwd}}(\boldsymbol{x}_n \mid \boldsymbol{x}_0, \boldsymbol{u}_{0:n-1}, \boldsymbol{\beta}, \boldsymbol{\tau}_{0:n-1}). \tag{10}$$

We obtain $\boldsymbol{\tau}_{0:n-1}$ differently for training and prediction. During training, we evaluate τ at ground-truth states, i.e. $\boldsymbol{\tau}_0 = \tau(\boldsymbol{x}_0), \boldsymbol{\tau}_1 = \tau(\boldsymbol{x}_1)$, etc. During prediction, we do not have access to ground-truth states, and obtain $\boldsymbol{\tau}_{0:n-1}$ autoregressively from predictions as $\boldsymbol{\tau}_0 = \tau(\boldsymbol{x}_0), \boldsymbol{\tau}_1 = \tau(\hat{\boldsymbol{x}}_1)$, etc.

To capture terrain-specific properties, we extend EtC as follows. We introduce an additional encoder e_τ which encodes a terrain feature $\boldsymbol{\tau}$. The encoded value is passed as input to the GRU, such that Eq. (5) is updated to

$$\mathbf{z}_{n+1} = \mathrm{GRU}\left(\boldsymbol{z}_0, [e_\tau(\boldsymbol{\tau}_n), e_u(\boldsymbol{u}_n), e_\beta(\boldsymbol{\beta})]\right). \tag{11}$$

Also, the context set is extended to contain terrain features

$$\mathcal{C}^\alpha = \{(\boldsymbol{x}^{(k)}, \tau(\boldsymbol{x}^{(k)}), \boldsymbol{u}^{(k)}, \boldsymbol{x}_+^{(k)}, \tau(\boldsymbol{x}_+^{(k)}))\}_{k=1}^K. \tag{12}$$

We refer to Fig. 2 for a depiction of our model.

For each training example, the context set size K is uniformly sampled in $\{0, \ldots, 50\}$. The target rollout length is $N = 50$. As in EtC [18], we set $\lambda_{KL} = 5$. The dimensionality of the latent variable $\boldsymbol{\beta}$ is 16. For details on the networks $(e_u, e_\beta, d_{x,\mu}, d_{x,\sigma^2}, e_{\text{trans}}, d_{\beta,\mu}, d_{\beta,\sigma^2})$ we refer to [18], as we strictly follow the architecture described therein. The additional encoder network we introduce, e_β, follows the architecture of e_τ and e_u. It contains a single hidden layer with 200 units and ReLU activations, and an output layer which maps to an embedding of dimensionality 200.

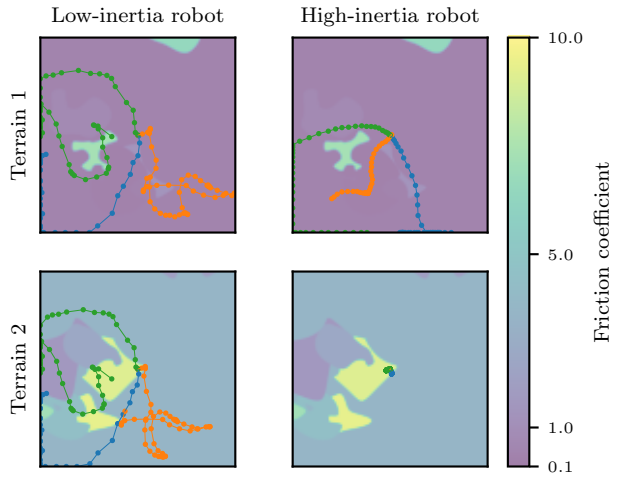


Fig. 3. Exemplary rollouts (length 50) on two different terrain layouts (rows) and for two exemplary robot configurations (low-inertia, high-inertia) (columns). Rollouts start from the center; actions are sampled time-correlated. The low-inertia robot has minimal mass $m = 1$ and maximal control gains $k_{\text{throttle}} = 1000$, $k_{\text{steer}} = \pi/4$. The high-inertia robot has maximal mass $m = 4$ and minimal control gains $k_{\text{throttle}} = 500$, $k_{\text{steer}} = \pi/8$. Equally colored trajectories (●, ●, ●) correspond to identical sequences of applied actions. See Section V-A for details.

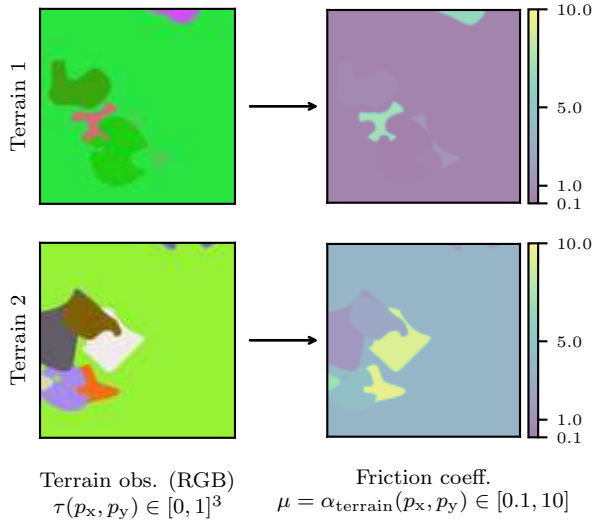


Fig. 4. Relationship of RGB terrain features $\boldsymbol{\tau}$ (left column) to friction coefficient μ (right column). See Section V-A.2 for details.

B. Path Planning and Motion Control

We use TRADYN in a model-predictive control setup. The model q_{fwd} yields state predictions $\hat{\boldsymbol{x}}_{1:H}$ for an initial state $\boldsymbol{x_0}$ and controls $\boldsymbol{u}_{0:H-1}$. For calibration, i.e., inferring $\boldsymbol{\beta}$ from a context set $\mathcal{C}$ with the context encoder q_{ctx}, calibration transitions are collected on the target environment prior to planning. This allows adapting to varying robot parameters. The predictive terrain feature lookup (see Fig. 2) with $\tau(\boldsymbol{x})$ allows adapting to varying terrains. We use the Cross-Entropy Method (CEM [22]) for planning. We aim to reach the target position with minimal throttle control energy, given by the sum of squared throttle commands during navigation. This gives rise to the following planning objective, which penalizes high throttle control energy and a deviation of the robot's terminal position to the target position $\boldsymbol{p}^*$:

$$J(\boldsymbol{u}_{0:H-1}, \hat{\boldsymbol{x}}_{1:H}) = \frac{1}{2}\sum_{n=0}^{H-1} u^2_{\text{throttle},n} + ||[\hat{p}_{\text{x},H}, \hat{p}_{\text{y},H}]^\top - \boldsymbol{p}^*||_2^2. \quad (13)$$

In our CEM implementation, we normalize the distance term in Eq. (13) to have zero mean and unit variance over all CEM candidates, to trade-off control- and distance cost terms even under large terrain variations. At each step, we only apply the first action and plan again from the resulting state in a receding horizon scheme.

V. EXPERIMENTS

A. Simulation environment

1) Simulated Robot Dynamics: We perform experiments in a 2D simulation with a unicycle-like robot setup where the continuous time-variant 2D dynamics with position $\mathbf{p} = [p_x, p_y]^\top$, orientation φ', and directional velocity v, for control input $\boldsymbol{u} = [u_{\text{throttle}}, u_{\text{steer}}]^\top \in [-1, 1]^2$, are given by

$$\begin{aligned} \dot{\boldsymbol{p}}(t) &= \begin{bmatrix}\cos\varphi' & \sin\varphi'\end{bmatrix}^\top v(t) \\ \dot{v}(t) &= \frac{1}{m}(F_{\text{throttle}} + F_{\text{fric}}) \\ F_{\text{throttle}} &= u_{\text{throttle}}\, k_{\text{throttle}} \\ F_{\text{fric}} &= -\operatorname{sign}(v(t))\, \mu\, m\, g. \end{aligned} \quad (14)$$

As our method does not use continuous-time observations, but only discrete-time samplings with stepsize $\Delta_T = 0.01\,\text{s}$, we approximate the state evolution between two timesteps as follows. First, we apply the change in angle as $\varphi' = \varphi(t + \Delta_T) = \varphi(t) + u_{\text{steer}}\, k_{\text{steer}}$. We then query the terrain friction coefficient μ at the position $\boldsymbol{p}(t)$. With the friction coefficient μ and angle φ' we compute the evolution of position and velocity with Eq. (14). The existence of the friction term in Eq. (14) requires an accurate integration, which is why we solve the initial value problem in Eq. (14) numerically using an explicit Runge Kutta (RK45) method, yielding $\boldsymbol{p}(t + \Delta_T)$ and $v(t + \Delta_T)$. Our simulated system dynamics are deterministic. To avoid discontinuities, we represent observations of the above system as $\boldsymbol{x}(t) = [p_x(t), p_y(t), v(t), \cos\varphi(t), \sin\varphi(t)]^\top$. We use $g = 9.81\,\text{m}\,\text{s}^{-2}$ as gravitational acceleration. Positions $\boldsymbol{p}(t)$ are clipped to the range $[0, 1]$ m; the directional velocity $v(t)$ is clipped to $[-5, 5]\ \text{m}\,\text{s}^{-1}$. The friction coefficient $\mu = \alpha_{\text{terrain}}(p_x, p_y) \in [0.1, 10]$ depends on the terrain layout α_{terrain} and the robot's position. The mass m and control gains $k_{\text{throttle}}, k_{\text{steer}}$ are robot-specific properties, we refer to Table I for their value ranges.

2) Terrain layouts: To simulate the influence of varying terrain properties on the robots' dynamics, we programmatically generate 50 terrain layouts for training the dynamics model and 50 terrain layouts for testing (i.e., in prediction- and planning evaluation). For generating terrain k, we first generate an unnormalized feature map $\hat{\tau}^{(k)}$, from which we

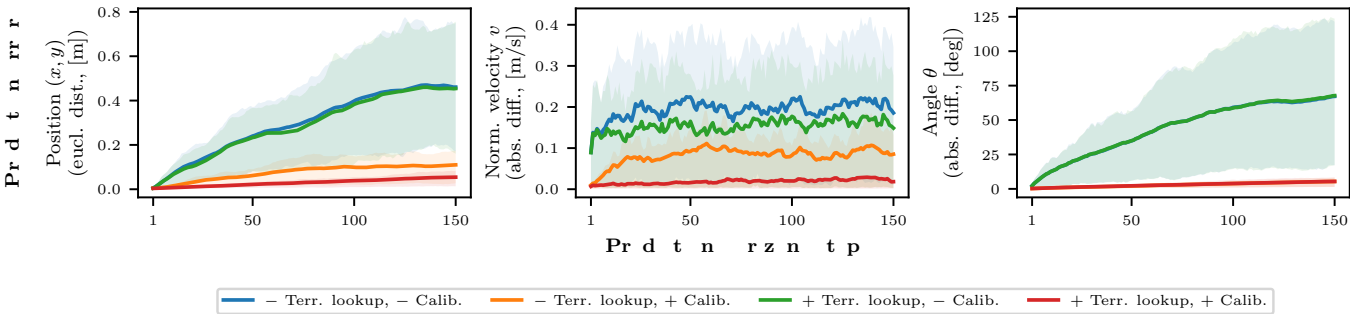


Fig. 5. Prediction error evaluation for the proposed model and its ablations (no terrain lookup / no calibration), plotted over the prediction horizon (number of prediction steps). From left to right: Positional error (euclidean distance), velocity error (absolute difference), angular error (absolute difference). Depicted are the mean and 20%, 80% percentiles over 150 evaluation rollouts for 5 independently trained models per model variant. Our approach with terrain lookup and calibration clearly outperforms the other variants in position and velocity prediction (left and center panel). For predicting the angle (right panel), terrain friction is not relevant, which is why the terrain lookup brings no advantage. However, calibration is important for accurate angle prediction. See Section V-C for details.

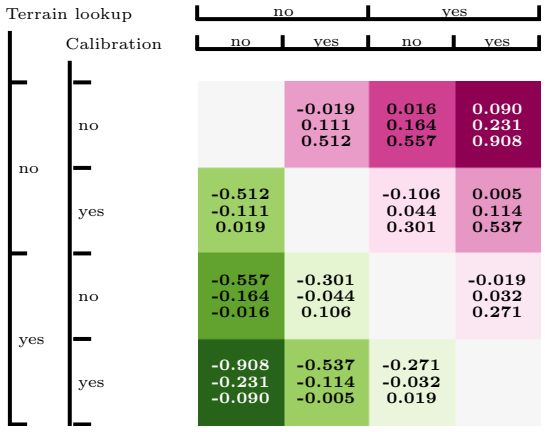
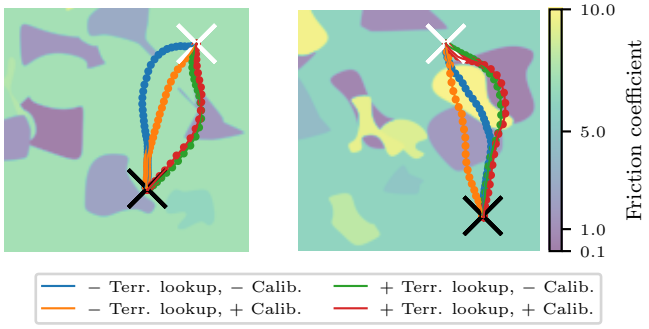


Fig. 6. Comparison of model variants with and without terrain lookup and calibration. $E^{v}_{k,i}$ denotes the throttle control energy for method v on navigation task $k \in \{1,\dots,150\}$ for a trained model with seed $i \in \{1,\dots,5\}$. We show statistics (20% percentile, median, 80% percentile) on the set of pairwise comparisons of control energies $\{E^{\text{row}}_{k,i_1} - E^{\text{col}}_{k,i_2} \mid \forall k \in \{1,\dots,K\}, i_1 \in \{1,\dots,5\}, i_2 \in \{1,\dots,5\}\}$. Significant ($p < 0.05$) results are printed **bold** (see Section V-D). Exemplarily, both performing terrain lookup and calibration (last row) yields navigation solutions with significantly lower throttle control energy (negative numbers) compared to all other methods (columns). See Section V-D for details.



Variant	left terrain		right terrain	
	Thr. ctrl. energy	Target dist [mm]	Thr. ctrl. energy	Target dist [mm]
● −T, −C	4.08	7.82	3.96	2.69
● −T, +C	3.47	4.89	2.78	4.87
● +T, −C	2.01	4.57	1.88	5.14
● +T, +C	2.02	5.55	1.43	6.66

Fig. 7. Exemplary navigation trajectories and their associated throttle control energy and final distance to the target (see table). The robot starts at the white cross, the goal is marked by a black cross. With terrain lookup (● +T, -C and ● +T, +C), our method circumvents areas of high friction coefficient (i.e., high energy dissipation), resulting in lower throttle control energy (see table). Enabling calibration (+C) further reduces throttle control energy on the right terrain. See Section V-D for details.

compute $\alpha^{(k)}_{\text{terrain}}$ and the normalized feature map $\tau^{(k)}$. The unnormalized feature map is represented by a 2D RGB image of size $460\,\text{px} \times 460\,\text{px}$. For its generation, first, a background color is randomly sampled, followed by sequentially placing randomly sampled patches with cubic bezier contours. The color value $(r,g,b) \in \{0,\dots,255\}^3$ at each pixel maps to the friction coefficient $\mu = \alpha_{\text{terrain}}(p_{\text{x}}, p_{\text{y}})$ through bitwise left-shifts $\ll$ as

$$\begin{aligned} \eta &= ((r \ll 16) + (g \ll 8) + b)/(2^{24}-1) \\ \alpha_{\text{terrain}}(p_{\text{x}}, p_{\text{y}}) &= 0.1 + (10 - 0.1)\eta^2. \end{aligned} \tag{15}$$

The agent can observe the normalized terrain color $\tau^{(k)}(p_{\text{x}}, p_{\text{y}}) \in [0,1]^3$ with $\tau^{(k)}(p_{\text{x}}, p_{\text{y}}) = \hat{\tau}^{(k)}(p_{\text{x}}, p_{\text{y}})/255$, and can query $\tau^{(k)}(p_{\text{x}}, p_{\text{y}})$ at arbitrary $p_{\text{x}}, p_{\text{y}}$. The simulator has direct access to $\mu = \alpha^{(k)}_{\text{terrain}}(p_{\text{x}}, p_{\text{y}})$. We denote the training set of terrains $\mathcal{A}_{\text{train}} = \{\alpha^{(k)}_{\text{terrain}} \mid k \in \{1,\dots,50\}\}$ and the test set of terrains $\mathcal{A}_{\text{test}} = \{\alpha^{(k)}_{\text{terrain}} \mid k \in \{51,\dots,100\}\}$. We refer to Fig. 4 for a visualization of two terrains and the related friction coefficients.

3) Environment instance: The robot's dynamics depends on the terrain α_{terrain} as it is the position-dependent friction coefficient, and the robot-specific parameters $\boldsymbol{\alpha}_{\text{robot}} = (m, k_{\text{throttle}}, k_{\text{steer}})$. A fixed tuple $(\alpha_{\text{terrain}}, \boldsymbol{\alpha}_{\text{robot}})$ forms an environment *instance*.

4) Trajectory generation: We require the generation of trajectories at multiple places of our algorithm for training and evaluation: To generate training data, to sample candidate trajectories for the cross-entropy planning method, to generate calibration trajectories, and to generate trajectories for evaluating the prediction performance. One option would be to generate trajectories by independently sampling actions

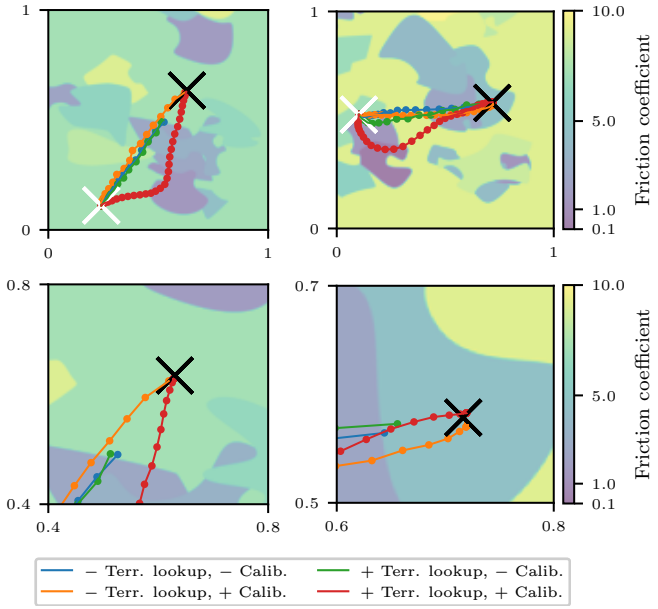


Fig. 8. Failure cases for the non-calibrated models. The top row shows the full terrain of extent $[0, 1]$m. The bottom row is zoomed around the goal. In the cases shown, planning with the non-calibrated models does not succeed in reaching the goal marked by the black cross within the given step limit of 50 steps, in contrast to the calibrated models. See Section V-D for details.

TABLE I

ROBOT-SPECIFIC PROPERTIES.

Property	Min.	Max.
Mass m [kg]	1	4
Throttle gain k_{throttle}	500	1000
Steer gain k_{steer}	$\pi/8$	$\pi/4$

from a Gaussian distribution at each timestep. However, this Brownian random walk significantly limits the space traversed by such trajectories [23]. To increase the traversed space, [23] propose to use time-correlated (colored) noise with a power spectral density $\text{PSD}(f) \propto \frac{1}{f^\omega}$, where f is the frequency. We use $\omega = 0.5$ in all our experiments.

5) Exemplary rollouts: We visualize exemplary rollouts on different terrains and with different robot parametrizations in Fig. 3. We observe that both the terrain-dependent friction coefficient μ, as well as the robot properties, have a significant influence on the shape of the trajectories, highlighting the importance of a model to be able to adapt to these properties.

B. Model training

We train our proposed model on a set of precollected trajectories on different terrain layouts and robot parametrizations. First, we sample a set of 10000 unique terrain layout / robot parameter settings to generate training trajectories. For validation, a set of additional 5000 settings is used. On each setting, we generate two trajectories, used later during training to form the target rollout D^α and context set $\mathcal{C}^\alpha$, respectively. Terrain layouts are sampled uniformly from the training set of terrains, i.e. $\mathcal{A}_{\text{train}}$. Robot parameters are sampled uniformly from the parameter ranges given in Table I. The robots' initial state $\boldsymbol{x_0} = [p_{\text{x},0}, p_{\text{y},0}, v_0, \varphi_0]^\top$ is uniformly sampled from the ranges $p_{\text{x},0}, p_{\text{y},0} \in [0, 1]$, $v_0 \in [-5, 5]$, $\varphi_0 \in [0, 2\pi]$. Each trajectory consists of 100 applied actions and the resulting states. We use time-correlated (colored) noise to sample actions (see previous paragraph). We follow the training procedure described in [18].

1) Model ablation: As an ablation to our model, we only input the terrain features $\boldsymbol{\tau}$ at the current and previously visited states of the robot as terrain observations, but do not allow for terrain lookups in a map at future states during prediction. We will refer to this ablation as *No(−) terrain lookup* in the following.

C. Prediction evaluation

In this section we evaluate the prediction performance of our proposed model. To this end, we generate 150 test trajectories of length 150, on the *test* set of terrain layouts $\mathcal{A}_{\text{test}}$. Robot parameters are uniformly sampled as during data collection for model training. The robot's initial position is sampled from $[0.1, 0.9]^2$, the orientation from $[0, 2\pi]$. The initial velocity is fixed to 0. Actions are sampled with a time-correlated (colored) noise scheme. In case the model is *calibrated*, we additionally collect a small trajectory for each trajectory to be predicted, consisting of 10 transitions, starting from the same initial state $\boldsymbol{x_0}$, but with different random actions. Transitions from this trajectory form the context set $\mathcal{C}$, which is used by the context encoder $q_{\text{ctx}}(\boldsymbol{\beta} \mid \mathcal{C})$ to output a belief on the latent context variable $\boldsymbol{\beta}$. In case the model is *not calibrated*, the distribution is given by the context encoder for an empty context set, i.e. $q_{\text{ctx}}(\boldsymbol{\beta} \mid \mathcal{C} = \{\})$. We evaluate two model variants; first, our proposed model which utilizes the terrain map $\tau(p_{\text{x}}, p_{\text{y}})$ for lookup during predictions, and second, a model for which the terrain observation is concatenated to the robot observation. All results are reported on 5 independently trained models. Figure 5 shows that our approach with terrain lookup and calibration clearly outperforms the other variants in position and velocity prediction. As the evolution of the robot's angle is independent of terrain friction, for angle prediction, only performing calibration is important.

D. Planning evaluation

Aside the prediction capabilities of our proposed method, we are interested whether it can be leveraged for efficient navigation planning. To evaluate the planning performance, we generate 150 navigation tasks, similar to the above prediction tasks, but with an additional randomly sampled target position $\boldsymbol{p}^* \in [0.1, 0.9]^2$ for the robot. We perform receding horizon control as described in Section IV-B.

Again, we evaluate four variants of our model. We compare models with and without the ability to perform terrain lookups. Additionally, we evaluate the influence of calibration, by either collecting 10 additional calibration transitions for each planning task setup, or not collecting any calibration transition ($\mathcal{C} = \{\}$), giving four variants in total.

TABLE II

DISTANCE TO GOAL (MEDIAN AND 20% / 80% PERCENTILES) AND FAILURE RATE FOR 5 cm DISTANCE THRESHOLD TO GOAL. VARIANTS ARE WITH/WITHOUT TERRAIN LOOKUP (±T) AND WITH/WITHOUT CALIBRATION (±C). OUR FULL APPROACH (+T, +C) YIELDS BEST PERFORMANCE IN REACHING THE GOAL AND SUCCEEDS IN ALL RUNS.

Variant	Euclidean distance to goal [mm] P20	median	P80	Failed tasks
● −T, −C	3.00	5.19	8.67	6/750
● −T, +C	3.13	5.23	7.65	**0/750**
● +T, −C	2.33	4.22	6.49	14/750
● +T, +C	**2.16**	**3.85**	**5.61**	**0/750**

As we have trained five models with different seeds, over all models, we obtain 750 navigation results. We count a navigation task as *failed* if the final Euclidean distance to the goal exceeds 5 cm.

We evaluate the efficiency of the navigation task solution by the sum of squared throttle controls over a fixed trajectory length of $N = 50$ steps, which we denote as $E = \sum_{n=0}^{N-1} u_{\text{throttle},n}^2$. We introduce super- and subscripts $E_{k,i}^{v}$ to refer to model variant v, planning task index k and model seed i. Please see Figs. 6 and 7 for results comparing the particular variants. For pairwise comparison of control energies E we leverage the Wilcoxon signed-rank test with a p-value of 0.05. We can conclude that, regardless of calibration, performing terrain lookups yields navigation solutions with significantly lower throttle control energy. The same holds for performing calibration, regardless of performing terrain lookups. Lowest control energy is obtained for both performing calibration and terrain lookup.

We refer to Table II for statistics on the number of failed tasks and final distance to the goal. As can be seen, our terrain- and robot-aware approach yields overall best performance in Euclidean distance to the goal and succeeds in all runs in reaching the goal. Planning with non-calibrated models variants occasionally fails, i.e., the goal is not reached. We show such failure cases in Fig. 8.

VI. CONCLUSIONS

In this paper, we propose a forward dynamics model which can adapt to variations in unobserved variables that govern the system's dynamics such as robot-specific properties, as well as to spatial variations. We train our model on a simulated unicycle-like robot, which has varying mass and actuator gains. In addition, the robot's dynamics are influenced by instance-wise and spatially varying friction coefficients of the terrain, which are only indirectly observable through terrain observations. In 2D simulation experiments, we demonstrate that our model can successfully cope with such variations through calibration and terrain lookup. It exhibits smaller prediction errors compared to model variants without calibration and terrain lookup, and yields solutions to navigation tasks which require lower throttle control energy. In future work, we plan to extend our novel learning-based approach for real-world robot navigation problems with partial observability, noisy state transitions, and noisy observations.

REFERENCES

[1] R. Sonker and A. Dutta, "Adding terrain height to improve model learning for path tracking on uneven terrain by a four wheel robot," *IEEE Robotics Autom. Lett.*, 2021.

[2] A. Valada, J. Vertens, A. Dhall, and W. Burgard, "Adapnet: Adaptive semantic segmentation in adverse environmental conditions," in *IEEE ICRA*, 2017.

[3] K. Yang, L. M. Bergasa, E. Romera, R. Cheng, T. Chen, and K. Wang, "Unifying terrain awareness through real-time semantic segmentation," in *IEEE Intelligent Vehicles Symposium*, 2018.

[4] Z. Zhu, N. Li, R. Sun, H. Zhao, and D. Xu, "Off-road autonomous vehicles traversability analysis and trajectory planning based on deep inverse reinforcement learning," *CoRR*, vol. abs/1909.06953, 2019.

[5] G. Kahn, P. Abbeel, and S. Levine, "BADGR: an autonomous self-supervised learning-based navigation system," *IEEE Robotics Autom. Lett.*, 2021.

[6] S. Grigorescu, C. Ginerica, M. Zaha, G. Macesanu, and B. Trasnea, "LVD-NMPC: A learning-based vision dynamics approach to nonlinear model predictive control for autonomous vehicles," *International Journal of Advanced Robotic Systems*, 2021.

[7] S. Siva, M. B. Wigness, J. G. Rogers, and H. Zhang, "Enhancing consistent ground maneuverability by robot adaptation to complex off-road terrains," in *CoRL*, 2021.

[8] X. Xiao, J. Biswas, and P. Stone, "Learning inverse kinodynamics for accurate high-speed off-road navigation on unstructured terrain," *IEEE Robotics Autom. Lett.*, 2021.

[9] K. S. Sikand, S. Rabiee, A. Uccello, X. Xiao, G. Warnell, and J. Biswas, "Visual representation learning for preference-aware path planning," in *IEEE ICRA*, 2022.

[10] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *ICML*, 2011.

[11] I. Lenz, R. A. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control," in *Robotics: Science and Systems*, 2015.

[12] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *NeurIPS*, 2015.

[13] C. Finn, I. J. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *NeurIPS*, 2016.

[14] D. Hafner, T. P. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *ICML*, 2019.

[15] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *IEEE ICRA*, 2018.

[16] V. Shaj, P. Becker, D. Büchler, H. Pandya, N. van Duijkeren, C. J. Taylor, M. Hanheide, and G. Neumann, "Action-conditional recurrent kalman networks for forward and inverse dynamics learning," in *CoRL*, 2020.

[17] K. Lee, Y. Seo, S. Lee, H. Lee, and J. Shin, "Context-aware dynamics model for generalization in model-based reinforcement learning," in *ICML*, 2020.

[18] J. Achterhold and J. Stueckler, "Explore the Context: Optimal data collection for context-conditional dynamics models," in *AISTATS*, 2021.

[19] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh, "Neural Processes," *CoRR*, vol. abs/1807.01622, 2018.

[20] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP*, 2014.

[21] T. A. Le, H. Kim, M. Garnelo, D. Rosenbaum, J. Schwarz, and Y. W. Teh, "Empirical evaluation of neural process objectives," in *Third Workshop on Bayesian Deep Learning*, 2018.

[22] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and Computing in Applied Probability*, 1999.

[23] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stueckler, M. Rolínek, and G. Martius, "Sample-efficient cross-entropy method for real-time planning," in *CoRL*, 2020.

Social Robot Navigation through Constrained Optimization: a Comparative Study of Uncertainty-based Objectives and Constraints

Timur Akhtyamov[1], Aleksandr Kashirin[1], Aleksey Postnikov[1,2], Gonzalo Ferrer[1]

***Abstract*— This work is dedicated to the study of how uncertainty estimation of the human motion prediction can be embedded into constrained optimization techniques, such as Model Predictive Control (MPC) for the social robot navigation. We propose several cost objectives and constraint functions obtained from the uncertainty of predicting pedestrian positions and related to the probability of the collision that can be applied to the MPC, and all the different variants are compared in challenging scenes with multiple agents. The main question this paper tries to answer is: what are the most important uncertainty-based criteria for social MPC? For that, we evaluate the proposed approaches with several social navigation metrics in an extensive set of scenarios of different complexity in reproducible synthetic environments. The main outcome of our study is a foundation for a practical guide on when and how to use uncertainty-aware approaches for social robot navigation in practice and what are the most effective criteria.**

I. Introduction

Social robot navigation remains a difficult problem since navigating in a socially acceptable manner, in a dynamic and complex environment, is often unpredictable and uncertain mostly due to its human nature. This involves not only avoiding obstacles but also interacting with humans in a way that is natural, safe, and comfortable.

One of the main challenges is that human behaviour is ambiguous and difficult to predict. People may move in unexpected ways, change direction suddenly, or give non-verbal cues that are difficult for robots to interpret. Fortunately, with modern techniques now it is possible to predict accurately and with a correct measure of the inherent uncertainty [1]. In addition, social norms and conventions vary between cultures and contexts, making it difficult to develop a one-size-fits-all approach.

Finally, safety is a critical concern in social navigation, as robots must avoid collisions and other hazards while navigating in close proximity to humans. This requires advanced planning and control algorithms that can take into account the robot's own capabilities and limitations, as well as those of the people in the environment. It is unclear which are the dominant criteria in social robot navigation, and our initial hypothesis is that accurate uncertainty prediction should play a fundamental role on the social navigation task.

MPC is one of the world's industrial standards for the variety of control and planning tasks, especially in robotics. Modern MPC solutions are built on top of the efficient solvers that achieve real-time or near-real-time performance in various deterministic settings. Constrained optimization techniques employed by MPC allow to leverage different navigation objectives and constrains, for instance, distance to goal, probability of collision or deterministic geometric collision constraints.

In this work, we propose to study how pedestrians trajectory prediction uncertainty can be embedded into MPC-based planning via various objectives and constraints derived from the uncertainty in the environment, and how it influences in practice the performance of the controller. The main contributions of the paper are:

- Several uncertainty-unaware and uncertainty-aware MPC designs that incorporate CovarianceNet-based approach [1] for pedestrian trajectory prediction;
- Extensive evaluation of the proposed approaches in simulation environments with practice-oriented conclusions;
- Introduction of the novel simulation environment targeted for social robot navigation tasks.

II. Related Works

A. Social robot navigation approaches

Generally, social robot navigation problem has been studied for several decades, and variety of approaches have been proposed [2]–[4]. Methods based on the enhancement of the classical path planning [5]–[8] are built on top of the algorithms like A*, RRT or RRT*. Adaptivity to the pedestrian dynamics is achieved by using time-based variations of those algorithms, dynamic cost maps that are built using pedestrians motion prediction and socially-aware transition or steering functions.

Optimization-based methods employ advances in nonlinear programming to generate a sequence of safe robot control inputs. These methods first of all include MPC schemes adapted to the social navigation and dynamic collision avoidance [9]–[11]. The core idea is to use an external pedestrian trajectory prediction method and embed its output into the cost function or constraints.

Some authors also relate reaction-based methods like Social Force Model [12]–[15] and velocity obstacles [16] to the possible social navigation approaches. But in practice, those methods usually applied as supervisors for pre-training learning-based models or combined with optimization-based or learning-based approaches.

With the rising popularity of Deep Learning, learning-based social navigation, especially Reinforcement Learning

[1]The authors are with Skolkovo Institute of Science and Technology (Skoltech), Center for AI Technology. Corresponding e-mail: timur.akhtyamov@skoltech.ru

[2]The author is with Sber Robotics group

979-8-3503-0704-7/23/$31.00

(RL)-based methods have become their own direction in robotics [17]–[20]. For today, main directions in the RL-based social navigation research are modeling interaction between pedestrians and robot [18]–[20], efficient usage of the pedestrians motion prediction by RL policy [19]–[21] and combination of the RL-based methods with non-learnable approaches [10], [22], [23].

B. Uncertainty-aware objectives and constraints

The goal of incorporating into robot navigation in general is to minimize collision probability, directly or indirectly. A common approach is to model robot and pedestrian as circles (or spheres, if going to 3D), but calculating exact collision probability even for such simple representation is a challenging problem [24], [25].

One way of tackling this issue is the chance constraint which gives approximate bounds on collision up to fixed probability. Several groups of chance constraints are present in the literature. The first group is based on approximation of the collision probability or finding its upper bound [25]–[29]. The second group represents dynamic obstacles as circles or ellipses whose sizes derived via Gaussian level-sets of some fixed probability [8], [30], [31].

Another way of incorporating uncertainty into planning is using the concept of risk introduced in [32]. According to [32], risk is defined as a mapping of the cost random variable to a real number that should follow a set of axioms. Most popular risk metrics that can be found in the literature are Expected Cost [9], Conditional Value at Risk (CVaR) [33]–[35] and Mean-Variance [36].

Recent works also made steps towards uncertainty-awareness in RL via risk-aware RL [37]–[39] and Distributional RL [40], [41], but application of those method for social robot navigation problem is not well-studied problem yet.

C. Uncertainty-aware trajectory prediction

Uncertainty-aware trajectory prediction has been an active research area in robotics and autonomous navigation, particularly for applications involving social interactions. Traditional approaches to trajectory prediction rely on deterministic models [12], which may not account for the inherent uncertainty in the environment and the behavior of other agents. To address this issue, several recent works have proposed uncertainty-aware prediction models that explicitly model the uncertainty in the trajectory estimation [1], [42], [43]. In this work we use a variation of the CovarianceNet [1] model as an explicit method for uncertainty prediction in pedestrian trajectory estimation. In sake of simplicity and computational efficiency, our implementation is not using the Conditional Variational Autoencoder (CVAE) part of the original model. While the CVAE has been shown to produce diverse and realistic trajectories, we expect it would suffice to achieve the desired performance to omit the CVAE from our implementation of CovarianceNet. Also, as an underlying trajectory prediction method for CovarianceNet, the Constant Velocity (CV) model is used. Despite its extreme simplicity, in practice CV often produces results comparable to more sophisticated models in both prediction and navigation tasks [20], [44].

TABLE I: Variable Definition Table.

Variable	Definition
N	number of pedestrians
$i = \{1, \dots, N\}$	pedestrian index
H	number of receding horizon steps
$k = \{0, 1, \dots, H-1\}$	receding horizon step index
Δt	receding horizon time step interval, $[s]$
T^{sim}	number of simulation steps
Δt^{sim}	simulation time step interval, $[s]$
H^{ghost}	number of receding horizon steps to track ghost pedestrians
r^{rob}	robot circumference radius, $[m]$
V^S	volume of the sphere used for Mahalanobis constraints, $[m^3]$
r^{ped}	pedestrian circumference radius, $[m]$
d^{safe}	safe distance between robot and pedestrian circumferences, $[m]$
ε	target reach threshold, $[m]$
ℓ^{vis}	robot vision range, $[m]$
φ^{vis}	robot angle of view, $[rad]$
δ	adaptive margin constraint value
x	position along x axis, $[m]$
y	position along y axis, $[m]$
$\theta \in [-\pi; \pi)$	angular position, $[rad]$
v	linear velocity, $[\frac{m}{s}]$
ω	angular velocity, $[\frac{m}{s}]$
$\mathbf{r}_k = [x_k^{rob}, y_k^{rob}]^\top$	robot position vector at step k
$\mathbf{r}_{target}$	robot target position
$\mathbf{x}_k = [x_k^{rob}, y_k^{rob}, \theta_k^{rob}]^\top$	robot state vector at step k
$\mathbf{u}_k = [v_k^{rob}, \omega_k^{rob}]^\top$	robot control vector at step k
$\bar{\mathbf{u}}_k = [v_k^{rob}, \omega_k^{rob}, \delta_k]^\top$	augmented robot control vector at step k
$\mathbf{p}_{k,i} = [x_{k,i}^{ped}, y_{k,i}^{ped}]^\top$	i-th pedestrian position vector at step k
$\Sigma_{k,i}$	covariance of i-th pedestrian at k step
$\lambda_{k,i}^{(1)}, \lambda_{k,i}^{(2)}$	eigenvalues of the $\Sigma_{k,i}$
γ	number of standard deviations
$a_{k,i}, b_{k,i}$	length of the ellipsoid constraint semi-axes
$\psi_{k,i}$	rotation angle of the ellipsoid constraint
$\mathrm{Rot}(\psi)$	rotation matrix
P^{col}	collision probability threshold
$Q_{\mathbf{u}}$	control input weight matrix
$Q_{\bar{\mathbf{u}}}$	augmented control input weight matrix
$Q_{\mathbf{r}}$	position unattainability factor
$Q_{\mathbf{ED}}$	Euclidean distance cost weight
$Q_{\mathbf{MD}}$	Mahalanobis distance cost weight
$d_{k,i}^{\mathbf{ED}}$	Euclidean distance for i-th pedestrian at k step
$d_{k,i}^{\mathbf{MD}}$	Mahalanobis distance for i-th pedestrian at k step
$\mathbb{W}$	position space
$\mathbb{X}$	state space
$\mathbb{U}$	action space
$\bar{\mathbb{U}}$	augmented action space

III. Method

In this section, we present a comprehensive explanation of our proposed approaches that utilize MPC. For ease of reference, Table I is provided to define the main variables used throughout this section.

A. Robot Model

We first introduce the target robot model and system dynamics. The deterministic Markov decision process serves as a critical constraint that governs the behavior of the

system. For this study, we selected the kinematic unicycle model of the robot, which can be represented as a discrete system:

$$\begin{aligned} x_{k+1}^{rob} &= x_k^{rob} + v_k^{rob} \cdot cos(\theta_k^{rob}) \cdot \Delta t, \\ y_{k+1}^{rob} &= y_k^{rob} + v_k^{rob} \cdot sin(\theta_k^{rob}) \cdot \Delta t, \\ \theta_{k+1}^{rob} &= \theta_k^{rob} + w_k^{rob} \cdot \Delta t. \end{aligned} \tag{1}$$

It should be noted that in all proposed methods, the robot (ego-agent) and the pedestrians (agents) are modeled as circles with respective radii (r^{rob} and r^{ped}). The control input vector consists of linear and angular velocities, denoted as $\mathbf{u}_k = [v_k^{rob}, \omega_k^{rob}]^\top$.

B. Model Predictive Control

This section outlines the methods we have studied, all of which are based on MPC. MPC is an advanced control strategy that predicts the future behavior of a system using a mathematical model and a cost function as an optimization objective that encapsulates the target behavior of the agent. The cost function in our proposed methods consists of two parts: the stage cost (4) and the terminal cost (5). The stage cost is accumulated at each stage of the prediction horizon up to the terminal step and includes the control input cost (2) and the normalized target distance cost (3), which was inspired by the cost function presented in [10].

The control input cost 2 penalizes the usage of the control signal, which consists of the linear and angular velocities:

$$J_k^{\mathbf{u}}(\mathbf{u}_k) = \mathbf{u}_k^\top Q_{\mathbf{u}} \mathbf{u}_k. \tag{2}$$

The normalized target distance cost (3) penalizes the robot's deviation from the target position during the optimization process. This cost decreases as the robot gets closer to the target position at each iteration relative to its initial position at the beginning of the horizon:

$$J_k^{\mathbf{r}}(\mathbf{r}_k) = Q_{\mathbf{r}} \left(\frac{\|\mathbf{r}_k - \mathbf{r}_{target}\|_2}{\|\mathbf{r}_0 - \mathbf{r}_{target}\|_2} \right)^2. \tag{3}$$

The combination of the control input cost and the normalized target point distance cost results in the definition of the basic stage cost (4):

$$J_k(\mathbf{u}_k, \mathbf{r}_k) = J_k^{\mathbf{u}}(\mathbf{u}_k) + J_k^{\mathbf{r}}(\mathbf{r}_k). \tag{4}$$

In the terminal step of the optimization problem, we only penalize the robot's inability to reach the target position (5):

$$J_H := J_k^{\mathbf{r}}(\mathbf{r}_k) \mid k = H. \tag{5}$$

Combining all of the previously mentioned terms results in a basic MPC optimization problem, which we refer to as *MPC* (6) in the Table II:

$$\begin{aligned} \min_{\mathbf{r}_{1:H}, \mathbf{u}_{0:H-1}} \quad & \sum_{k=0}^{H-1} J_k(\mathbf{u}_k, \mathbf{r}_k) + J_H(\mathbf{r}_H) \\ \text{subject to} \quad & \mathbf{r}_0 = \mathbf{r}(0) \\ & \mathbf{u}_k \in \mathbb{U} \\ & \mathbf{r}_k \in \mathbb{W}. \end{aligned} \tag{6}$$

Currently, we have defined a basic MPC optimization problem that is suitable for navigation tasks. However, it does not consider pedestrians in the environment. In the following section, we discuss how we can incorporate pedestrians into the optimization problem, both considering and not considering uncertainty.

1) Uncertainty-unaware: We introduce a classical uncertainty-unaware approach commonly used in motion planning to account for obstacles, other agents, and environmental borders - the Euclidean distance. The Euclidean distance (ED) is a measure of the straight-line length between two points in Euclidean space 7:

$$d_{k,i}^{\mathbf{ED}}(\mathbf{r}_k, \mathbf{p}_{k,i}) = \|\mathbf{r}_k - \mathbf{p}_{k,i}\|_2. \tag{7}$$

It is often used in optimization problems to prevent controllers from colliding with obstacles by imposing a constraint on the distance between the ego-agent and other agents [10]. However, in our approach, we also study the utilization of the Euclidean distance as a component 8 of the stage-cost function. This is usually referred to as penalty-based optimization:

$$J_k^{\mathbf{ED}}(\mathbf{r}_k, \mathbf{p}_{k,1:N}) = Q_{\mathbf{ED}} \sum_{i=0}^{N} \frac{1}{d_{k,i}^{\mathbf{ED}}(\mathbf{r}_k, \mathbf{p}_{k,i})^2}. \tag{8}$$

We refer to the optimization problem that includes Euclidean distance as an additional component of the stage-cost function as ED-MPC 9:

$$\begin{aligned} \min_{\mathbf{r}_{1:H}, \mathbf{u}_{0:H-1}} \quad & \sum_{k=0}^{H-1} \left(J_k(\mathbf{u}_k, \mathbf{r}_k) + J_k^{\mathbf{ED}}(\mathbf{r}_k, \mathbf{p}_{k,1:N}) \right) + \\ & + J_H(\mathbf{r}_H) \\ \text{subject to} \quad & \mathbf{r}_0 = \mathbf{r}(0) \\ & \mathbf{u}_k \in \mathbb{U} \\ & \mathbf{r}_k \in \mathbb{W}. \end{aligned} \tag{9}$$

In order to use Euclidean distance as a constraint, an inequality must be introduced 10 to ensure that the safe distance between the ego-agent and pedestrians is not violated:

$$d_{k,i}^{\mathbf{ED}}(\mathbf{r}_k, \mathbf{p}_{k,i})^2 \geq (r^{rob} + r^{ped} + d^{safe})^2. \tag{10}$$

An optimization problem that includes Euclidean distance as an inequality constraint is referred to as MPC-EDC (11):

$$\begin{aligned} \min_{\mathbf{r}_{1:H}, \mathbf{u}_{0:H-1}} \quad & \sum_{k=0}^{H-1} J_k(\mathbf{u}_k, \mathbf{r}_k) + J_H(\mathbf{r}_H) \\ \text{subject to} \quad & \mathbf{r}_0 = \mathbf{r}(0) \\ & d_{k,i}^{\mathbf{ED}}(\mathbf{r}_k, \mathbf{p}_{k,i})^2 \geq (r^{rob} + r^{ped} + d^{safe})^2 \\ & \mathbf{u}_k \in \mathbb{U} \\ & \mathbf{r}_k \in \mathbb{W}. \end{aligned} \tag{11}$$

2) Uncertainty-aware: For the uncertainty-awareness, we first introduce approaches based on the Mahalanobis distance, which measures the distance between a point, e.g. robot position $\mathbf{r}_k$, and a distribution, e.g. predicted pedestrian position modeled as a Gaussian distribution with mean $\mathbf{p}_{k,i}$ and covariance matrix $\Sigma_{k,i}$ (which may include off-diagonal elements):

$$d_{k,i}^{\mathbf{MD}}\left(\mathbf{r}_k, \mathbf{p}_{k,i}, \Sigma_{k,i}\right) = \\ = \sqrt{\left(\mathbf{r}_k - \mathbf{p}_{k,i}\right)^\top \Sigma_{k,i}^{-1} \left(\mathbf{r}_k - \mathbf{p}_{k,i}\right)}. \quad (12)$$

We propose to employ Mahalanobis distance as and alternative to the Euclidean distance that will allow MPC to capture uncertainty of the pedestrian trajectories prediction.

First, we propose to add the Mahalanobis distance as an additional component to the stage cost function. To do this, we compute a weighted sum of the inverse Mahalanobis distance to each pedestrian at each horizon step:

$$J_k^{\mathbf{MD}}(\mathbf{r}_k, \mathbf{p}_{k,1:N}) = Q_{\mathbf{MD}} \sum_{i=0}^{N} \frac{1}{d_{k,i}^{\mathbf{MD}}\left(\mathbf{r}_k, \mathbf{p}_{k,i}, \Sigma_{k,i}\right)^2}. \quad (13)$$

This allows us to take into account for the uncertainty associated with each pedestrian's trajectory and adjust the cost function accordingly. An MPC controller that incorporates the Mahalanobis distance as an additional component to the stage cost function is referred to as MD-MPC.

The Mahalanobis distance can also be added as an inequality constraint to the optimization problem. Work [25] derives approximation for the collision probability for the spherical robot, and corresponding constraint expression for holding collision probability lower than given threshold probability P^{col}. We adopt this approximation to out problem where pedestrian position is uncertain and introduce following constraint:

$$d_{k,i}^{\mathbf{MD}}\left(\mathbf{r}_k, \mathbf{p}_{k,i}, \Sigma_{k,i}\right)^2 \geq 2\ln\left(\sqrt{\det\left(2\pi\Sigma_{k,i}\right)}\frac{P^{col}}{V^S}\right), \quad (14)$$

where V^S is the volume of the sphere with radius $r^{rob} + r^{ped} + d^{safe}$, P^{col} is the fixed collision probability threshold.

Along with Mahalanobis distance-based constraints, we propose another type of chance constraints, based on the idea of Gaussian iso-contours, proposed in [30], [31]. Assuming that $\Sigma_{k,i}$ is the covariance of the i-th pedestrian's position at horizon step k (which may include off-diagonal correlation terms), the parameters of the ellipsoid corresponding to the γ standard deviations are derived. We calculate eigenvalues of the covariance matrix $\lambda_{k,i}^{(1)}$ and $\lambda_{k,i}^{(2)}$ which define ellipsoid semi-axes lengths and angle $\psi_{k,i}$ which define the rotation of the coordinate system related to the ellipsoid. Taking into account robot and pedestrian radii along with safe distance, length of the semi-axes of the bounding ellipsoid $a_{i,k}$ and $b_{i,k}$ are defined as:

$$\begin{bmatrix} a_{k,i} \\ b_{k,i} \end{bmatrix} = \gamma \begin{bmatrix} \sqrt{\lambda_{k,i}^{(1)}} \\ \sqrt{\lambda_{k,i}^{(2)}} \end{bmatrix} + r^{rob} + r^{ped} + d^{safe}. \quad (15)$$

Final equation for the ellipsoid constraints has form:

$$\left(\mathbf{r}_k - \mathbf{p}_{k,i}\right)^\top \mathrm{Rot}(\psi_{k,i})^\top \begin{bmatrix} \frac{1}{a_{k,i}^2} & 0 \\ 0 & \frac{1}{b_{k,i}^2} \end{bmatrix} \times \\ \times \mathrm{Rot}(\psi_{k,i}) \left(\mathbf{r}_k - \mathbf{p}_{k,i}\right) > 1, \quad (16)$$

where $\mathrm{Rot}(\psi_{k,i})$ defines the rotation matrix. This constraint holds that robot will not move inside the ellipsoid around pedestrian, and size of this ellipsoid is based on selected number of Gaussian standard deviations, and thus connected with collision probability.

We refer to a variant of the MPC that uses ellipsoid constraints as MPC-ELC. Stage cost and terminal cost are defined by equations 4 and 5 correspondingly.

3) Adaptive constraint: We present an additional approach, called the *adaptive constraint*, initially introduced in [31] via slack variable. The adaptive constraint approach involves introducing a new optimization variable, denoted as δ, which is added to the augmented control input vector $\bar{\mathbf{u}}$. To properly formalize this approach, we introduce an augmented control input cost function, which replaces the original control input cost function (2) in the optimization problem (6):

$$J_k^{\bar{\mathbf{u}}}\left(\bar{\mathbf{u}}_k\right) = \bar{\mathbf{u}}_k^\top Q_{\bar{\mathbf{u}}} \bar{\mathbf{u}}_k. \quad (17)$$

Note that the initial robot model is not changed, and slack variable is added to the control vector for the ease of regularization.

The adaptive constraint can be in conjunction with the Euclidean distance constraint to increase the safe distance between the ego-agent and the pedestrian:

$$d_{k,i}^{\mathbf{ED}}\left(\mathbf{r}_k, \mathbf{p}_{k,i}\right)^2 \geq (r^{rob} + r^{ped} + d^{safe})^2 + \delta. \quad (18)$$

The controller that incorporates both the adaptive constraint and the Euclidean distance constraint is referred to as MPC-AEDC. This approach provides additional flexibility in adjusting the safe distance between the ego-agent and the pedestrians, making it particularly useful in dynamic and uncertain environments.

The adaptive constraint can also be used in conjunction with the Mahalanobis distance constraint to adjust the converted collision probability:

$$d_{k,i}^{\mathbf{MD}}\left(\mathbf{r}_k, \mathbf{p}_{k,i}\right)^2 \geq 2\ln\left(\sqrt{\det\left(2\pi\Sigma_{k,i}\right)}\frac{P^{col}}{V^S}\right) + \delta. \quad (19)$$

The controller that incorporates both the adaptive constraint and the Mahalanobis distance constraint is referred to as MPC-AMDC. This approach provides additional flexibility in adjusting the safety margin and collision probability.

We propose to apply adaptive constraint with ellipsoid constraints in a similar way to the original work [31]. We adjust the semi-axes of the bounding ellipsoid:

$$\begin{bmatrix} a_{k,i} \\ b_{k,i} \end{bmatrix} = \gamma \begin{bmatrix} \sqrt{\lambda_{k,i}^{(1)}} \\ \sqrt{\lambda_{k,i}^{(2)}} \end{bmatrix} (1-\delta) + r^{rob} + r^{ped} + d^{safe}. \quad (20)$$

We refer to such a controller as MPC-AELC.

Table II summarizes all the methods proposed in this paper and provides an overview of the design of each method.

TABLE II: Summary of Methods.

Controller Name	Cost Component	Constraint Type
ED-MPC	Euclidean (8)	-
ED-MPC-EDC	Euclidean (8)	Euclidean (10)
ED-MPC-MDC	Euclidean (8)	Mahalanobis (14)
MD-MPC-MDC	Mahalanobis (13)	Mahalanobis (14)
MD-MPC-EDC	Mahalanobis (13)	Euclidean (10)
ED-MPC-AEDC	Euclidean (8)	Adaptive Euclidean (18)
MD-MPC-AEDC	Mahalanobis (13)	Adaptive Euclidean (18)
MPC-AEDC	-	Adaptive Euclidean (18)
MPC-AMDC	-	Adaptive Mahalanobis (19)
MPC-ELC-2	-	Ellipsoid, $\gamma = 2$ (15, 16)
MPC-ELC-3	-	Ellipsoid, $\gamma = 3$ (15, 16)
MPC-AELC-2	-	Adaptive Ellipsoid, $\gamma = 2$ (20, 16)
MPC-AELC-3	-	Adaptive Ellipsoid, $\gamma = 3$ (20, 16)

IV. Evaluation

In this section, we provide a detailed description of our experimental setup, present and discuss the results of our experiments.

A. Software and Datasets

We utilized the *do-mpc* framework [45] for implementation[1] of all of the MPC-based controllers, which is built upon the *CasADi* software package [46] for nonlinear optimization and algorithmic differentiation. *MUltifrontal Massively Parallel sparse direct Solver (MUMPS)* is used as a base solver for MPC problem. CovarianceNet implementation uses *PyTorch* framework. As a simulation tool we have developed an open-source lightweight and flexible framework called *PyMiniSim*[2] (Fig. 1a). Our implemented CovarianceNet model[3] was trained on the subset of the Stanford Drone Dataset (SDD) [47].

B. Experimental Setup

We utilized the Headed Social Force Model (HSFM) [14], an extension of a highly-regarded Social Force Model (SFM) [48], as a model for simulation of pedestrians behavior.

To evaluate the effectiveness of the proposed methods, we designed and simulated three types of scenarios- *circular crossing*, *random crossing*, and *parallel crossing*, as illustrated in Figure 1b. These scenarios were inspired by the work [49]. For each scenario, we consider a set of scenes - configurations of number of pedestrians, initial pedestrians' poses, pedestrians' goal, initial robot pose and robot goal. Once pedestrians reach their goal positions, they oscillate between their initial and goal positions, resulting in continuous movement without stopping within the scene. Possible number of pedestrians varies from 3 to 8. For each number of the pedestrians, 100 scenes were generated, resulting in 600 scenes per scenario and 1800 scenes in total. Each of the controllers were evaluated on this set of scenes.

For evaluating the performance of the controllers, we utilized standard metrics such as *Simulation steps to Target,* $[\#]$, which represents the time taken for the controller to reach the target position, *Number of Collisions,* $[\#]$, and *Number of Timeouts,* $[\#]$, which depict the cautiousness of the controller. Target is assumed to be reached by the robot if the following criterion holds:

$$\|\mathbf{r}_k - \mathbf{r}_{target}\|_2 - r^{rob} < \varepsilon. \tag{21}$$

Here are the parameters that we used for the evaluation of the controllers:

$$\begin{array}{ll} N \in \{3,4,5,6,7,8\} & r^{ped} = 0.3 \\ H = 25 & d^{safe} = 0.3 \\ \Delta t = 0.1 & P^{col} = 0.01 \\ T^{sim} = 2000 & \varepsilon = 0.1 \\ \Delta t^{sim} = 0.01 & \ell^{vis} = 5 \\ H^{ghost} = 20 & \varphi^{vis} = 2\pi \\ Q_{\bar{\mathbf{u}}} = \begin{pmatrix} 0.005 & 0 & 0 \\ 0 & 0.005 & 0 \\ 0 & 0 & 100000 \end{pmatrix} & Q_{\mathbf{u}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ r^{rob} = 0.35 & Q_{\mathbf{ED}} = 500 \\ & Q_{\mathbf{MD}} = 1000, \end{array}$$

$Q_{\mathbf{r}} = 100$ if an additional cost component is Euclidean, otherwise $Q_{\mathbf{r}} = 1000$. We make the assumption that the robot is imperceptible to pedestrians, and therefore, they do not respond to its presence. The unicycle kinematics model described in Section III is used to model the robot in both the controller optimization problem and the simulation model. However, there is a difference in the time intervals used- $\Delta t = 0.1$ in the optimization problem and $\Delta t^{sim} = 0.01$ in the simulation model. The controller is invoked every $\Delta t = 0.1$ time interval within the simulation model, which is known as a 'sample and hold' system. To enhance the navigation capabilities, we implemented the *ghost-pedestrian* feature in PyMiniSim. This feature enables the robot to continue tracking the pedestrian using his last trajectory prediction when the pedestrian leaves the robot's field of view, up to H^{ghost} steps.

C. Results and Analysis

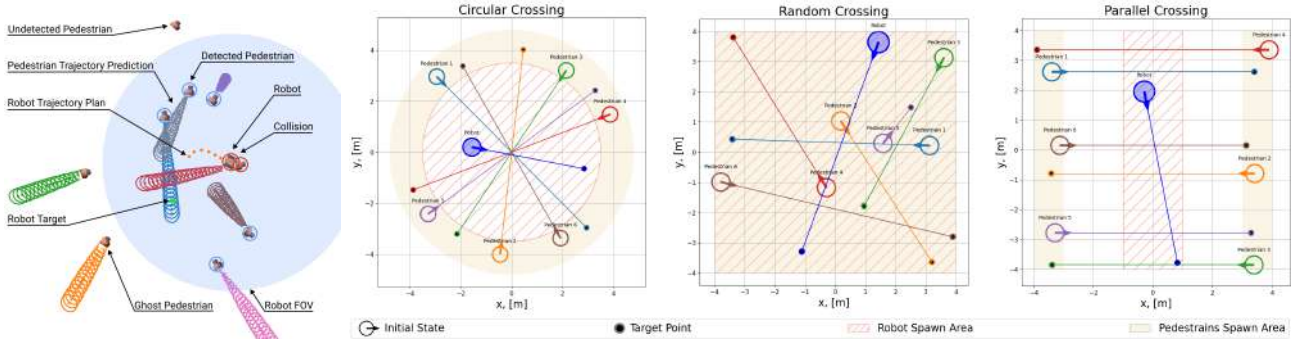
Results of the experiments are represented by the statistics, collected for each of the three proposed types of the scenarios, showed at Fig. 2, 3 and 4 with means, medians and interquartile ranges (IQR). Using this data, we provide both scenario-specific analysis and derive general conclusions on the practical applications of the proposed controllers.

According to our observations, the *circular crossing* scenario (Fig. 2) is the most challenging scenario in practice, since when the goal is sampled inside the inner circle, robot needs to reach it as fast as possible until it becomes cramped by the pedestrians; when the goal is sampled outside the inner circle, robot needs to carefully break out of it. In terms of the number of collisions, for the lowest number of pedestrians all methods perform similarly. For the larger numbers, we can observe degradation of several uncertainty-aware methods and methods that do not employ adaptive constraints. Generally, for this scenario good performance in terms of collisions is achieved by MD-MPC-AEDC, MPC-AEDC, ED-MPC-AEDC, MPC-ELC-3 and MPC-AELC-3. In terms of numbers of simulation steps and timeouts, we see

[1] https-//github.com/TimeEscaper/social_nav_baselines

[2] https-//github.com/TimeEscaper/pyminisim

[3] https-//github.com/alexpostnikov/CovarianceNet



(a) PyMiniSim environment.

(b) Scenario generation.

Fig. 1: Experimental setup. In this study we used simulation environment based on PyMiniSim (Fig. 1a) framework. We used scenarios represented in (Fig. 1b).

that ellipsoid constraints based method with largest number of standard deviations (MPC-ELC-3) tend to be much more conservative, and usage of the adaptive constraints (MPC-AELC-3) partially tackles this issue. With smaller number of standard deviations (MPC-ELC-2 and MPC-AELC-3), ellipsoid constraints based methods show level of conservative much closer to the other methods mentioned above. For the hardest case in this scenario, we provide detailed results in Table III. For this case, we highlight performance of MD-MPC-AEDC and MPC-AEDC approaches.

The *random crossing* scenario (Fig. 3) tends to be a 'medium-complexity' problem for the controllers. We again see the trend of degrading performance of the Mahalanobis and Euclidean non-adaptively constrained controllers. Comparing the hardest cases of 7-8 pedestrians, we can see that good performance is shown by MPC-ELC-2, MPC-ELC-3, MPC-AELC-2, MPC-AELC-3, MD-MPC-AEDC, ED-MPC-AEDC. In terms of simulation steps and timeouts, a gap between MPC-AELC-2 and both MPC-AELC-3 and ED-MPC-AEDC can be seen, same for the MPC-ELC-2 and MPC-ELC-3.

While visually looking like a relatively simple problem, the *parallel crossing* scenario (Fig. 4) still tends to be a challenging problem for the controllers, especially when the robot becomes close to the two pedestrians approaching each other. The trend of degrading performance of the Mahalanobis and Euclidean non-adaptively constrained controllers can be seen again. Comparing the hardest cases of 6-8 pedestrians, we can see that good performance is shown by MPC-AEDC, MD-MPC-AEDC, MPC-AELC-2, MPC-AELC-3 and ED-MPC-AEDC. Performance in terms of simulation steps and timeouts is similar to the previous case, still we see a huge gap between MPC-AELC-2 and MPC-AELC-3 which gives insight on the influence of the number of standard deviations on the agility.

Based on our findings for each of the scenarios, we can propose following conclusions-

1) *Adaptive constraints are the crucial part for MPC-based methods*. We see that leading controllers employ the concept of adaptive constraints. We also observed that adaptive constraints in some cases make controllers more stable.
2) *Designing uncertainty-aware MPC components is still a hard task*. Poor performance of the methods that use non-adaptive Mahalanobis distance based constraints tells that approximation introduced in 14 is too coarse approximation, and, as was discussed in II-B, introducing more precise approximations can be a tricky task. On the other hand, chance constraints require tuning to find trade-off between safety and agility. Still, both chance constraints and methods employing Mahalanobis distance based cost showed their potential in making the system safer.

TABLE III: Circular Crossing Scenario Results, $N = 8$.

Controller Name	Simulation Steps to Target Q1 | Median | Mean | Q2	Number of Collisions	Number of Timeouts
ED-MPC	312.5 | 703.0 | 671.6 | 934.0	1.0 | 2.0 | 1.62 | 2.0	0.0 | 0.0 | 0.01 | 0.0
ED-MPC-EDC	593.0 | 780.0 | 940.9 | 1275.0	0.0 | 1.0 | 1.33 | 2.0	0.0 | 0.0 | 0.35 | 1.0
ED-MPC-MDC	612.3 | 802.0 | 819.6 | 1002.8	0.0 | 2.0 | 2.71 | 4.0	0.0 | 0.0 | 0.04 | 0.0
MD-MPC-MDC	318.0 | 483.0 | 575.8 | 791.0	1.0 | 2.0 | 3.11 | 5.0	0.0 | 0.0 | 0.00 | 0.0
MD-MPC-EDC	549.0 | 730.5 | 865.5 | 1189.8	0.0 | 1.0 | 1.23 | 2.0	0.0 | 0.0 | 0.28 | 1.0
ED-MPC-AEDC	384.0 | 780.0 | 760.9 | 1022.0	0.0 | 1.0 | 0.96 | 2.0	0.0 | 0.0 | 0.03 | 0.0
MD-MPC-AEDC	**315.3 | 411.5 | 496.3 | 562.8**	**0.0 | 0.0 | 0.83 | 2.0**	**0.0 | 0.0 | 0.00 | 0.0**
MPC-AEDC	**296.0 | 406.0 | 483.0 | 538.0**	**0.0 | 0.0 | 0.87 | 2.0**	**0.0 | 0.0 | 0.00 | 0.0**
MPC-AMDC	260.3 | 329.0 | 380.5 | 430.8	1.0 | 2.0 | 2.21 | 3.0	0.0 | 0.0 | 0.00 | 0.0
MPC-ELC-2	417.0 | 582.0 | 741.9 | 989.0	0.0 | 1.0 | 1.30 | 2.0	0.0 | 0.0 | 0.05 | 0.0
MPC-ELC-3	461.0 | 560.0 | 696.4 | 703.0	0.0 | 1.0 | 1.18 | 2.0	0.8 | 1.0 | 0.75 | 1.0
MPC-AELC-2	422.5 | 549.0 | 713.2 | 989.0	0.0 | 1.0 | 1.52 | 2.0	0.0 | 0.0 | 0.09 | 0.0
MPC-AELC-3	403.3 | 576.5 | 699.0 | 824.0	0.0 | 0.0 | 1.17 | 2.0	0.0 | 1.0 | 0.56 | 1.0

V. Conclusion

In this work, we studied several approaches for designing socially-aware MPC in both uncertainty-unaware and uncertainty-aware settings. We provided their comprehensive evaluation which includes the development of the simulation environment, design of the social scenarios, collection of the statistics on controllers' performance and analysis of those results. We derive several conclusions which may help developers and researchers to decide when embedding uncertainty-awareness may work efficiently and when simpler uncertainty-unaware controllers may provide the same or even better performance. Further directions for our study is to explore uncertainty-unaware and uncertainty-aware MPC implementations with numerical and sampling-based solvers and conducting experiments on the real robotic platform.

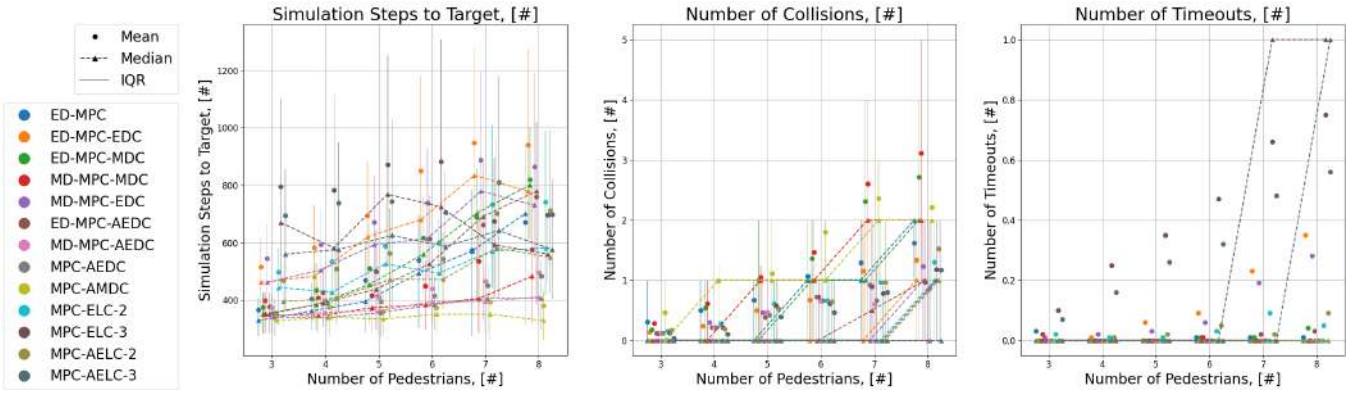


Fig. 2: Statistical results for the Circular Crossing Scenario.

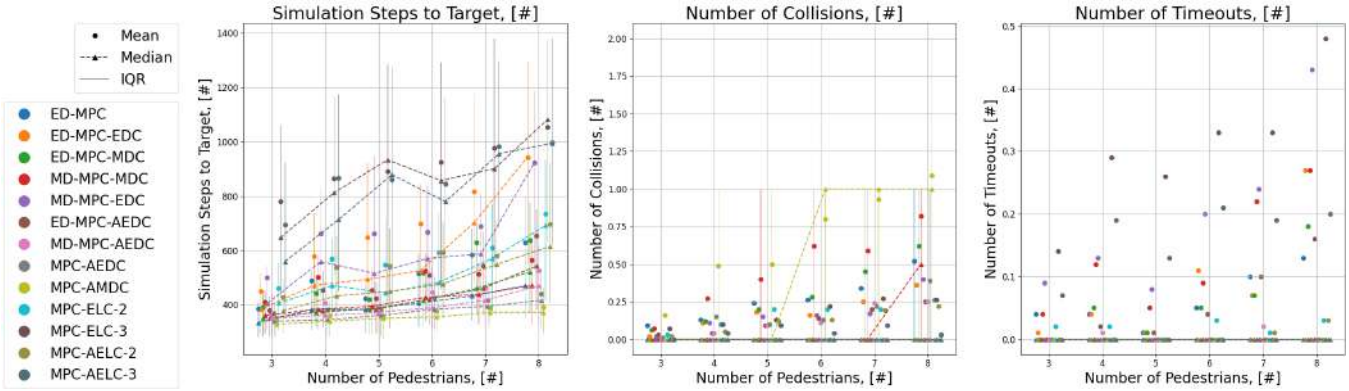


Fig. 3: Statistical results for the Random Crossing Scenario.

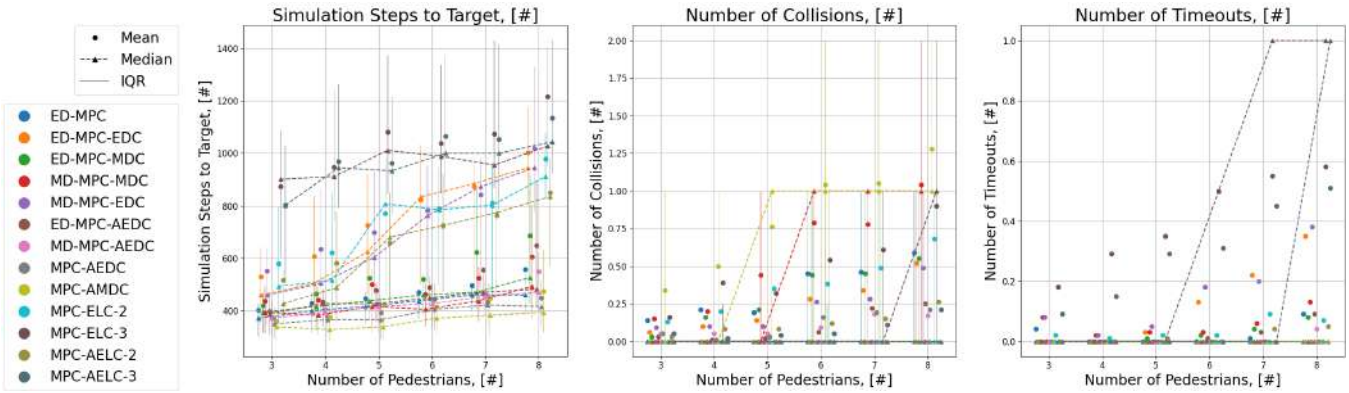


Fig. 4: Statistical results for the Parallel Crossing Scenario.

REFERENCES

[1] A. Postnikov, A. Gamayunov, and G. Ferrer, "CovarianceNet: Conditional generative model for correct covariance prediction in human motion prediction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[2] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[3] K. Charalampous, I. Kostavelis, and A. Gasteratos, "Recent trends in social aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 93, pp. 85–104, 2017.

[4] C. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld, and J. Oh, "Core challenges of social robot navigation: A survey," *arXiv preprint arXiv:2103.05668*, 2021.

[5] M. Kollmitz, K. Hsiao, J. Gaa, and W. Burgard, "Time dependent planning on a layered social cost map for human-aware robot navigation," in *2015 European Conference on Mobile Robots (ECMR)*. IEEE, 2015, pp. 1–6.

[6] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "Understanding human interaction for probabilistic autonomous navigation using risk-rrt approach," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 2014–2019.

[7] W. Chi and M. Q.-H. Meng, "Risk-rrt: A robot motion planning algorithm for the human robot coexisting environment," in *2017 18th International Conference on Advanced Robotics (ICAR)*. IEEE, 2017, pp. 583–588.

[8] K. Majd, S. Yaghoubi, T. Yamaguchi, B. Hoxha, D. Prokhorov, and G. Fainekos, "Safe navigation in human occupied environments using sampling and control barrier functions," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

IEEE, 2021, pp. 5794–5800.

[9] Y. Chen, F. Zhao, and Y. Lou, "Interactive model predictive control for robot navigation in dense crowds," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 4, pp. 2289–2301, 2021.

[10] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, "Where to go next: Learning a subgoal recommendation policy for navigation in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4616–4623, 2021.

[11] S. Poddar, C. Mavrogiannis, and S. S. Srinivasa, "From crowd motion prediction to robot navigation in crowds," *arXiv preprint arXiv:2303.01424*, 2023.

[12] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[13] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, "The walking behaviour of pedestrian social groups and its impact on crowd dynamics," *PloS one*, vol. 5, no. 4, p. e10047, 2010.

[14] F. Farina, D. Fontanelli, A. Garulli, A. Giannitrapani, and D. Prattichizzo, "Walking ahead: The headed social force model," *PloS one*, vol. 12, no. 1, p. e0169734, 2017.

[15] G. Ferrer and A. Sanfeliu, "Anticipative kinodynamic planning: Multi-objective robot navigation in urban and dynamic environments," *Autonomous Robots*, vol. 43, no. 6, pp. 1473–1488, 2019.

[16] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Optimal reciprocal collision avoidance for multi-agent navigation," in *Proc. of the IEEE International Conference on Robotics and Automation, Anchorage (AK), USA*, 2010.

[17] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[18] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6015–6022.

[19] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 007–10 013.

[20] S. Liu, P. Chang, Z. Huang, N. Chakraborty, W. Liang, J. Geng, and K. Driggs-Campbell, "Intention aware robot crowd navigation with attention-based interaction graph," *arXiv preprint arXiv:2203.01821*, 2022.

[21] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, "Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 11 345–11 352.

[22] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "Dwa-rl: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6057–6063.

[23] Z. Xie and P. Dames, "Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles," *arXiv preprint arXiv:2301.06512*, 2023.

[24] E. A. Cooper and H. Farid, "A toolbox for the radial and angular marginalization of bivariate normal distributions," *arXiv preprint arXiv:2005.09696*, 2020.

[25] N. E. Du Toit and J. W. Burdick, "Probabilistic collision checking with chance constraints," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 809–815, 2011.

[26] D. Althoff, M. Althoff, D. Wollherr, and M. Buss, "Probabilistic collision state checker for crowded environments," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 1492–1498.

[27] J. S. Park, C. Park, and D. Manocha, "Efficient probabilistic collision detection for non-convex shapes," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1944–1951.

[28] C. Park, J. S. Park, and D. Manocha, "Fast and bounded probabilistic collision detection for high-dof trajectory planning in dynamic environments," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 980–991, 2018.

[29] A. Thomas, F. Mastrogiovanni, and M. Baglietto, "Probabilistic collision constraint for motion planning in dynamic environments," in *Intelligent Autonomous Systems 16: Proceedings of the 16th International Conference IAS-16*. Springer, 2022, pp. 141–154.

[30] W. Schwarting, J. Alonso-Mora, L. Pauli, S. Karaman, and D. Rus, "Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1928–1935.

[31] F. L. Busch, J. Johnson, E. L. Zhu, and F. Borrelli, "A gaussian process model for opponent prediction in autonomous racing," *arXiv preprint arXiv:2204.12533*, 2022.

[32] A. Majumdar and M. Pavone, "How should a robot assess risk? towards an axiomatic theory of risk in robotics," in *Robotics Research: The 18th International Symposium ISRR*. Springer, 2020, pp. 75–84.

[33] R. S. Novin, A. Yazdani, A. Merryweather, and T. Hermans, "Risk-aware decision making for service robots to minimize risk of patient falls in hospitals," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3299–3305.

[34] X. Cai, M. Everett, L. Sharma, P. R. Osteen, and J. P. How, "Probabilistic traversability model for risk-aware motion planning in off-road environments," *arXiv preprint arXiv:2210.00153*, 2022.

[35] S. Triest, M. G. Castro, P. Maheshwari, M. Sivaprakasam, W. Wang, and S. Scherer, "Learning risk-aware costmaps via inverse reinforcement learning for off-road navigation," *arXiv preprint arXiv:2302.00134*, 2023.

[36] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[37] A. Tamar, Y. Chow, M. Ghavamzadeh, and S. Mannor, "Policy gradient for coherent risk measures," *Advances in neural information processing systems*, vol. 28, 2015.

[38] S. Zhang, B. Liu, and S. Whiteson, "Mean-variance policy iteration for risk-averse reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 905–10 913.

[39] S. Jaimungal, S. M. Pesenti, Y. S. Wang, and H. Tatsat, "Robust risk-aware reinforcement learning," *SIAM Journal on Financial Mathematics*, vol. 13, no. 1, pp. 213–226, 2022.

[40] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 449–458.

[41] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[42] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Gohome: Graph-oriented heatmap output for future motion estimation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 9107–9114.

[43] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer, 2020, pp. 683–700.

[44] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, "What the constant velocity model can teach us about pedestrian motion prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1696–1703, 2020.

[45] S. Lucia, A. Tatulea-Codrean, C. Schoppmeyer, and S. Engell, "Rapid development of modular and sustainable nonlinear model predictive control solutions," *Control Engineering Practice*, vol. 60, p. 51–62, 03 2017.

[46] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[47] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory prediction," in *European Conference on Computer Vision (ECCV)*, vol. 3, 2016.

[48] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[49] J. Wang, W. P. Chan, P. Carreno-Medrano, A. Cosgun, and E. Croft, "Metrics for evaluating social conformity of crowd navigation algorithms," in *2022 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*. IEEE, 2022, pp. 1–6.

Delta filter – robust visual-inertial pose estimation in real-time: A multi-trajectory filter on a spherical mobile mapping system*

Fabian Arzberger[1], Fabian Wiecha[1], Jasper Zevering[1], Julian Rothe[2], Dorit Borrmann[3], Sergio Montenegro[2], and Andreas Nüchter[1]

***Abstract*— Many state-of-the-art mobile mapping systems accomplish reliable and robust pose estimation utilizing combinations of inertial measurement units (IMUs), global navigation satellite systems (GNSS), visual-inertial- or LiDAR-inertial odometry (VIO/LIO). However, on a spherical mobile mapping system the underlying inherent rolling motion introduces high angular velocities, thus the quality of pose estimates, images, and laser-scans, degrade. In this work we propose a pose filter design that is able to do real-time sensor fusion between two unreliable trajectories into one, more reliable trajectory. It is a simple yet effective filter design that does not require the user to estimate the uncertainty of the sensors. The approach is not limited to spherical robots and theoretically is also suitable for sensor fusion of an arbitrary number of estimators. This work compares this filter against two pose estimation methods on our spherical system: (1) An approach that is based solely on IMU measurements, and (2) stereo-VIO with an Intel® RealSense™ tracking camera. The proposed "Delta" filter takes as input (1), (2), and a motion model. Our implementation gets rid of the drift in (1) and (2), estimates the scale of the trajectory, and deals with slow and fast motion as well as driving curves. Our source code can be found on github [1].**

I. Introduction

Spherical mobile mapping systems are just comming of age, as current research in the robotics community shows: The majority of research dealing with spherical systems is about locomotion mechanisms, e.g. [2]–[7]. Using spherical robots for mobile mapping (see Figure 1) is a rather novel field. To the best of our knowledge, Borrmann et al. [8] first used a 2D laser-scanner mounted on a unicycle's wheel axis, to generate maps via offline- simultaneous localization and mapping (SLAM). In a follow-up study from our own lab [9] we used the same laser-scanner inside a spherical robot with a protective outer plastic shell. The robot is capable of self-initiated motion via flywheels utilizing an IBCOAM (impulse by conservation of angular momentum) approach.

*We acknowledge funding from the ESA Contract No. 4000130925/20/NL/GLC for the study "DAEDALUS – Descent And Exploration in Deep Autonomy of Lava Underground Structures" within the Open Space Innovation Platform (OSIP) lunar caves-system and the Elite Network Bavaria (ENB) for providing funds for the academic program "Satellite Technology"

[1]The authors are with Computer Science XVII – Robotics, Julius-Maximilians-Universität Würzburg, 97074 Am Hubland, Germany. Contact: fabian.arzberger@uni-wuerzburg.de

[2]The authors are with Computer Science VIII – Aerospace Information Technology, Julius-Maximilians-Universität Würzburg, 97074 Am Hubland, Germany. Contact: julian.rothe@uni-wuerzburg.de

[3]Dorit Borrmann is with THWS Robotics, Technische-Hochschule-Würzburg-Schweinfurt, 97421 Schweinfurt, Germany. Contact: dorit.borrmann@thws.de

979-8-3503-0704-723$31.00 ©2023 IEEE

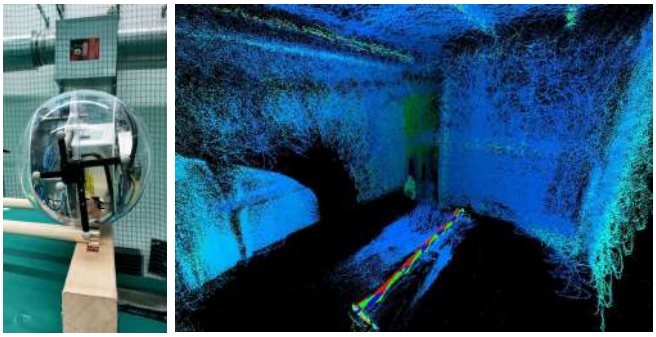


Fig. 1: (Left:) Labotary setup with the spherical mobile mapping system. (Right:) Resulting point cloud when applying the estimated trajectory of the proposed Delta-filter to LiDAR data. A high-resolution video where the filter runs onboard in real-time is available at https://youtu.be/2yu1RHtTesc.

The idea of using spherical robots for mapping was explored in more depth by the European Space Agency (ESA) in 2021 during a concurrent design facility (CDF) study. This CDF study considers the general concept of a spherical robot for environment mapping and exploring lunar caves, but also terrestrial vents, to be feasible [10], [11]. Advantages of using spherical robots are a shell that protects internal sensors and a versatile locomotion mechanism that inherently results in sensor rotation leading to optimal coverage of the environment. During SLAM, large and aggressive rotations are the least favorable motions that a mobile mapping system could experience. This is because for any falsely estimated translation, the errors in the resulting environment grow linearly, whereas for rotation these errors grow exponentially with increasing distance. While working with spherical robots, non-centered rotation is the main movement of the internal sensors, which proposes a huge challenge to state of the art SLAM algorithms. In previous work, we proposed initial offline-SLAM solutions for simplified sub-problems, i.e., rotation while descending [12], and rolling on flat surfaces [13]. However, in this work we address only the localization of the system and do not perform offline-SLAM, by introducing a pose estimation filter. Our implementation fuses information from three IMUs and a stereo-tracking camera onboard in real-time. The contributions of this work are as follows:

- A robust yet simple 6-DoF multi-trajectory filter, designed for but not limited to visual-inertial sensor

fusion. The implementation on our spherical robot requires only one hyperparameter: the radius of the sphere.
- An evaluation of our spherical mobile mapping systems accuracy based not only on ground truth point-clouds, but also on ground truth trajectories, which is stated as an open problem in [13].

The paper is structured as follows: In the next section, we provide an overview of state of the art 6-DoF pose filters, and outline the most similar approaches. Then, we introduce the "Delta"-filter in a general fashion and show an example implementation on a spherical mobile mapping system. Finally, we introduce our accuracy measures and experiments and show that the filter is able to deal with slow and fast motion as well as driving curves.

II. State-of-the-art

Many onboard multi-sensor pose estimation approaches exist in the community. The majority of which being implemented and developed towards autonomous driving cars [14], [15], and unmanned aerial vehicles (UAV) [16], [17]. Soloviev et al. [18] give a broad outline on the sensor types used for navigation: They define a self-contained inertial navigation system (INS) as the primary sensor, as it is available on any platform. Further, the authors consider the following secondary sensors which are qualified for fusion with the INS solution: Global Navigation Satellite System (GNSS) based (e.g. GPS), feature based (e.g. cameras or LiDAR), beacon based (e.g. using specialized navigation signals), or based on signals of opportunity (SoOP) (e.g. radio-frequency signals). In this work we will focus on visual-inertial navigation systems (VINS) and later propose a filter for our spherical system. Santoso et al. [19] categorize popular filters in the robotics community: (1) The Kalman Filter (KF) [20] has been designed to estimate the most likely system state under Gaussian noise by minimizing the covariances of the estimation error. It has since been reinvented and extended serval times, leading to variants such as the Unscented Kalman Filter (UKF) [21], Extended Kalman Filter (EKF) [22], or Multistate Constrained Kalman Filter (MSCKF) [23], just to name a few. KF-based approaches are by far the most popular state estimators among the robotics community. Example implementations on different systems include [17], [24]–[28]. (2) The H_∞ filter approach originates from control theory where it is used as an optimal robust controller. Instead of minimizing the covariance of the estimation error, the H_∞ filter minimizes the worst-case estimation error, which leads to better performance if modelling uncertanties are present [29]. (3) Particle filters (PF), or Monte-Carlo Methods, are known for being applied in many stochastic estimation problems [30]. By now, it is well-known that PF outperforms KF in nonlinear systems underlying non-Gaussian noise [31]. Its biggest drawback is the computational load required for processing many particles representing a single state. (4) Rao-Blackwellized Particle filters (RBPF) combine the advantages of PF and KF while getting rid of their major issues [32]. Therefore, if the system state model contains linear parts with Gaussian noise, these components are seperated and processed using KFs, while nonlinear parts with non-Gaussian noise are dealt with PFs. And finally, in recent years we have noticed the use of (5) graph optimization based methods such as GOMSF [33] and VIRAL-Fusion [34], where the system states are represented and optimized in a pose-graph.

The above mentioned examples solely treat filters implemented on ground vehicles or UAV. Yet other examples exist that implement multi-sensor pose estimation on more challenging systems. Kim et al. [35] fuse data from four sensing modalities on an unmanned underwater vehicle (UUV) using an approach using covariance intersection based on nonlinear optimization. They consider measurements taken via acoustic ultra-short baseline (USBL), Differential GPS (DGPS), Doppler Velocity Logs (DVL), and an INS. Fang et al. [36] use three different sensors for pose estimation on wearable augmented reality (WAR): a monocular camera, a depth sensor, and an INS. They use a KF-based approach in a sliding window fashion. To our knowledge there exists only one onboard pose estimation filter for spherical robots [37]. This approach [37] comes from our own lab and uses only data from inertial measurement units (IMU). The basic idea is to combine the well known IMU orientation filters: the Madgwick filter [38] and Complementary filter [39]. As for translation, the filter performs dead-reckoning using the motion model of a rolling sphere and adding constraints for slipping and sliding effects. Furthermore, the output of the filter in [37] is being utilized as input for the filter proposed in this paper. Lastly, we want to mention another filter that is much simpler than any of the approaches stated above, yet surprisingly effective: Gyrodometry [40]. This filter has been implemented to combine data from wheel encoders (Odometry) with data from a gyroscope by considering not the measured state, but instead the change of state. Therefore, the filter considers the similarity of the measurements to each other to eliminate outliers and update the current state accordingly. The proposed Delta-filter in this paper is similar in these two aspects (change of state and similarity of measurements), but extends the idea to an arbitrary number of estimators in 6-DoF and adds a motion model.

III. Sensor fusion with 6-DoF Delta-filter

In this section we propose a new pose filter design: the "Delta" filter. Its purpose is to receive 6-DoF trajectory estimates from multiple sources, which are known to be unreliable, and filter them in a probabilistic way. We consider a trajectory "unreliable" if it accumulates drift or makes sudden jumps - which are common effects in IMU- and VIO-based estimators. The filtered trajectory does not use any information from future measurements and is computed in real-time. However, similar to a Kalman filter, the Delta-filter requires a motion model, which is also considered unreliable. In our implementation we filter only two trajectory estimates with a given motion model, yet the Delta-filter is theoretically suitable for an arbitrary number of estimators.

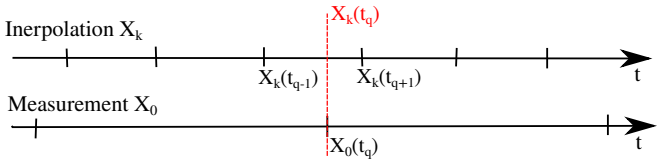


Fig. 2: Timelines showing two sensors publishing pose data at different rates. The sensor having the slower rate is defined as the "measurement", the other trajectories $\boldsymbol{X}_k$ get interpolated at measurement time t_q.

A. Proposed filter design

Suppose we have multiple 6-DoF pose estimators $\boldsymbol{X} = [\boldsymbol{R}, \boldsymbol{p}]^\tau \in \mathrm{SE}(3)$, where $\boldsymbol{R}$ is a 3×3 rotation matrix and $\boldsymbol{p}$ is a vector in $\mathbb{R}^3$. The pose of the k-th estimator at time t is denoted by $\boldsymbol{X}_k(t) = [\boldsymbol{R}_k(t), \boldsymbol{p}_k(t)]^\tau : \mathbb{R} \to \mathrm{SE}(3)$. Note that all poses from all estimators must first be transfered in a shared global coordinate frame. As the poses arrive at different time stamps, it is necessary to interpolate between measurements to capture all estimates at the same point in time. Thus, the Delta-filter computes an estimate at the rate of the slowest estimator, denoted as $\boldsymbol{X}_0$, yielding a query time t_q. We call the resulting pose $\boldsymbol{X}_0(t_q)$ the "measurement". All other estimators $\boldsymbol{X}_k$ are queried at time t_q, by interpolating between two measurements at given timestamps $t_{q\pm1}$, as shown in Figure 2. Note that rotation matrices and unit quaternions are isomorphic, thus we use $\boldsymbol{q}_k(t)$ and $\boldsymbol{R}_k(t)$ interchangeably as they represent the same elements in $\mathrm{SO}(3)$. Then, the interpolation is constructed using quaternion slerp and linear vector interpolation as described by Equations (1) - (5):

$$\boldsymbol{X}_k(t_q) = [\boldsymbol{R}_k(t_q), \boldsymbol{p}_k(t_q)]^\tau \ , \tag{1}$$

$$\hat{t} = \frac{t_q - t_{q+1}}{t_{q-1} - t_{q+1}} \in [0;1] \ , \tag{2}$$

$$\Omega = \cos^{-1}\left(\boldsymbol{q}_k(t_{q-1}) \cdot \boldsymbol{q}_k(t_{q+1})\right) \ , \tag{3}$$

$$\begin{aligned}\boldsymbol{R}_k(t_q) &= \mathrm{Slerp}\left(\boldsymbol{q}_k(t_{q-1}), \boldsymbol{q}_k(t_{q+1}), \hat{t}\right) \\ &= \frac{\sin((1-\hat{t})\Omega)}{\sin(\Omega)} \cdot \boldsymbol{q}_k(t_{q-1}) + \frac{\sin(\hat{t}\Omega)}{\sin(\Omega)} \cdot \boldsymbol{q}_k(t_{q+1}),\end{aligned} \tag{4}$$

$$\boldsymbol{p}_k(t_q) = (1-\hat{t}) \cdot \boldsymbol{p}_k(t_{q-1}) + \hat{t} \cdot \boldsymbol{p}_k(t_{q+1}) \tag{5}$$

The idea of the Delta-filter is to track the changes between given timestamps t_1 and t_2 (also known as "deltas") of the measurements and interpolations

$$\Delta\boldsymbol{X} = \left[\boldsymbol{R}^{-1}(t_2) \cdot \boldsymbol{R}(t_1) \ , \ \boldsymbol{p}(t_2) - \boldsymbol{p}(t_1)\right]^\tau \tag{6}$$

and estimate a new delta that is more meaningful. That is to say that the Delta-filter estimates the most likely pose change between given timestamps. Therefore, the filter first estimates a model delta

$$\begin{aligned}\Delta\boldsymbol{X}_m &= [\Delta\boldsymbol{R}_m, \Delta\boldsymbol{p}_m]^\tau \\ &= f\left(\Delta\boldsymbol{X}_0 \ , \ \{\Delta\boldsymbol{X}_k : k \in \mathbb{N}\}\right)\end{aligned} \tag{7}$$

where f denotes the motion model that estimates the true motion given the measured and interpolated deltas, $\Delta\boldsymbol{X}_0$ and $\Delta\boldsymbol{X}_k$. In a later section we will give an example for the motion model f when implementing the filter on a spherical robot.

1) Measurement, interpolation, and model: The measurement, interpolation, and model deltas $\Delta\boldsymbol{X}_0$, $\Delta\boldsymbol{X}_k$, and $\Delta\boldsymbol{X}_m$ respectiveley, are all considered unreliable. They are used to estimate the filtered pose $\boldsymbol{X}_e(t_j)$ by iterativeley applying an estimated filtered delta $\Delta\boldsymbol{X}_e$ that happened between t_{j-1} and t_j:

$$\begin{aligned}\boldsymbol{X}_e(t_j) &= \Delta\boldsymbol{X}_e \cdot \boldsymbol{X}_e(t_{j-1}) \\ &= [\Delta\boldsymbol{R}_e \cdot \boldsymbol{R}_e(t_{j-1}), \Delta\boldsymbol{p}_e + \boldsymbol{p}_e(t_{j-1})]^\tau \ .\end{aligned} \tag{8}$$

We separate the rotation and translation parts by assuming that the measured and interpolated orientations are sufficiently reliable estimates, i.e., they do not drift or jump during a short time period. This assumption is valid for most inertial- and visual-tracking systems. To obtain the estimated filtered rotation delta $\Delta\boldsymbol{R}_e$, we compute

$$\Delta\boldsymbol{R}_e = \mathrm{Slerp}\left(\Delta\boldsymbol{q}_0, \Delta\boldsymbol{q}_k, \frac{1}{2}\right) \tag{9}$$

Note that for more than two estimators, the Slerp in Equation (9) must be replaced with a different quaternion average, e.g. [41]. Furthermore, we assume that the estimated translation deltas $\Delta\boldsymbol{p}_0$, $\Delta\boldsymbol{p}_k$, and $\Delta\boldsymbol{p}_m$ are not sufficiently reliable to just average them, as inertial-tracking tends to drift and visual-tracking tends to jump.

2) Probabilistic weighted geometric mean: Therefore, we use a probabilistic approach that averages the translation direction and then scales it.

$$\Delta\boldsymbol{p}_e = \frac{d}{|\sum_i \Delta\boldsymbol{p}_i|} \cdot \sum_i \Delta\boldsymbol{p}_i \tag{10}$$

where $\Delta\boldsymbol{p}_i$ refers to the measurement, interpolation, and model deltas. An estimate of the true scale of the translated distance d is given by a probabilistically weighted geometric mean:

$$d = \left(\prod_{\omega_i} |\Delta\boldsymbol{p}_i|^{\omega_i}\right)^{\left(\sum_i \omega_i\right)^{-1}} \tag{11}$$

We calculate weights ω_i for each delta that correspond to the similarity of the deltas to their geometric mean, thus outliers get a damped weight while similar values get a higher weighting:

$$|\hat{\boldsymbol{p}}| = \left(\prod_{i=1}^{n} |\Delta\boldsymbol{p}_i|\right)^{n^{-1}} \ , \tag{12}$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n}\left(|\Delta\boldsymbol{p}_i| - |\hat{\boldsymbol{p}}|\right)^2} \ , \tag{13}$$

$$w_i = 1 - s^{-1} \cdot \left(|\Delta\boldsymbol{p}_i| - |\hat{\boldsymbol{p}}|\right) \tag{14}$$

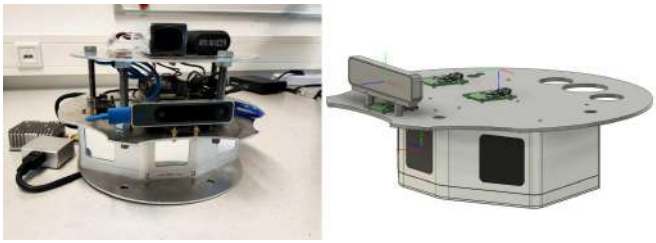


Fig. 3: (Left:) Spherical mobile mapping system without its protecting shell. (Right:) CAD model of the spherical system with sensor frames.

B. Implementation on a spherical system

The implementation on our spherical robot uses two estimators, the IMU operating at 125 Hz defines the measurement $\boldsymbol{X}_0$, and the camera operating at 200 Hz defines the interpolation $\boldsymbol{X}_k$. For the motion model f of the spherical system with known radius $r = 0.145$ m, we assume that rotation leads to translation, thus we calculate the estimated model delta using the arc length of rotation:

$$f(\Delta\boldsymbol{X}_0, \Delta\boldsymbol{X}_k) = \left[\Delta\boldsymbol{R}_e,\ r \cdot \angle(\Delta\boldsymbol{R}_e) \cdot \frac{\Delta\boldsymbol{p}_0 + \Delta\boldsymbol{p}_k}{|\Delta\boldsymbol{p}_0 + \Delta\boldsymbol{p}_k|}\right]^\tau, \tag{15}$$

where $\angle(\cdot)$ denotes the angle around the axis described by the rotation matrix. Note that we just defined the model rotation $\Delta\boldsymbol{R}_m$ from Equation (8) to be equal to $\Delta\boldsymbol{R}_e$ from Equation (9), as the orientation estimation is considered sufficiently reliable.

The simplicity of the filter design allows for the introduction of simple but effective design choices. As an example we notice that our IMUs tends to drift without the use of a magnetometer, especially in the yaw-axis, whereas the tracking camera does not. Due to the background of our spherical system, we do not want to use the magnetometers by design. Thus, we must rely more on the camera estimations for the yaw angle, which is why we exchange the estimation of the rotation delta in Equation (9). Instead of only using Slerp, which is more universal, we first use Slerp and then replace the yaw-part of the resulting delta with the interpolated camera yaw delta. Hence, the change in yaw is only estimated via the camera.

IV. Experiments and Evaluation

Qualitative results are presented in Figure 4, i.e., the resulting point clouds when applying the trajectory estimates of the different estimators. The IMU-based approach (a) suffers from drift in the yaw axis and overestimates the scale of the trajectory. The visual-inertial (b) tracking approach tends to jump whenever the camera looses track, which happens quite often given the unfavorable type of sensor motion. Our proposed Delta-filter (c) combines both trajectories in real-time at 125 Hz on a Raspberry Pi 4, gets rid of the drift and jumps, and estimates the scale of the trajectory better. The following sections quantify the results using ground truth trajectories and maps.

A. Error metrics

To quantify the quality of pose estimation, we use two principal approaches: On the one hand, we measure ground truth trajectories with an Optitrack system using IR reflectors. On the other hand, we also compare the resulting point clouds against ground truth measurements in larger environments, when Optitrack is no longer available. We denote the ground truth trajectory $\boldsymbol{X}_{\mathrm{ref}} = [\boldsymbol{R}_{\mathrm{ref}}, \boldsymbol{p}_{\mathrm{ref}}]^\tau$, and the other estimated trajectories $\boldsymbol{X}_{\mathrm{est}} = [\boldsymbol{R}_{\mathrm{est}}, \boldsymbol{p}_{\mathrm{est}}]^\tau$. For each timestamp in the ground truth trajectory, we sample the closest pose in time from the estimated trajectory for correspondence. Note that all the trajectories must be aligned with the ground truth trajectory. Therefore we align the origins of the trajectories first, as we know that all trajectories started from the same point. Afterwards we rotate around the shared origin using a least-squares alignment according to Umeyama [43]. Note that we only use the estimated rotation of the Umeyama method, since we already aligned the origins. From this point, we use Grupps [44] software for trajectory evaluation. The resulting point clouds are aligned to ground truth using the well-known Iterative Closest Points (ICP) algorithm. We use 3DTK [45] for the processing of the point clouds.

1) Absolute position error: The absolute position error (APE) represents the error of the translation estimation and is given by

$$\mathrm{APE}_i = |\boldsymbol{p}_{\mathrm{est},i} - \boldsymbol{p}_{\mathrm{ref},i}| \,. \tag{16}$$

2) Relative pose error: The relative pose error (RPE) represents the error of the orientation estimation and is given by

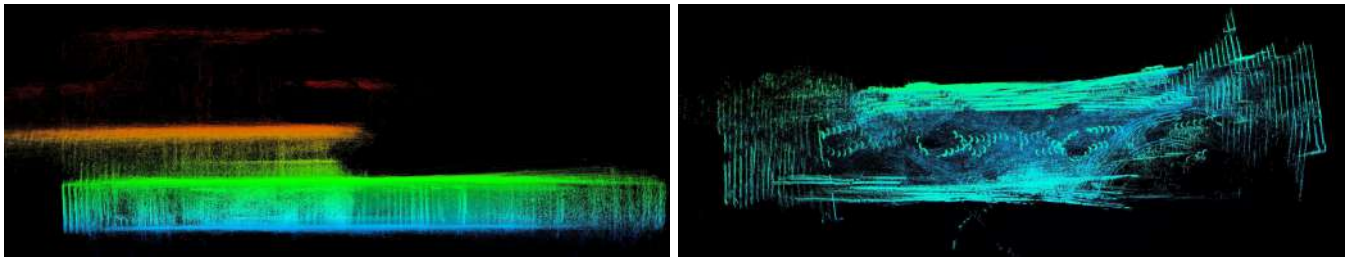
$$\mathrm{RPE}_i = \left| \angle \left(\left(\boldsymbol{R}_{\mathrm{ref},i}^{-1} \boldsymbol{R}_{\mathrm{ref},i-1} \right)^{-1} \left(\boldsymbol{R}_{\mathrm{est},i}^{-1} \boldsymbol{R}_{\mathrm{est},i-1} \right) \right) \right| \,. \tag{17}$$

3) Point cloud error: The point cloud error represents the root of the mean squared point-to-point errors (RMSE). Suppose, after matching with ICP, there are N corresponding model- and data-points in the same coordinate frame, denoted $\boldsymbol{m}_i, \boldsymbol{d}_i \in \mathbb{R}^3$ respectiveley. Then, the root mean squared error is given by

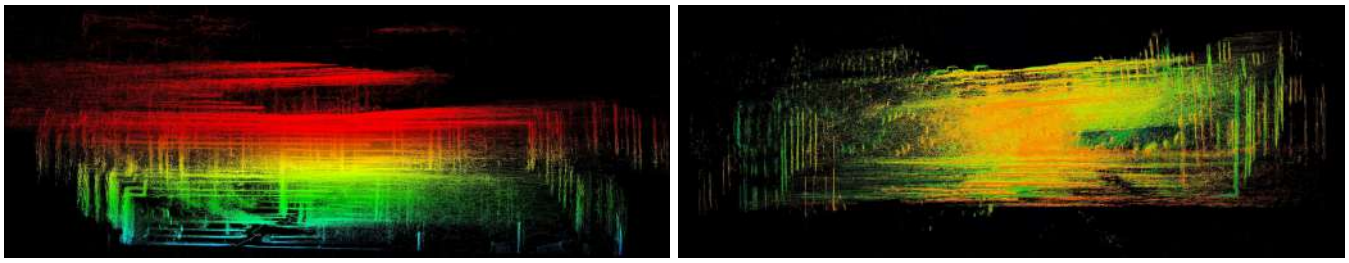
$$\mathrm{RMSE} = \sqrt{\frac{1}{N} \sum_{i=0}^{N} |\boldsymbol{m}_i - \boldsymbol{d}_i|^2} \tag{18}$$

B. Experiments

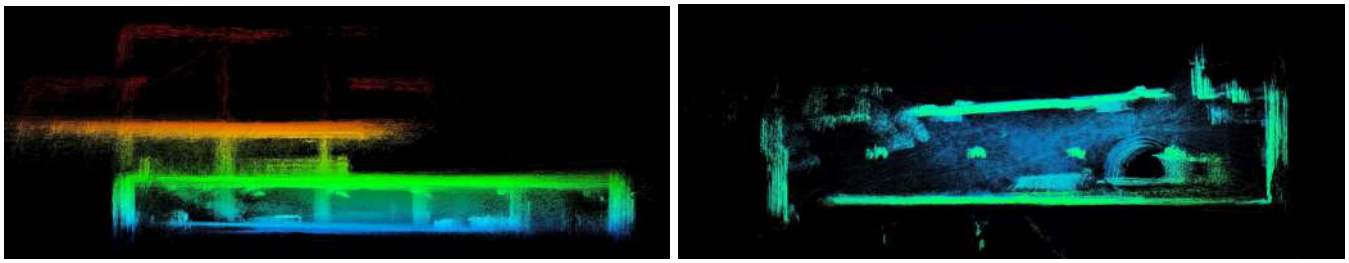
The experiments consist of three types of motion: rolling a straight line slowly, fast, and driving curves at moderate speed. In the first two experiments, an OptiTrack system is available to capture ground truth trajectories, such that we are able to use Equations (17) and (16). However, in the last experiment (driving curves), the environment and trajectory is larger, making the OptiTrack system unavailable. In this experiment, we use a Riegl VZ-400 terrestrial laser-scanner (TLS) with an angular resolution of $0.04°$ and accuracy of $5\,\mathrm{mm}$ to provide accurate ground truth point clouds. As our system is equipped with a laser-scanner (see Figure 3), we



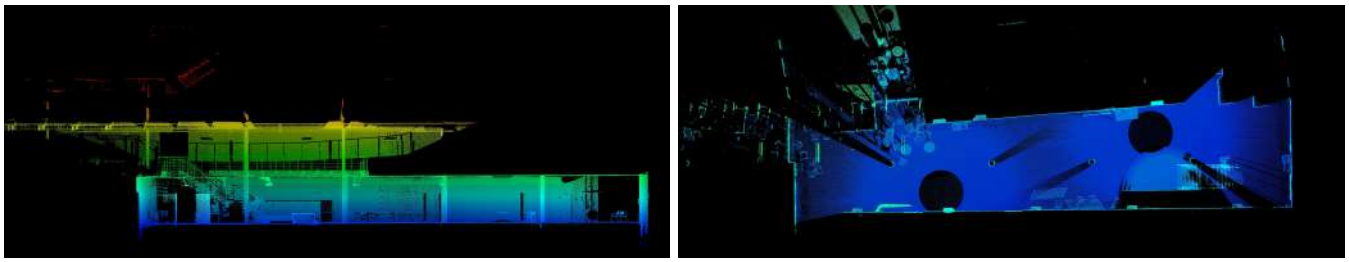
(a) Dead-reckoning IMU-only based pose estimation due to Zevering et al. [42]



(b) Visual-inertial tracking using an Intel RealSense T265 stereo-camera



(c) The proposed Delta-filter using (a), (b), and a motion model



(d) Ground truth point cloud available from a RIEGL VZ-400 TLS

Fig. 4: Resulting point clouds when using three different estimators (a), (b), and (c) are orthographically visualized. A ground truth point cloud is shown in (d). Images in one column were shot from the same point of view. The colors in the point clouds denotes height. The left column shows sliced views from the side, wheras the right column shows sliced views from the birdseye perspective.

compare the resulting point cloud to the ground truth map using Equation (18). Both setups are shown in Figure 5.

1) Fast motion: In this experiment, the sphere traversed a distance of approx. $4\,\mathrm{m}$ in about $10\,\mathrm{s}$. Figure 6 shows the APE (16) of all estimators over time. The T265 suffers from the highest error due to tracking loss, which forces it to rely solely on error prone double integration of acceleration measurements. The IMU-based approach show a considerable increase of error due to the accumulated drift. The error of the proposed Delta-filter are orders of magnitude smaller compared to the IMUs and T265. Figure 7 shows the comparison of RPE (17) over time. Note that the Savgol-filter [46] is applied to the error signals. This is because the ground truth orientations from the OptiTrack system contain many outliers due to mirroring of the IR-reflectors on the spherical shell. The Savgol-filter removes the effect of these outliers but preserves the signal tendency. The RPE of all estimators do not differ particularly from each other, which is also evident from the error metrics in Table II. In fact, the RMSE of the RPE of the Delta-filter is between the INS- and T265-solution, which makes sense considering the interpolation in Equation (9).

2) Slow motion: In this experiment, the sphere traversed a distance of approx. $4\,\mathrm{m}$ in about $45\,\mathrm{s}$. Figure 8 shows the comparison of APE over time. The Delta-filter compensates for the linear accumulation of error of the IMU and the

TABLE I: Comparison of the estimated translation of the trajectory produced by the Delta-filter with its two source estimators, based on several statistical metrics. Each column compares three values where lower is better.

Error metrics to ground truth trajectories for fast and slow motion with respect to translation								
Estimator	RMSE [m]		Mean [m]		Std. [m]		Max. [m]	
	Slow	Fast	Slow	Fast	Slow	Fast	Slow	Fast
Dead-reckoning INS	1.713	1.736	1.447	1.291	0.917	1.160	2.882	3.001
Intel T265 Stereo-VIO	4.486	7.441	4.012	5.290	2.008	5.234	5.848	13.549
Proposed Delta-filter	**0.114**	**0.248**	**0.103**	**0.193**	**0.049**	**0.165**	**0.189**	**0.428**

TABLE II: Comparison of the estimated rotation of the trajectory produced by the Delta-filter with its two source estimators, based on several statistical metrics. Each column compares three values where lower is better.

Error metrics to ground truth trajectories for fast and slow motion with respect to rotation								
Estimator	RMSE [deg]		Mean [deg]		Std. [deg]		Max. [deg]	
	Slow	Fast	Slow	Fast	Slow	Fast	Slow	Fast
Dead-reckoning INS	1.389	4.318	1.281	3.270	**0.537**	2.819	**2.653**	8.883
Intel T265 Stereo-VIO	**1.374**	**4.213**	**1.264**	**3.190**	0.541	**2.753**	2.701	**8.852**
Proposed Delta-filter	1.384	4.305	1.273	3.248	0.543	2.825	2.752	9.199

Fig. 5: (Left:) Laboratory test setup in a flycage equipped with an Optitrack system. The sphere has IR reflectors attached to its shell, which are detected by the cameras (red circles). (Right:) Laboratory test setup in the Computer Science building. A RIEGL VZ-400 TLS captures a precise ground truth point cloud for comparison with the spherical mobile mapping system. In both images, motion of the sphere is initiated manually by hand.

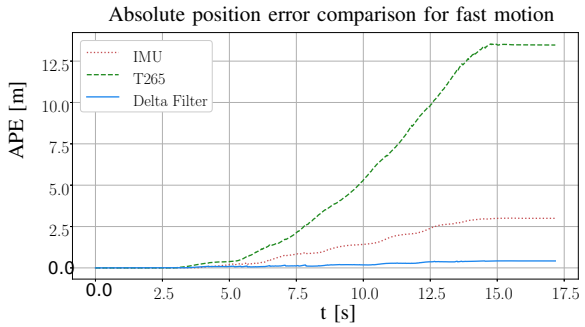


Fig. 6: The absolute position error of all estimators during fast motion over time.

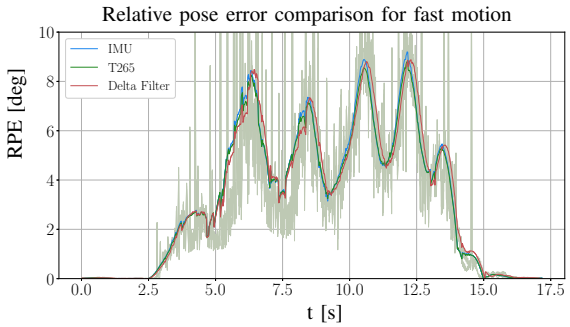


Fig. 7: The relative pose error of all estimators during fast motion over time. The Savgol-filter is applied with a window size of 51 and a polynomial degree of 3 to remove the effect of outliers. In the background the noisy pre-filtered data is shown with low opacity.

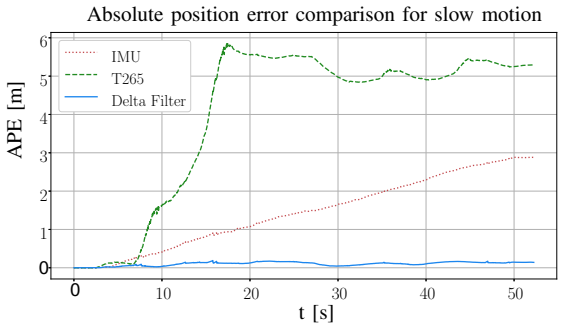


Fig. 8: The absolute position error of all estimators during slow motion over time.

sudden jump of the T265, resulting in a lower overall translation error. Table I confirms this observation. Figure 9 presents the comparison of RPE over time. As mentioned above, the Savgol-filter is applied on the error signals. The orientation errors of all estimators are similar to each other, yet overall smaller compared to fast motion.

3) Curves: Figure 10 shows the result of the point cloud analysis. The error to ground truth is visualized in a point-to-point distance distribution histogram. Note that the large errors at the pillars are caused by global filter drift. On the other hand, the errors at the ceiling of the upper floor are rather caused by missing points in the ground truth. These points, however, are so few that they are only barely visible in the histogram. The mean point-to-point error from Equation (18), which is our accuracy estimate for mapping, is 18.6 cm.

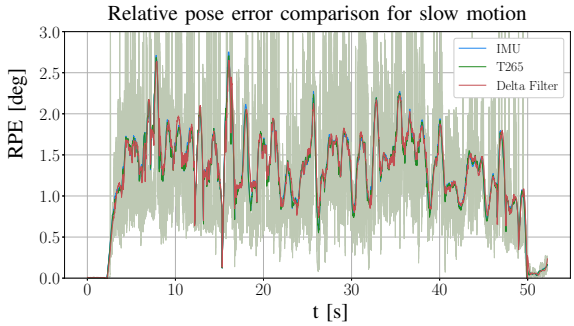


Fig. 9: The relative pose error of all estimators during slow motion over time. The Savgol-filter is applied with a window size of 51 and a polynomial degree of 3 to remove the effect of outliers. In the background the noisy pre-filtered errors are shown with low opacity.

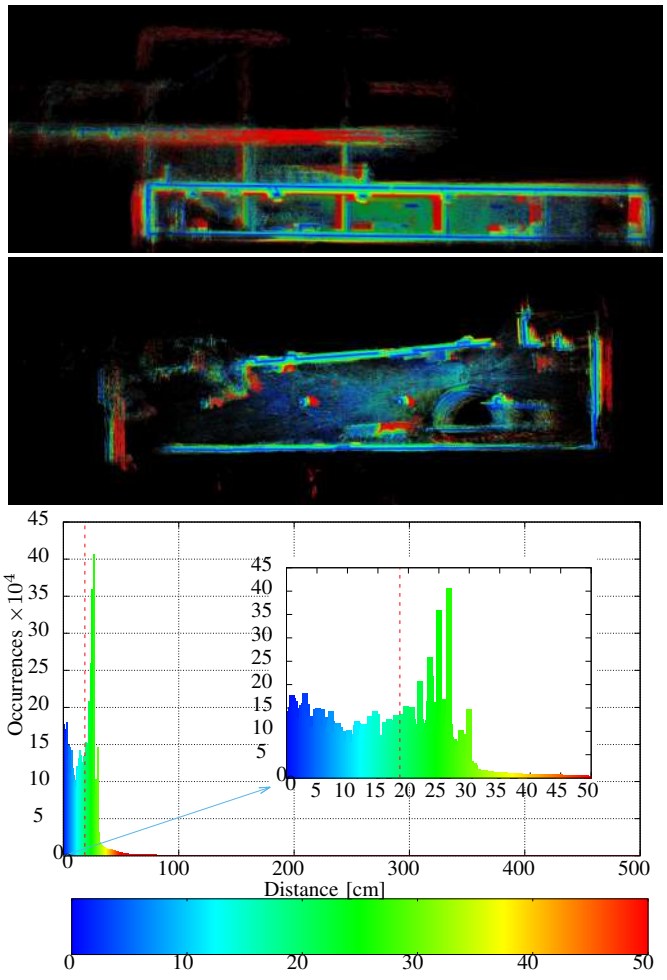


Fig. 10: Resulting point cloud (sliced side-view and birds-eye view) using the trajectory of the proposed filter, as well as a histogram showing a distribution of point-to-point distances. These distances to ground truth are also visualized using color. The red dashed line in the histogram indicates the mean point-to-point error, which is 18.6 cm

.

C. Discussion

The evaluation shows that the Delta-filter significantly improves the pose estimation accuracy, reduces drift, and eliminates jumps. However, despite reducing the drift, all experiments show that the filter still suffers from global drift regarding translation. Furthermore, in the resulting point clouds, the walls appear to be thicker than in the ground truth point cloud, which comes down to two factors: First, the Livox Mid-100 used in the experiment has higher measurement noise, especially when the laser goes through the plastic shell. And second, the extrinsic calibration of the sensors in the spherical system is rather poor, as all the sensors assume to sit inside the center of the sphere.

V. Conclusions

In this paper we addressed the problem of precise, real-time, and onboard localization in 6-DoF for spherical mobile mapping systems. Usually on these systems, the large angular velocities and constant aggressive dynamics when rolling makes state-of-the-art approaches, e.g. INS- or VIO-based solutions, more difficult. We therefore proposed the simple yet effective Delta-filter, which is able to do real-time sensor fusion of an INS- with a VIO-based solution. The filter needs a motion model defined by the user, greatly decreases the INS drift, and gets rid of the jumps caused by the VIO. We showed that the filter is reliable in slow and fast motion, as well as driving curves. Furthermore, we estimated the mapping accuracy of the spherical mobile mapping system to be 18.6 cm without the use of offline-SLAM, which is considered to be an improvement to our previous work. Having such a trajectory estimate brings real-time, highly precise laser-based SLAM for spherical robots closer to reality in the near future. However, needlessly to say, a lot of work remains to be done. In the future, we need to address a proper extrinsic calibration between all sensors to further increase the accuracy. We will also incorporate the LiDAR measurements into the localization by building a real-time onboard laser-based SLAM algorithm designed for spherical systems. This will also include the extension of the motion model using environment data to account for slopes, uneven terrain, or free falling for a short period of time.

References

[1] F. Arzberger, "6-DoF Delta pose filter for sensor fusion." https://github.com/fallow24/delta_pose_filter, 2023.

[2] R. Armour, K. Paskins, A. Bowyer, J. Vincent, and W. Megill, "Jumping robots: a biomimetic solution to locomotion across rough terrain," *Bioinspiration & biomimetics*, vol. 2, no. 3, p. S65, 2007.

[3] K. W. Wait, P. J. Jackson, and L. S. Smoot, "Self locomotion of a spherical rolling robot using a novel deformable pneumatic method," in *2010 IEEE International Conference on Robotics and Automation*, pp. 3757–3762, IEEE, 2010.

[4] R. Mukherjee, "Spherical mobile robot," 2001. US Patent 6,289,263.

[5] R. Chase and A. Pandya, "A review of active mechanical driving principles of spherical robots," *Robotics*, vol. 1, no. 1, pp. 3–23, 2012.

[6] D. Liu, H. Sun, Q. Jia, and L. Wang, "Motion control of a spherical mobile robot by feedback linearization," in *2008 7th World Congress on Intelligent Control and Automation*, pp. 965–970, 2008.

[7] J. Zevering, K. Braun, M. Hesse, K. Mathewos, D. Borrmann, A. , Bredenbeck, and A. Nüchter, "The concept the virtual pose instruction plane (vpip) for controlling rod-driven spherical robots," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023. submitted for review.

[8] D. Borrmann, S. Jörissen, and A. Nüchter, "RADLER – A RADial LasER scanning device," in *Proceedings of the International Symposium on Experimental Research*, (Buenos Aires, Argentina), pp. 655–664, 01 2020.

[9] J. Zevering, A. Bredenbeck, F. Arzberger, D. Borrmann, and A. Nüchter, "Luna-a laser-mapping unidirectional navigation actuator," in *Experimental Robotics: The 17th International Symposium*, pp. 85–94, Springer, 2021.

[10] A. P. Rossi, F. Maurelli, V. Unnithan, H. Dreger, K. Mathewos, N. Pradhan, D.-A. Corbeanu, R. Pozzobon, M. Massironi, S. Ferrari, *et al.*, "Daedalus-descent and exploration in deep autonomy of lava underground structures," 2021.

[11] J. Zevering, D. Borrmann, A. Bredenbeck, and A. Nüchter, "The concept of rod-driven locomotion for spherical lunar exploration robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5656–5663, 2022.

[12] F. Arzberger, A. Bredenbeck, J. Zevering, D. Borrmann, and A. Nüchter, "Towards spherical robots for mobile mapping in human made environments," *ISPRS Open Journal of Photogrammetry and Remote Sensing*, vol. 1, p. 100004, 2021.

[13] F. Arzberger, J. Zevering, A. Bredenbeck, D. Borrmann, and A. Nüchter, "Mobile 3d scanning and mapping for freely rotating and vertically descended lidar," in *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 122–129, 2022.

[14] Xue, Jian-ru and Wang, Di and Du, Shao-yi and Cui, Di-xiao and Huang, Yong and Zheng, Nan-ning, "A vision-centered multi-sensor fusing approach to self-localization and obstacle perception for robotic cars," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 1, pp. 122–138, 2017.

[15] C. Merfels and C. Stachniss, "Sensor fusion for self-localisation of automated vehicles," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 85, pp. 113–126, 2017.

[16] G. Abdi, F. Samadzadegan, and F. Kurz, "Pose estimation of unmanned aerial vehicles based on a vision-aided multi-sensor fusion," in *XXII ISPRS Congress, Technical Commission I*, vol. 41, pp. 193–199, 2016.

[17] Du, Hao and Wang, Wei and Xu, Chaowen and Xiao, Ran and Sun, Changyin, "Real-time onboard 3d state estimation of an unmanned aerial vehicle in multi-environments using multi-sensor data fusion," *Sensors*, vol. 20, no. 3, 2020.

[18] A. Soloviev and M. M. Miller, *Navigation in Difficult Environments: Multi-Sensor Fusion Techniques*, pp. 199–229. New York, NY: Springer New York, 2012.

[19] F. Santoso, M. A. Garratt, and S. G. Anavatti, "Visual–inertial navigation systems for aerial robotics: Sensor fusion and technology," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 260–275, 2017.

[20] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[21] E. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pp. 153–158, 2000.

[22] F. Daum, "Nonlinear filters: beyond the kalman filter," *IEEE Aerospace and Electronic Systems Magazine*, vol. 20, no. 8, pp. 57–69, 2005.

[23] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3565–3572, 2007.

[24] A. Sakai, Y. Tamura, and Y. Kuroda, "An efficient solution to 6dof localization using unscented kalman filter for planetary rovers," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4154–4159, 2009.

[25] G. Ligorio and A. M. Sabatini, "Extended kalman filter-based meth-

[26] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *The International Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, 2011.

ods for pose estimation using visual, inertial and magnetic sensors: Comparative analysis and performance evaluation," *Sensors*, vol. 13, no. 2, pp. 1919–1941, 2013.

[27] G. Huang, K. Eckenhoff, and J. Leonard, *Optimal-State-Constraint EKF for Visual-Inertial Navigation*, pp. 125–139. Cham: Springer International Publishing, 2018.

[28] Liao, Jianchi and Li, Xingxing and Wang, Xuanbin and Li, Shengyu and Wang, Huidan, "Enhancing navigation performance through visual-inertial odometry in gnss-degraded environment," *Gps Solutions*, vol. 25, pp. 1–18, 2021.

[29] N. Abdelkrim, N. Aouf, A. Tsourdos, and B. White, "Robust nonlinear filtering for ins/gps uav localization," in *2008 16th Mediterranean Conference on Control and Automation*, pp. 695–702, 2008.

[30] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[31] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[32] A. J. Haug, *Bayesian estimation and tracking: a practical guide*. John Wiley & Sons, 2012.

[33] R. Mascaro, L. Teixeira, T. Hinzmann, R. Siegwart, and M. Chli, "Gomsf: Graph-optimization based multi-sensor fusion for robust uav pose estimation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1421–1428, 2018.

[34] Nguyen, Thien-Minh and Cao, Muqing and Yuan, Shenghai and Lyu, Yang and Nguyen, Thien Hoang and Xie, Lihua, "Viral-fusion: A visual-inertial-ranging-lidar sensor fusion approach," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 958–977, 2022.

[35] Kim, Kihun and Choi, Hyun-Taek and Lee, Chong-Moo, "Underwater precise navigation using multiple sensor fusion," in *2013 IEEE International Underwater Technology Symposium (UT)*, pp. 1–4, 2013.

[36] Wei Fang and Lianyu Zheng and Xiangyong Wu, "Multi-sensor based real-time 6-dof pose tracking for wearable augmented reality," *Computers in Industry*, vol. 92-93, pp. 91–103, 2017.

[37] J. Zevering, A. Bredenbeck, F. Arzberger, D. Borrmann, and A. Nüchter, "Imu-based pose-estimation for spherical robots with limited resources," in *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 1–8, IEEE, 2021.

[38] S. Madgwick *et al.*, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," *Report x-io and University of Bristol (UK)*, vol. 25, pp. 113–118, 2010.

[39] Min, Hyung Gi and Jeung, Eun Tae, "Complementary filter design for angle estimation using mems accelerometer and gyroscope," *Department of Control and Instrumentation, Changwon National University, Changwon, Korea*, pp. 641–773, 2015.

[40] J. Borenstein and L. Feng, "Gyrodometry: a new method for combining data from gyros and odometry in mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, pp. 423–428 vol.1, 1996.

[41] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.

[42] J. Zevering, A. Bredenbeck, F. Arzberger, D. Borrmann, and A. Nüchter, "Imu-based pose-estimation for spherical robots with limited resources," in *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 1–8, 2021.

[43] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[44] M. Grupp, "evo: Python package for the evaluation of odometry and SLAM.." `https://github.com/MichaelGrupp/evo`, 2017.

[45] A. Nüchter and K. Lingemann, "3DTK—The 3D Toolkit. 2011." `https://slam6d.sourceforge.io/index.html`, 2011.

[46] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures.," *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

Dataset Generation for Deep Visual Navigation in Unstructured Environments

Yoshinobu Uzawa, Shigemichi Matsuzaki, Hiroaki Masuzawa, and Jun Miura
Department of Computer Science and Engineering
Toyohashi University of Technology

***Abstract*— Visual navigation in unstructured environments is one of the essential functions of autonomous mobile robots. Most existing navigation methods are based on extracting traversable regions, and the visibility of the traversable region is crucial. However, in situations like plant-rich environments, such visibility is not guaranteed. A possible option is to directly infer the moving direction from images using deep learning techniques. Although this approach is promising because of a wide variety of application environments, obtaining a high-quality, large dataset will be an issue. This paper describes a data acquisition system that can easily collect various types of sensor data and a dataset generation method. We evaluate the generated datasets in several aspects. We also release the datasets, including image-path pairs and various raw sensor data.**

I. Introduction

Autonomous navigation has been a core research topic in robotics and transportation. One of the essential functions is to recognize the surrounding environment and plan an appropriate motion. Many of the existing approaches rely on local geometrical mapping (i.e., obstacle detection and path planning) [1], traversable region segmentation [2], [3], or road boundary detection [4], [5], [6], [7], [8]. Furthermore, they deal with structured or semi-structured environments. Those methods assume that navigational features such as traversable regions and road boundaries are visible. However, in unstructured environments, such as the ones shown in Fig. 1, this assumption does not hold, and navigation relying on such features will fail. Geometry-based approaches will fail without semantic information, too, if plant regions are considered obstacles.

Humans can traverse the environments, as shown in the figure, probably based on learning from experience. We can discriminate traversable plans from untraversable ones and directly choose feasible paths from a view. This paper deals with the latter: end-to-end path estimation from an image. Generating a dataset is crucial to adopt this strategy. Manual annotation is tedious and time-consuming, so we develop a system for quickly collecting data and automatically generating a dataset. We extend our previous work on dataset generation [9] so that we can collect data in a more variety of environments. We also evaluate the dataset with more in-deep criteria and with autonomous navigation in several environments.

(a) Greenhouse. (b) Nature trail.

Fig. 1. Plant-rich environment examples.

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

The rest of the paper is organized as follows. Sec. II surveys related papers. Sec. III explains our path estimation and robot control approach. Sec. IV details the data collection system and the dataset generation method. Sec. V shows evaluation experiments. Sec. VI concludes the paper and discusses future work.

II. Related Work

A. Datasets for autonomous driving

With the advancement of autonomous driving and deep learning, there have been many datasets for traffic scene recognition, including Cityscapes [10], KITTI [11], and nuScenes [12]. We can also use traffic scene simulators such as CARLA [13] for simultaneously generating sensor data and vehicle control commands. Such datasets cannot directly be applied to unstructured and specialized scenes.

B. Datasets for unstructured environments

There are several datasets for navigation in unstructured environments. Wigness et al. [14] made a dataset of RGB and semantic segmentation images. Jiang et al. [15] used various sensors such as LiDARs and stereo cameras for constructing a dataset. Valada et al. [16] made a dataset in forest environments using multispectral cameras. Although these datasets include semantic segmentation images and are helpful for traversable region detection-based navigation, we cannot directly apply them to situations with frequent occlusions of traversable regions.

C. Automatic dataset generation

Meyer et al. [17] used the vehicle trajectory for automatic annotation. Onozuka et al. [18] took a similar approach to choose preferred traverse regions for a mobile robot. Wellhausen et al. [19] automatically selected traversable regions from the footpoints of a quadruped robot. Giusti et al.

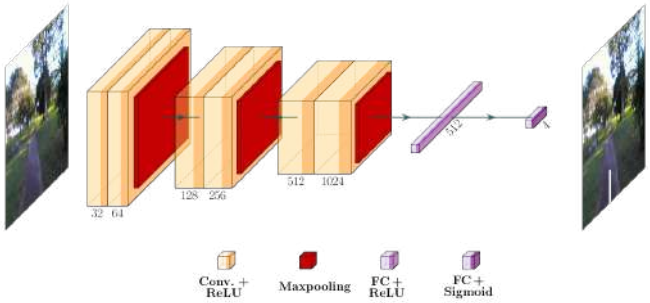


Fig. 2. Moving direction estimation network.

[20] determine a label of three directions using images taken from three wearable cameras. Sanchez et al. [21] constructed a natural simulated environment in the Gazebo simulator and used it for automatic annotation. Most of the existing works deal with the annotation of semantic labels. Our proposed approach enables the annotation of estimated path direction applicable to the scene where traversable regions may be heavily occluded.

III. End-to-end Path Direction Estimation and Robot Control

This section briefly explains our approach to path direction estimation and robot control [9]. We use a CNN model shown in Fig. 2. The input is a 240×128 RGB image, and the output is a pair of the start and end points representing the estimated path direction. The vertical positions of the points are currently set at the bottom of the image and 20% of the image height from the bottom, respectively. These positions should be determined considering several factors, such as the camera height and the field of view.

The condition of the training is as follows: the batch size is 16; the number of epochs is 100; the initial training rate is 10^{-3} and decayed by the exponential learning rate scheduling [22]; Adam [23] is used as the optimization method.

Once the path is predicted in the image, it is mapped onto the ground plane using the camera's extrinsic parameters. A simple control law [9] is then applied to determine the translational and the rotational velocity.

IV. Dataset Generation

A. Data collection system

We use an RGB-D camera with IMU for collecting data for our purpose (i.e., image-based navigation). We additionally use a GNSS sensor and a 3D LiDAR for possible future comparison with other methods. Table I lists the sensors used. We collected data in two ways (see Fig. 3). One way uses a mobile robot for relatively planar terrains such as outdoor roads or greenhouses, and the other is human-carried for rough terrain such as nature trails.

In this research, we collected data for three different locations: a university greenhouse (called *greenhouse*), a university park (*park*), and a nearby natural trail (*trail*). Example scenes of the locations are shown in Fig. 7. Table IV-A summarizes the collected data.

TABLE I
Sensors of the data collection system.

sensor	product	manufacturer
RGB-D with IMU	Realsense D435i	Intel
GNSS	ZED-F9P	u-blox
3D LiDAR	VLP-16	Velodyne

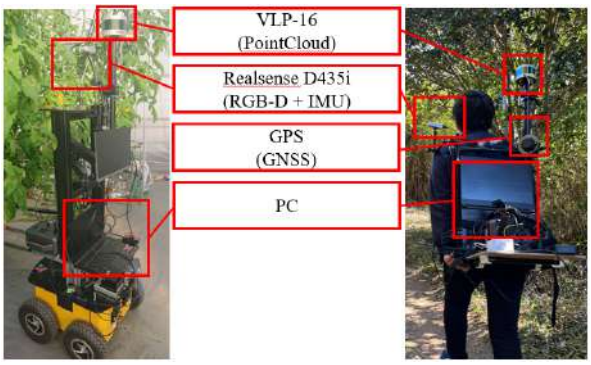


Fig. 3. Data collection system used in two ways.

B. Dataset generation method

Fig. 4 shows an outline of dataset generation. We apply RTAB-Map [24], a visual SLAM (vSLAM) method, to a stream of RGB-D and IMU data to estimate the camera's trajectory. Fig. 5 is an example SLAM result, showing a 3D map and the camera trajectory. The calculated camera locations are mapped onto images such that a future camera motion (i.e., a sequence of camera locations in the subsequent frames) can be viewed on each image. These mapped points are used for annotating the motion direction in each image.

TABLE II
Collected data.

Location	Data size [GB]	Time of the day	Duration [min]
greenhouse	28.3	daytime & late afternoon	27
park	133.5	daytime	18
trail	610.6	daytime	70

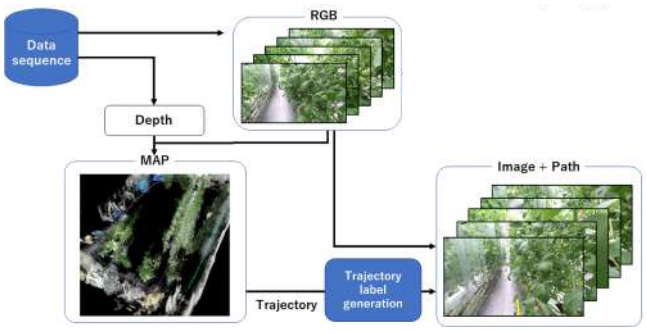


Fig. 4. Outline of dataset generation.

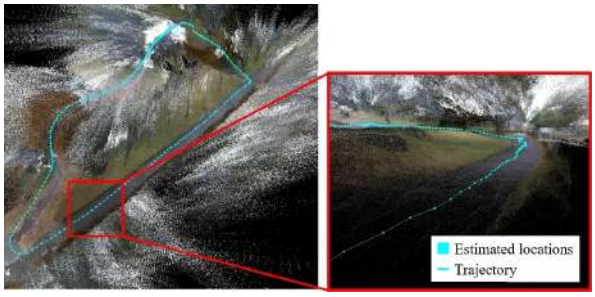


Fig. 5. Example vSLAM result.

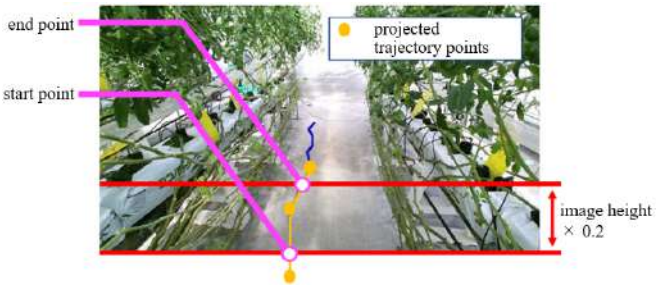


Fig. 6. Path label calculation.

Mapping from a point in the scene to the corresponding image point is given by:

$$s\,(u\ v\ 1)^t = K\,{}^c_r T^{-1}\,{}^r_w T^{-1}\,(X_w\ Y_w\ Z_w\ 1)^t\,,$$

where (u,v) is an image point, (X_w, Y_w, Z_w) is a scene point, K is the intrinsic parameter of the camera, ${}^c_r T$ and ${}^w_r T$ are transformation from the robot coordinates to the camera coordinates and that from the robot coordinates to the world coordinates, respectively. K and ${}^c_r$ are calibrated in advance. ${}^r_w T$ is obtained by the vSLAM.

We then approximate the projected trajectory by a line segment, which is then used as a *path label*. Fig. 6 shows the definition of the line segment's start and end point. As described above, the vertical positions of the points are set at the bottom of the image and 20% of the image height from the bottom, respectively. The x coordinate of the points is calculated as the intersections between the bi-linear approximated trajectory and the horizontal lines at the two vertical positions. Fig. 7 shows several examples of data collection, trajectory estimation and projection, and path label annotation for the three experimental locations (greenhouse, park, and trail).

We adopt image cropping as data augmentation to make the model more robust to deviation from the ideal paths. The images are cropped with a window with the size of 80% of the original image, and the window is shifted horizontally and vertically by a length of 10% of the width and the height, respectively. Table III shows the numbers of training and test images for each location after augmentation. In addition, we manually annotate all images for evaluation purposes (see Sec. V).

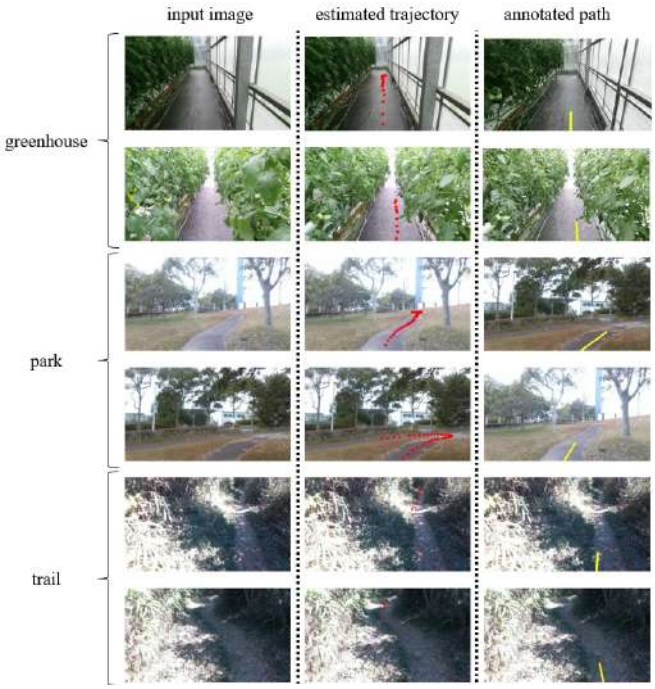


Fig. 7. Examples of generated data.

TABLE III
NUMBER OF IMAGES IN THE DATASETS.

	greenhouse	park	trail
training	4338	2734	13179
test	213	341	543

V. EXPERIMETAL EVALUATION

A. Evaluation criteria

We evaluate the generated datasets using the following four criteria:

- The time cost of dataset generation. Table IV compares the time for manual and automatic annotation. The time for automatic annotation includes trajectory estimation by vSLAM, trajectory projection onto images, and line segment approximation. Automatic annotation can save about two-thirds of annotation time.
- The similarity of automatic and manual annotation results.
- Accuracy of predicted path labels using the model trained with the automatically generated datasets.
- Autonomous navigation. We control an actual mobile robot using the model trained in the previous item.

TABLE IV
COMPARISON OF TIME FOR ANNOTATION.

	Greenhouse [min]	Park [min]	Trail [min]
Manual	150	90	120
Automatic	40	30	50

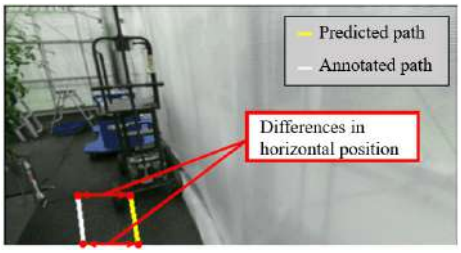
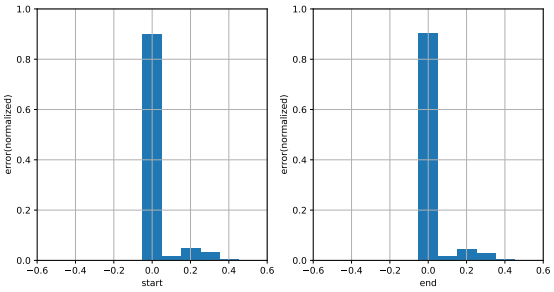
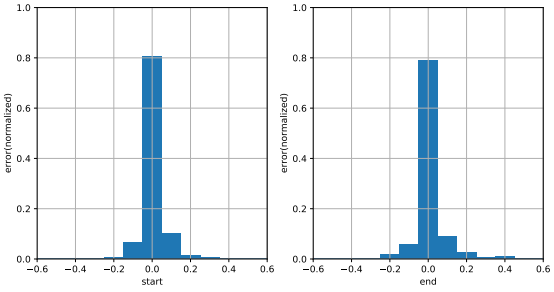


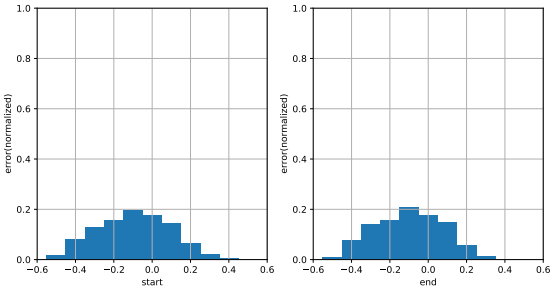
Fig. 8. Differences in horizonal position of endpoints.



(a) Greenhouse dataset.

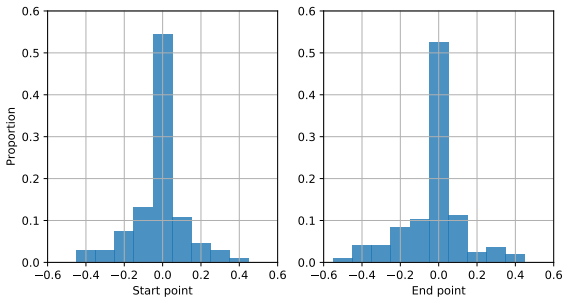


(b) Park dataset.

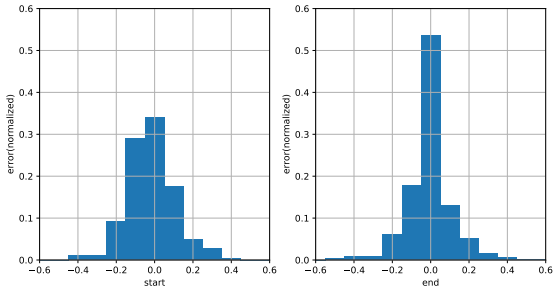


(c) Trail dataset.

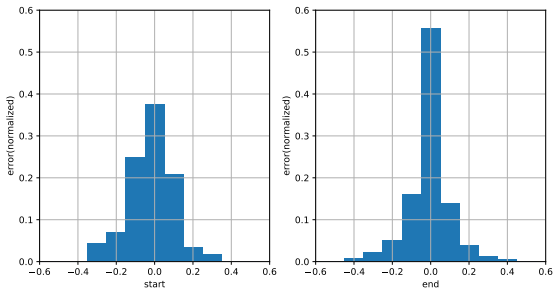
Fig. 9. Distribution of horizontal differences in the start (left) and end (right) point for the datasets.



(a) Greenhouse dataset.



(b) Park dataset.



(c) Trail dataset.

Fig. 10. Distribution of horizontal differences in the predicted start (left) and end (right) point using the model with automatically genereated dataset.

B. Similarity between automatic and manual annotation

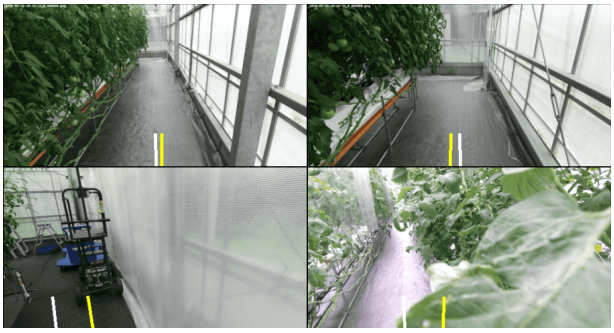
In criterion 2, we calculate the differences in the horizontal position of the start and the end point of the manual annotation and the automatic annotation (see Fig. 8) using the images for training.

Fig. 9 shows the distributions of differences for the three locations. The difference is small enough in the greenhouse dataset because the path direction is almost always clear, even under occlusions. The difference is also small for the park dataset because the path direction is apparent even with occasional unclear boundaries. In the case of the trail dataset, however, the difference is significant; possible causes are much unclear boundaries and difficulty in recognizing the traversable regions only from images. The automatically generated path labels could sometimes be more accurate than the manually-annotated ones because the former is based on actual human traversals. In addition, non-smooth traversal paths of humans in uneven terrains might be another cause of significant differences.

C. Accuracy of predicted path labels

In criterion 3, we evaluate the performance of trained models. We train the model (see Fig. 2) with the automatically annotated and manually annotated datasets. Fig. 10 shows the accuracy of models trained with the automatically annotated dataset. The accuracy is represented by a similar metric as before, that is, the differences of the horizontal positions with respect to the manual annotations of the test data.

For the greenhouse dataset, 80% of predictions provide



(a) Greenhouse dataset.

(b) Park dataset.

(c) Trail dataset.

Fig. 11. Example prediction results by the models trained with automaticallygen-erated datasets. White and yellow lines indicate the manually-annotated and the predicted path segments, respectively.

accuracy within ±20%. Fig. 11(a) shows example prediction results. This environment is regularly structured, and prediction is stable. However, there are cases where the prediction performs poorly due to an irregular path shape (bottom left in the image) or a heavy occlusion (bottom right).

For the park dataset, 85% of predictions provide accuracy within ±20%. Fig. 11(b) shows example prediction results. The distribution of the start points is larger than that of the endpoints (see Fig. 10(b)). This is probably because the variation of the path width is large, and its effect on the accuracy will be larger for the start point (e.g., top right in Fig. 11(b)).

For the trail dataset, 70% of predictions provide accuracy within ±20%. Fig. 11(c) shows example prediction results. The results have a similar tendency with the park case but with larger errors, as the scene is more difficult to learn.

Table V compares the datasets quantitatively. For each

TABLE V

QUANTITATIVE COMPARISON OF DATASETS.

location	dataset	mean error (variance) start	end
greenhouse	manual	-0.012 (0.134)	-0.036 (0.159)
	automatic	0.005 (0.134)	-0.027 (0.157)
park	manual	-0.048 (0.098)	0.001 (0.001)
	automatic	-0.017 (0.123)	-0.001 (0.150)
trail	manual	0.047 (0.126)	0.029 (0.126)
	automatic	0.080 (0.150)	0.092 (0.155)

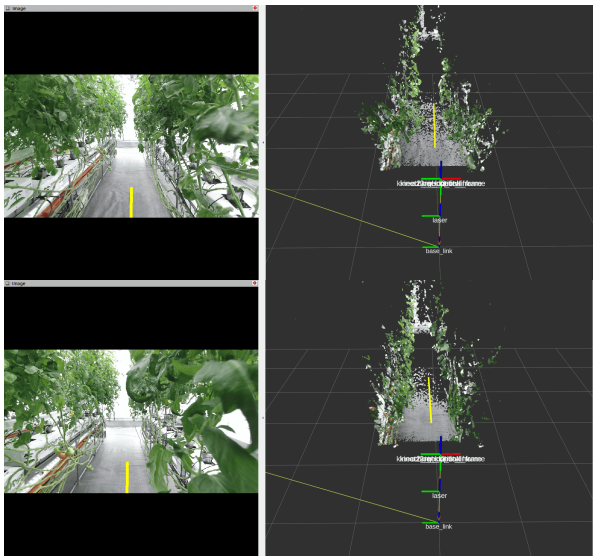


Fig. 12. Navigation experiments in the greenhouse.

location, we trained two models, one with the automatically generated dataset and the other with the manually annotated dataset. The table summarizes results for all combinations of locations and datasets. The automatic dataset generation method is a reasonable option considering the reduced generation cost (see Table IV).

D. Autonomous navigation

We conducted online path following experiments in the greenhouse and the park locations. Fig. 12 shows two snapshots during the experiments in the greenhouse; prediction in the image and the scene are shown. We conducted five trials of navigation on different paths between crop rows with a length of approximately 10 [m]. The robot completed the navigation in all the trials. When approaching the end of a row, the accuracy of path estimation degraded as the view from there is very different from the one in the training set.

Fig. 13 shows four snapshots during the experiments in the park; predictions in the images are shown. Navigation was successful on unbranched roads under fair lighting conditions (see Fig. 13(a)). The navigation was not successful under bad lighting conditions (see Fig. 13(b)) or in unclear boundary cases. We could improve the performance by adding more training data with various road shapes and lighting conditions

(a) Successful navigation.

(b) Failed path prediction.

Fig. 13. Navigation experiments in the park.

and adopting more data augmentation methods.

VI. Conclusions and Discussion

This paper described a dataset generation method for robot navigation in unstructured environments. Considering the possibility of the low visibility of traversable regions, we adopt a strategy of directly estimating the path labels from images using a CNN-based network. We developed a data collection system and a method of automatically generating path labels from the estimated camera trajectory obtained by a visual SLAM method. What we have to do is to move the robot or walk with the system for dataset generation. We tested our approach in three locations, with evaluations with four criteria, including autonomous navigation experiments. The evaluation results show the feasibility of the proposed approach. We also release the datasets, including image-path pairs and various raw sensor data[1].

The current method deals with only unbranched roads and cannot generate path labels for the other types, such as branches and junctions. As the path labels are based on the actual trajectory of the camera, the system needs to either recognize the other types or utilize some user's inputs to include road type variations. The navigation system also needs to be extended to accept operational commands. Increasing the variety of object appearance in the dataset is desirable for more robust navigation. This would require getting data under various weather conditions or introducing various data augmentation methods.

References

[1] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza, "Introduction to autonomous mobile robots, 2nd edition," 2011.

[2] M. Teichmann, M. Weber, M. Zollner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *Proceedings of 2018 IEEE Intelligent Vehicles Symp.*, 2018, pp. 1013–1020.

[3] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "Lidar-camera fusion for road detection using fully convolutional neural networks," *Robotics and Autonomous Systems*, vol. 111, 11 2018.

[4] E. Dickmanns, "The development of machine vision for road vehicles in the last decade," in *Proceedings of 2002 IEEE Intelligent Vehicle Symp.*, vol. 1, 2002, pp. 268–281.

[5] W. Wijesoma, K. Kodagoda, and A. Balasuriya, "Road boundary detection and tracking using ladar sensing," *IEEE Trans. on Robotics and Automation*, vol. 20, no. 3, pp. 456–464, 2004.

[6] P. Sun, X. Zhao, Z. Xu, R. Wang, and H. Min, "A 3d lidar data-based dedicated road boundary detection algorithm for autonomous vehicles," *IEEE Access*, vol. 7, pp. 29 623–29 638, 2019.

[7] Y. Matsushita and J. Miura, "On-line road boundary modeling with multiple sensory features, flexible road model, and particle filter," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 274–284, 2011.

[8] Y. Nakayama and J. Miura, "3d road boundary estimation using 3d lidar with scanline-wise 1d deep feature and particle filtering," in *Proceedings of 2021 European Conf. on Mobile Robots (ECMR2021)*, 2021.

[9] Y. Uzawa, S. Matsuzaki, H. Masuzawa, and J. Miura, "End-to-end path estimation and automatic dataset generation for robot navigation in plant-rich environments," in *Proceedings of 17th Int. Conf. on Intelligent Autonomous Systems (IAS-17)*, 2022.

[10] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.

[11] A. Geiger, P. Lenz, and R. Urtarun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of 2012 IEEE Conf. on Computer Vision and Pattern Recognition*, 2012.

[12] H. Caesar, V. Bankiti, A. Lang, S. Vora, V. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proceedings of 2017 Annu. Conf. on Robot Learning*, 2017.

[14] M. Wigness, S. Eum, J. G. Rogers, D. Han, and H. Kwon, "A rugd dataset for autonomous navigation and visual perception in unstructured outdoor environments," in *International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[15] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, "Rellis-3d dataset: Data, benchmarks and analysis," 2020.

[16] A. Valada, G. Oliveira, T. Brox, and W. Burgard, "Deep multispectral semantic scene understanding of forested environments using multimodal fusion," in *International Symposium on Experimental Robotics (ISER)*, 2016.

[17] A. Meyer, N. O. Salscheider, P. F. Orzechowski, and C. Stiller, "Deep Semantic Lane Segmentation for Mapless Driving," in *IEEE International Conference on Intelligent Robots and Systems*, 2018, pp. 869–875.

[18] Y. Onozuka, R. Matsumi, and M. Shino, "Weakly-supervised recommended traversable area segmentation using automatically labeled images for autonomous driving in pedestrian environment with no edges," *Sensors (Switzerland)*, vol. 21, no. 2, pp. 1–22, 2021.

[19] L. Wellhausen, A. Dosovitskiy, R. Ranftl, K. Walas, C. Cadena, and M. Hutter, "Where should i walk(Predicting terrain properties from images via self-supervised learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1509–1516, 2019.

[20] A. Giusti, J. Guzzi, D. C. Ciresan, F.-L. He, J. P. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016. [Online]. Available: https://www.ifi.uzh.ch/dam/jcr:38859211-28c8-43c9-af40-cb6d5bea0bbe/RAL16{_}Giusti.pdf

[21] M. Sánchez, J. Morales, J. Martinez, J. Fernández-Lozano, and A. García-Cerezo, "Automatically annotated dataset of a ground mobile robot in natural environments via gazebo simulations," *Sensors*, vol. 22, no. 15, 2022.

[22] Z. Li and S. Arora, "An Exponential Learning Rate Schedule for Deep Learning," *arXiv:1910.07454*, 2019. [Online]. Available: https://arxiv.org/pdf/1910.07454.pdf

[23] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] M. Labbé and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based slam," in *Proceedings of 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 2661–2666.

[1] https://github.com/ActiveIntelligentSystemsLab/Trajectory_generate.

Towards camera parameters invariant monocular depth estimation in autonomous driving

Karlo Koledić, Ivan Marković, and Ivan Petrović[1].

***Abstract*— Monocular depth estimation is an effective approach to environment perception due to simplicity of the sensor setup and absence of multisensor calibration. Deep learning has enabled accurate depth estimation from a single image by exploiting semantic cues such as the sizes of known objects and positions on the ground plane thereof. However, learning-based methods frequently fail to generalize on images collected with different vehicle-camera setups due to the induced perspective geometry bias. In this work, we propose an approach for camera parameters invariant depth estimation in autonomous driving scenarios. We propose a novel joint parametrization of camera intrinsic and extrinsic parameters specifically designed for autonomous driving. In order to supplement the neural network with information about the camera parameters, we fuse the proposed parametrization and image features via the novel module based on a self-attention mechanism. After thorough experimentation on the effects of camera parameter variation, we show that our approach effectively provides the neural network with useful information, thus increasing accuracy and generalization performance.**

I. INTRODUCTION

Scene depth is a key information in many three dimensional reconstruction and perception tasks in robotics, autonomous driving, and virtual reality. While fusion of different sensor modalities increases robustness and accuracy, depth estimation from camera data is effective due to the richness of information and relative simplicity of the sensor setup. Traditionally, scene depth is estimated within geometric Structure-from-Motion or Visual Simultaneous Localization and Mapping frameworks. Sparse or dense correspondences are established across different camera poses, enabling triangulation and subsequent optimization. However, such systems usually calculate depth for a limited set of sparse correspondences with robustness issues due to challenging scenarios such as occlusions and textureless regions. Given that, deep learning-based methods have been increasingly used for monocular depth estimation (MDE). Even though depth estimation from a single image is an ill-posed problem, neural networks leverage large amount of data in order to learn semantic and geometric cues, such as the size of known objects or position on the ground plane [1], and use them to infer the scene depth.

Early MDE works [2], [3] establish a standard supervised learning procedure to directly regress a depth map within an encoder-decoder architecture, often with residual connections. Various attempts have been made in order to improve the results, with addition of recurrent neural networks [4]–[6], conditional random fields [7]–[10] or adversarial training [11], [12] into the architecture. Recently, with advancements of transformers [13] in vision tasks [14], many methods take advantage of the global receptive field of the transformer that naturally complements locality of the convolutions, thus consequently achieving state-of-the-art results [15]–[17]. However, the main drawback of such supervised methods is the necessity of ground truth data acquisition, which is often sparse and difficult to collect. This constrains the training data to a narrow distribution leading to overfitting and inaccurate generalization on unseen environments. To that end, self-supervised methods [18]–[21] use view synthesis of nearby frames as a supervision signal, removing the requirement of ground truth data during training.

Even though self-supervised methods make data collection within distinctive environments relatively straightforward, effects of different camera extrinsic and intrinsic parameters during test time are often ignored. As the training data is usually collected with a single vehicle-camera setup, networks tend to overfit due to the perspective geometry bias in gathered data [22]. Embedding of known focal length [23], camera intrinsics [24] or camera extrinsics [25] within neural networks, along with usage of diverse synthetic training data, has shown to improve generalization capabilities. Although synthetic data has been widely used for MDE in automotive scenarios [26]–[29], variation in camera parameters has been left largely unexplored. In theory, if trained on diverse enough real-world data containing various camera parameters, the network could learn to estimate depth for the camera parameters within the training set; however, we argue that the process of data acquisition with sufficiently diverse camera parameters in distinctive environments is infeasible, which is why we use synthetic data in the present work.

In this paper, we propose a novel approach for camera parameters invariant MDE for automotive driving scenarios. We demonstrate the effects of the camera parameters variation in MDE and design a novel architecture which enhances the generalization capabilities of the system. We test and train our method on synthetic data, while designing the architecture to support further work for domain adaptation to real data. Our main contributions are as follows:

- a novel parametrization of known camera intrinsic and extrinsic parameters as depth of the ground plane, which has a strong semantic and geometric meaning in MDE

[1]Authors are with University of Zagreb Faculty of Electrical Engineering and Computing, Laboratory for Autonomous Systems and Mobile Robotics, Zagreb, Croatia {name.surname@fer.hr}.

This research has been funded by the H2020 project AIFORS under Grant Agreement No 952275 and supported by the European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS).

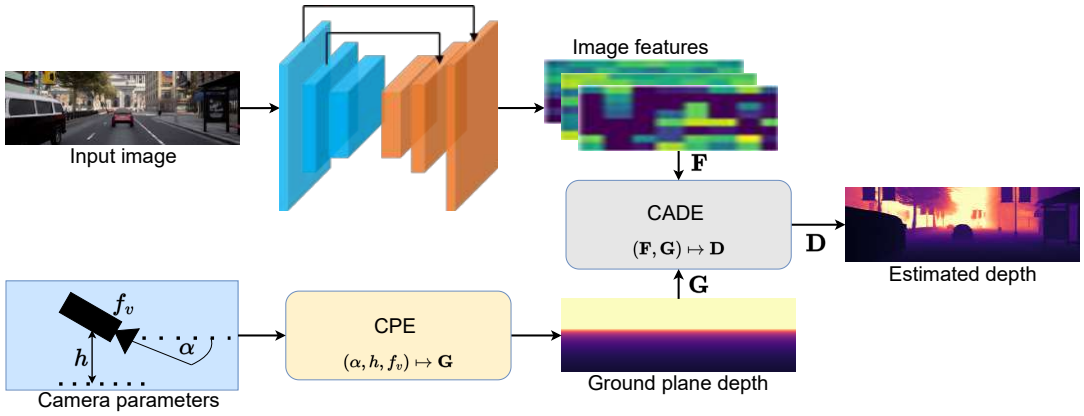


Fig. 1: Illustration of the proposed architecture. Our system embeds camera parameters as depth of the ground plane and learns depth that generalizes for various camera-car setups. CPE refers to the Camera Parameters Embedding described in Section II, while CADE refers to the Camera Adaptive Depth Estimation described in Section III.

for autonomous driving scenarios

- a network architecture with embedded parametrization as visualized in Fig. 1, specifically designed for generalization and further work in sim2real domain adaptation
- a large-scale annotated autonomous driving dataset within the CARLA simulator [30], created due to the unavailability of data with sufficiently diverse camera parameters[2]
- thorough experimentation on the effects of parameter variation and efficacy of the proposed approach.

II. Proposed Camera Parameters Embedding

Autonomous driving datasets such as the KITTI [31], Oxford RobotCar [32] or Cityscapes [33] frequently feature a single vehicle-camera setup. This means that correct depth values for certain pixels are almost identical across different images, e.g., on the ground plane. Even though convolution is an inherently positionally equivariant operation, convolutional neural networks tend to implicitly learn absolute position information from commonly used padding operations [34]. Additionaly, MDE networks have been shown to use the ground-plane contact point for object depth estimation [1].

In order for MDE to be practically used in automotive scenarios, depth estimation should be accurate for different vehicle-camera setups. However, if camera parameters during inference differ from the parameters used in training, depth estimation accuracy degrades significantly. For example, networks learn from the training data that the pixel at a particular position in the image tends to have certain depth value, which remains largely the same throughout the dataset. However, if the camera parameters are changed, this assumption breaks. Fig. 2 demonstrates the effects that camera parameters variation has on depth estimation accuracy. Changes in camera pitch, camera height, and vertical field of view (which also changes vertical focal length) during inference, compared to the training setup, significantly affect depth estimation accuracy, especially on the ground plane. On other hand, horizontal field of view and focal length changes do not significantly influence the estimation, as long as the context does not change dramatically.

In this work, we target the most plausible variations in vehicle-camera setups which can disturb depth estimation: camera height, camera pitch and vertical focal length. In order to learn metrically accurate depth with varying focal lengths, knowledge of the focal length should be embedded in the network due to inherent ambiguity between the focal length and depth [23], [24]. Additionally, while the network could learn the effects of camera height and pitch on the estimated depth, if trained with sufficiently diverse data, embedding of extrinsic parameters was shown to be beneficial [25]. Given that, we choose to embed all the three parameters in the network. To do so, for every pixel coordinate (u, v) we calculate the depth $\mathbf{G}(u, v)$ at which the optical ray intersects the ground plane via the following constraints

$$\begin{aligned} \mathbf{n}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}(\alpha)\mathbf{p} + h = \mathbf{0}, \\ \mathbf{p} = \mathbf{G}(u, v) \left[\begin{array}{ccc} \frac{u-c_u}{f_u} & \frac{v-c_v}{f_v} & 1 \end{array} \right]^T, \end{aligned} \tag{1}$$

where h is the camera height, $\mathbf{R}(\alpha)$ is the rotation matrix for camera pitch α, $\mathbf{n}$ is the ground plane normal, and $(f_u, f_v), (c_u, c_v)$ represent the camera focal length and principal point, respectively. With the assumption of ideal ground normal $\mathbf{n} = [0, -1, 0]^T$, depth $\mathbf{G}(u, v)$ can be calculated as a function of (α, h, f_v, c_v). In this work, we set the principal point at the center of the image plane, which is a fair assumption for most cameras. Our embedding function is thus a mapping

$$(\alpha, h, f_v) \mapsto \mathbf{G} \in [0, M]^{H \times W}, \tag{2}$$

where M is maximum depth specific to the dataset, and (H, W) are dimensions of the image. In Fig. 3 we show the visualization of our camera parameter embedding $\mathbf{G}$ for

[2]Dataset is publicly available at https://zenodo.org/record/7899804#.ZFT0oJFBzJV

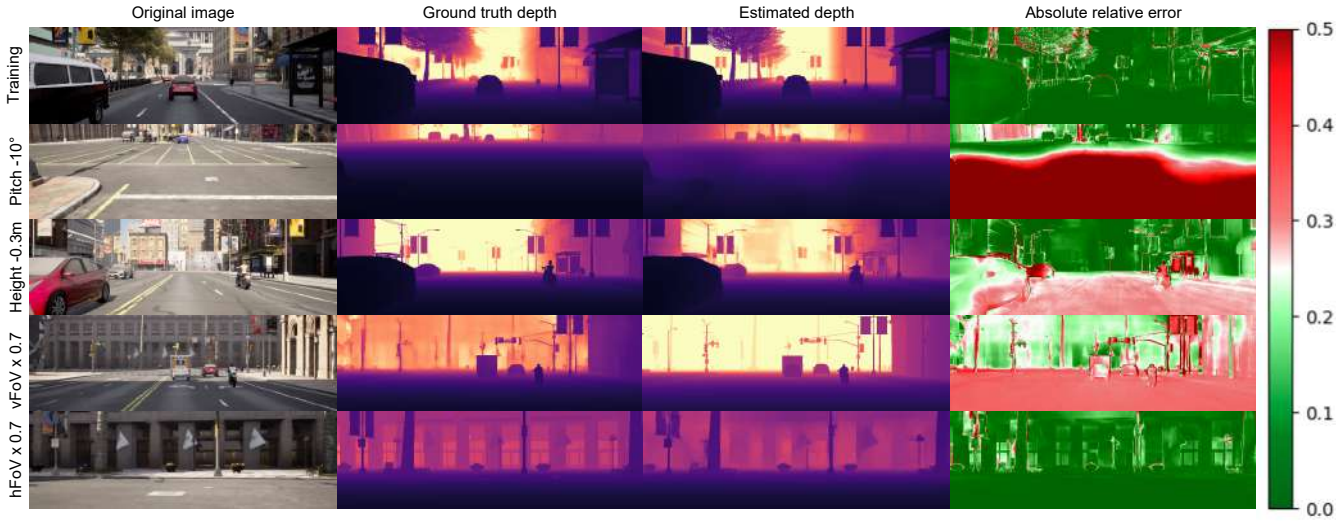
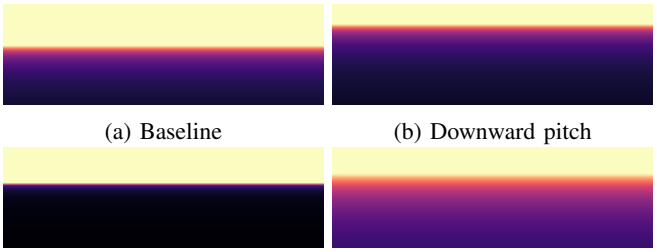


Fig. 2: Depth estimation results given common variations of the camera parameters compared to the training setup.



(a) Baseline (b) Downward pitch

(c) Smaller height (d) Larger focal length

Fig. 3: Visualization of the embedded camera parameters as ground plane depth $\mathbf{G}$.

different camera parameters. Variations of camera parameters (namely camera pitch, camera height and focal length) compared to the baseline are reflected in the embeddings, which provide the network with useful a-priori available information about the camera setup.

Our motivation for such a choice of camera parameters embedding is threefold:

- depth of the ground plane is a common and unique parametrization for camera pitch, camera height, and focal length, i.e., the mapping in (2) is injective
- embedding of $\mathbf{G}$ gives the neural network useful positional information, i.e., the network is explicitly informed about the expected depth for current camera parameters at a certain pixel position, if the ground plane is not occluded
- neural network can be forced to estimate depth as a function of $\mathbf{G}$, which leads to learning more robust features and better generalization accuracy for unseen camera parameters.

III. Proposed Network Architecture

Depth estimation networks often follow a standard encoder-decoder architecture with residual connections between encoder and decoder layers. Encoder learns spatially coarse features of higher dimensions, which are then continually upsampled towards original image resolution in the decoder. Our method is designed to work with arbitrary encoder-decoder architecture. While recent works use transformers for feature extraction and fusion [15]–[17], we choose to use a ResNet18 [35] encoder and a decoder combining convolutional and upsampling layers, thus recovering the feature map $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$. Instead of directly regressing the depth map $\mathbf{D}$ from $\mathbf{F}$, we forward it along with the map of embedded camera parameters $\mathbf{G}$ into the Camera Adaptive Depth Estimation (CADE) module.

A. CADE module

CADE transforms image features and camera parameters embedded as ground plane depth into the depth map, i.e., it performs the mapping $(\mathbf{F}, \mathbf{G}) \mapsto \mathbf{D} \in [0, M]^{H\times W}$. We fuse $\mathbf{G}$ and $\mathbf{F}$ inside a novel transformer architecture visualized in Fig. 4, as we want to exploit the global receptive field of the attention mechanism.

Firstly, we rearrange $\mathbf{F}$ and $\mathbf{G}$ into

$$\mathbf{Z'_F} = \begin{bmatrix} z'_{f_1} \\ \vdots \\ z'_{f_N} \end{bmatrix} \in \mathbb{R}^{N\times D'_f}, \mathbf{Z'_G} = \begin{bmatrix} z'_{g_1} \\ \vdots \\ z'_{g_N} \end{bmatrix} \in [0, M]^{N\times D'_g}, \tag{3}$$

where $N = \frac{HW}{p^2}, D'_f = Cp^2, D'_g = p^2$, with p being patch size. After layer normalization, these are then processed through a linear layer with addition of learnable positional embedding, resulting in sets of image feature tokens $\mathbf{Z_F} \in \mathbb{R}^{N\times D_f}$ and ground plane depth tokens $\mathbf{Z_G} \in \mathbb{R}^{N\times D_g}$:

$$\mathbf{Z_F} = \mathbf{Z'}_\mathbf{F}\mathbf{W_F} + \mathbf{p_F}, \mathbf{W_F} \in \mathbb{R}^{D'_f\times D_f}, \tag{4}$$

$$\mathbf{Z_G} = \mathbf{Z'}_\mathbf{G}\mathbf{W_G} + \mathbf{p_G}, \mathbf{W_G} \in \mathbb{R}^{D'_g\times D_g}. \tag{5}$$

Afterwards, we proceed with calculation of query, key and

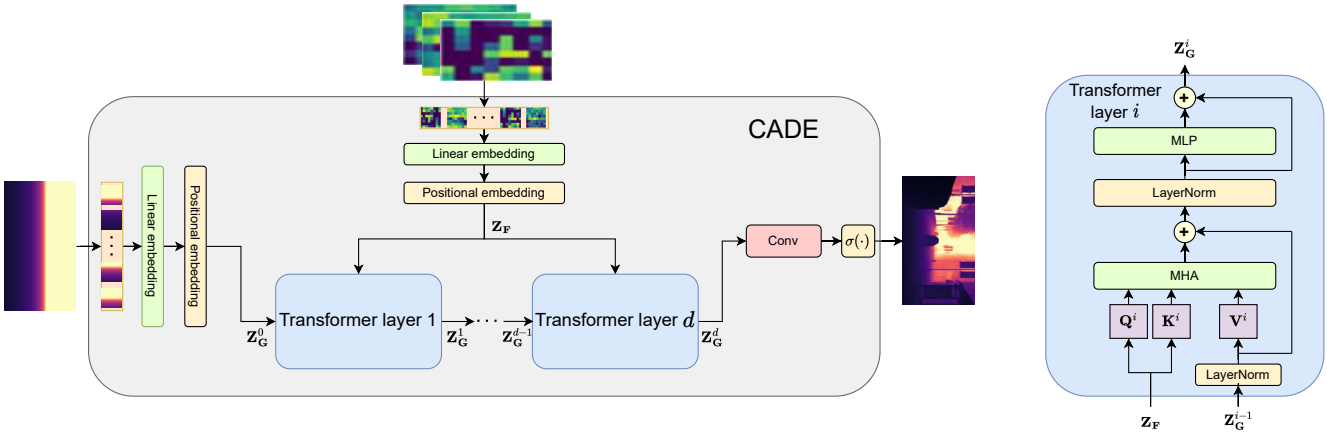


Fig. 4: Structure of the CADE module. Ground plane depths $\mathbf{Z_G}$ are processed through d successive transformer layers, with image features $\mathbf{Z_F}$ used in the calculation of attention weights.

value matrices needed for attention calculation:

$$\mathbf{Q} = \mathbf{Z_F}\mathbf{W_Q} \tag{6}$$
$$\mathbf{K} = \mathbf{Z_F}\mathbf{W_K} \tag{7}$$
$$\mathbf{V} = \mathbf{Z_G}\mathbf{W_V} \tag{8}$$

where $\mathbf{W_Q}, \mathbf{W_K} \in \mathbb{R}^{D_f \times D_h}$ and $\mathbf{W_V} \in \mathbb{R}^{D_g \times D_h}$ are projection matrices. Attended output is determined as

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{N}}\right)\mathbf{V}, \tag{9}$$

which is calculated for multiple heads and then fused via the linear layer. Notice how we calculate queries and keys from features tokens and values from ground plane depth tokens. This means that our attended output for particular token is a weighted function of ground plane depth tokens, with weights calculated as a self-attention of feature tokens.

We propagate tokens $\mathbf{Z_G}$ through d successive transformer layers consisting of multihead attention (MHA) and multilayer perceptron layers (MLP), along with residual connections where $\mathbf{Z_G^1}$ is initialized via (5):

$$\mathbf{Z_G^i} = \text{LayerNorm}(\mathbf{Z_G^i}), \tag{10}$$
$$\mathbf{Q}^i = \mathbf{Z_F}\mathbf{W_Q^i}, \mathbf{K}^i = \mathbf{Z_F}\mathbf{W_K^i}, \mathbf{V}^i = \mathbf{Z_G^i}\mathbf{W_V^i}, \tag{11}$$
$$\mathbf{Z_G^i} = \text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) + \mathbf{Z_G^i}, \tag{12}$$
$$\mathbf{Z_G^i} = \text{LayerNorm}(\mathbf{Z_G^i}), \tag{13}$$
$$\mathbf{Z_G^{i+1}} = \text{MLP}(\mathbf{Z_G^i}) + \mathbf{Z_G^i}. \tag{14}$$

Notice how through all CADE layers we use feature tokens $\mathbf{Z_F}$ only in calculation of attention weights. Depth estimation is thus forced to be a function of the embedded camera parameters $\mathbf{G}$, with $\mathbf{F}$ serving as a clue on how to properly combine embedding $\mathbf{G}$ into $\mathbf{D}$. CADE module is purposefully appended at the end of the network, which makes $\mathbf{F}$ independent of $\mathbf{G}$. In such a manner, network is incentivized to learn $\mathbf{F}$ that are invariant to different camera parameters.

Finally, we use Rearrange$(\cdot)$: $\mathbb{R}^{N \times D_g} \rightarrow \mathbb{R}^{C' \times H \times W}$ and a final convolutional layer with a sigmoid activation to regress depth map $\mathbf{D} \in [0, M]^{H \times W}$:

$$\mathbf{D} = \sigma(\text{Conv}(\text{Rearrange}(\mathbf{Z_G^d}))) * M. \tag{15}$$

For the training loss, we follow [15] and use Scale-Invariant loss (SI). With the logarithmic distance $g_i = log(\hat{d}_i) - log(d_i)$ between ground truth depth $\hat{d}_i$ and estimated depth d_i at pixel location i, SI loss is:

$$\mathcal{L} = \alpha\sqrt{\frac{1}{|\mathbf{D}|}\sum_i g_i^2 - \frac{\lambda}{|\mathbf{D}|^2}\left(\sum_i g_i\right)^2}, \tag{16}$$

where we use $\lambda = 0.85$ and $\alpha = 10$ as in [15].

Dataset	Size	Description
$\mathcal{B}$	20000	$\alpha = -5, h = 1.5, f_v = 570$
$\mathcal{U}_\alpha$	10000	$\alpha \sim \mathcal{U}(-15, 5), h = 1.5, f_v = 570$
$\mathcal{U}_h$	10000	$h \sim \mathcal{U}(1, 2), \alpha = -5, f_v = 570$
$\mathcal{U}_{f_v}$	10000	$f_v \sim \mathcal{U}(260, 880), \alpha = -5, h = 1.5$
$\mathcal{U}_{\alpha,h,f_v}$	40000	$(\alpha, h, f_v) \sim \mathcal{U}(-15, 5) \times \mathcal{U}(1, 2) \times \mathcal{U}(260, 880)$
$\mathcal{D}_{\alpha,h,f_v}$	40000	$\alpha \in \{-15, 5, 5\}, h \in \{1, 1.5, 2\}$, $f_v \in \{260, 570, 880\}$

TABLE I: Camera parameter specifications used in collected datasets. $\mathcal{B}$ – baseline dataset, parameters are constant throughout the dataset, $\mathcal{U}_\alpha, \mathcal{U}_h, \mathcal{U}_{f_v}$ – single varying parameter sampled from continuous uniform distribution, $\mathcal{U}_{\alpha,h,f_v}$ – all varying parameters sampled from continuous uniform distribution, $\mathcal{D}_{\alpha,h,f_v}$ – all varying parameters sampled from discrete uniform distribution of 3 possible values. Values for α, h, f_v are expressed in degrees, meters and pixels respectively.

IV. Experimental Results

A. Datasets

Due to unavailability of autonomous driving data with sufficiently diverse camera parameters, we created our own dataset using the CARLA simulator [30]. We simulate autonomous driving scenarios within urban, rural, and highway environments across 8 different maps *Town01* - *Town07* and

	Method	Training	Testing	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
a)	Baseline	$\mathcal{B}$	$\mathcal{B}$	0.046	0.482	**4.541**	**0.108**	0.959	**0.987**	**0.995**
	CPE + CADE	$\mathcal{B}$	$\mathcal{B}$	**0.044**	**0.475**	4.552	0.110	**0.961**	0.986	0.995
b)	Baseline	$\mathcal{B}$	$\mathcal{U}_{\alpha,h,f_v}$	0.261	2.076	7.478	0.297	0.547	0.846	0.960
	Baseline	$\mathcal{U}_{\alpha,h,f_v}$	$\mathcal{U}_{\alpha,h,f_v}$	0.064	0.437	3.549	0.102	0.960	0.991	0.996
	CPE + CADE	$\mathcal{U}_{\alpha,h,f_v}$	$\mathcal{U}_{\alpha,h,f_v}$	**0.039**	**0.387**	**3.382**	**0.085**	**0.970**	**0.991**	**0.997**
c)	Baseline	$\mathcal{B}$	$\mathcal{U}_{\alpha}$	0.244	1.248	5.748	0.198	0.699	0.931	0.989
	Baseline	$\mathcal{U}_{\alpha}$	$\mathcal{U}_{\alpha}$	0.041	0.310	3.445	0.078	0.973	0.991	**0.997**
	CPE + CADE	$\mathcal{U}_{\alpha}$	$\mathcal{U}_{\alpha}$	**0.035**	**0.301**	**3.410**	**0.076**	**0.975**	**0.992**	0.997
d)	Baseline	$\mathcal{B}$	$\mathcal{U}_{h}$	0.214	1.245	6.035	0.232	0.666	0.924	0.988
	Baseline	$\mathcal{U}_{h}$	$\mathcal{U}_{h}$	0.038	0.319	3.600	0.085	0.971	0.991	0.997
	CPE + CADE	$\mathcal{U}_{h}$	$\mathcal{U}_{h}$	**0.035**	**0.312**	**3.581**	**0.083**	**0.972**	**0.992**	**0.997**
e)	Baseline	$\mathcal{B}$	$\mathcal{U}_{f_v}$	0.254	1.746	7.286	0.268	0.563	0.868	0.987
	Baseline	$\mathcal{U}_{f_v}$	$\mathcal{U}_{f_v}$	0.040	**0.351**	3.722	0.088	0.972	0.990	**0.997**
	CPE + CADE	$\mathcal{U}_{f_v}$	$\mathcal{U}_{f_v}$	**0.035**	0.353	**3.647**	**0.084**	**0.972**	**0.991**	0.997
f)	Baseline	$\mathcal{D}_{\alpha,h,f_v}$	$\mathcal{U}_{\alpha,h,f_v}$	0.102	0.633	4.655	0.148	0.889	0.987	0.996
	CPE + CADE	$\mathcal{D}_{\alpha,h,f_v}$	$\mathcal{U}_{\alpha,h,f_v}$	**0.067**	**0.458**	**3.920**	**0.110**	**0.945**	**0.991**	**0.997**

TABLE II: Results of various model and dataset configurations. Baseline refers to the standard encoder-decoder architecture, with ResNet 18 encoder and decoder from [20], where depth $\mathbf{D}$ is directly regressed from image features $\mathbf{F}$, while CPE + CADE refers to addition of our contributions. Results are expressed in standard MDE metrics [20], red – lower is better, blue – higher is better.

Town10HD that include highly detailed and realistic textures. The maps are populated with a diverse set of traffic actors, which are then autonomously controlled while respecting the traffic rules.

In order to capture the training and testing data, we mount RGB and depth cameras in a way that no part of the car is within the field of view of the camera. Advanced RGB camera parameters such as distortions and postprocessing effects are adjusted to mimic the KITTI dataset [31] as close as possible. Camera sensors are repeatedly destroyed and reinitialized with new extrinsic and intrinsic parameters, thus avoiding the memory difficulties which are present with multiple camera sensors working at the same time. We collect several datasets with different distributions of camera parameters, as described in Table I.

B. Implementation details

As our method is adaptable for various encoder-decoder architectures, we use a simple convolutional residual network. Our encoder is a ResNet 18 network which encodes image features at a $\frac{H}{32} \times \frac{W}{32}$ resolution. Decoder then successively upsamples the features in 5 stages, each consisting of a 3x3 kernel convolution which fuses encoder features via skip connection and an upsampling layer followed by another convolutional layer. Finally, decoder outputs image features $\mathbf{F} \in \mathbb{R}^{C \times H \times W}$, where we use $C = 16, H = 320, W = 1024$.

For our CADE module, we choose to use a light architecture in order to prevent a significant increase in computation time and memory consumption. We use a standard patch size $p = 16$, with inner embedding dimensions $D_g = 1024$ and $D_f = 4096$. We calculate the MHA with 8 heads and a head dimension $D_h = 64$, which is then followed by a MLP with one hidden layer which increases the embedded dimension by two times. In order to keep our CADE module lightweight, we choose $d = 2$ for a number of transformer layers. Finally, following the standard practice in depth estimation [20], we estimate depth up to a maximum value $M = 80$m.

We train our network with an Adam optimizer [36] with a batch size 12. We decrease the learning rate linearly from 4×10^{-5} to 4×10^{-6}. All networks are trained and tested on a single Nvidia RTX A5000 GPU.

C. Results

We conduct a thorough experimentation on the effect of camera parameter variation and efficacy of our approach. In Table II we present results for various combinations of methods and dataset configurations. In order to test the generalization of each approach, for each dataset we create a 90%/10% training and testing split.

First of all, in Table II a) we perform the ablative experiments on the baseline dataset, where we both train and test the networks on the data collected with a single vehicle-camera setup. As expected, despite the increase of the model complexity due to the addition of our contributions, usage of CPE and CADE does not improve performance compared to the standard encoder-decoder architecture, since ground plane depth $\mathbf{G}$ fused in CADE does not supplement the network with useful information. This is a desired behavior, considering that the camera parameters are constant throughout the dataset. Ground plane depth is mostly the same across all images, thus enabling the baseline model to easily learn the information which is otherwise supplemented with $\mathbf{G}$ in our approach.

Afterwards in Table II b) we test the performance on the dataset with varying camera parameters $\mathcal{U}_{\alpha,h,f_v}$. Naturally, baseline method trained on a dataset with a constant vehicle-camera setup performs poorly since it is biased to a particular perspective geometry induced in the training data. On the contrary, baseline method trained on $\mathcal{U}_{\alpha,h,f_v}$ performs surprisingly well, with good generalization performance on unseen images. This shows that, when presented with sufficiently diverse perspective geometry, network can exploit

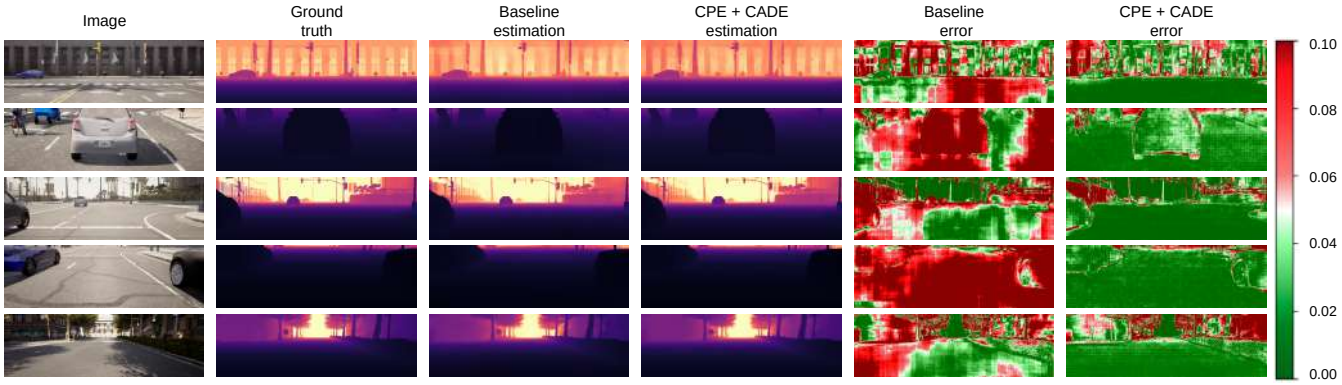


Fig. 5: Results of MDE trained and tested on the $\mathcal{U}_{\alpha,h,f_v}$ dataset with visualization of absolute relative error. Fusion of embedded camera parameters within CADE significantly reduces the absolute relative error compared to the baseline, especially on the ground plane.

Method	Abs Rel	Sq Rel	RMSE	RMSE log
Early fusion	0.051	0.401	3.401	0.090
Mid fusion	0.050	0.397	3.402	0.089
Late fusion	0.061	0.412	3.622	0.102
CADE	**0.039**	**0.387**	**3.382**	**0.085**

TABLE III: Results of the ablation experiments trained and tested on $\mathcal{U}_{\alpha,h,f_v}$, with fusion of $\mathbf{G}$ into convolutional channels at a certain point. Early fusion – fusion in the first encoder layer, Mid fusion – fusion in skip connections between encoder and decoder, Late fusion – fusion in last decoder layer.

semantic cues to infer depth for varying camera parameters. However, the accuracy of the baseline network is significantly lower compared to the proposed approach. Fusion of embedded camera parameters within the CADE module notably improves the results for all MDE metrics, proving the usefulness of information encoded in ground plane depth $\mathbf{G}$, and efficacy of its fusion within the CADE module. Figure 5 shows significant reduction in absolute relative error compared to the baseline, especially for inconsistent estimations on the ground plane.

In order to examine the generalization capability for each camera parameter separately, in Table II c) d) e) we repeat the same experiment while selectively varying only one camera parameter throughout the dataset. Again, while the model trained on a single vehicle-camera setup performs poorly, baseline network can learn to generalize when presented with diverse data in the training set. However, in contrast to results in Table II b), fusion of camera parameters in CADE module does not significantly improve the results. Since only one parameter is varied, network can learn to focus on semantic cues specific to that camera parameters, thus effectively reducing the need for embedding of $\mathbf{G}$.

In Table II f) we examine the ability of our approach to generalize for camera parameters not present in the training distribution. To do so, for training we use a sparse discrete distribution with 3 possible samples for each parameter, positioned at the tail ends and the mean of the continuous uniform distribution $\mathcal{U}_{\alpha,h,f_v}$. In such manner, network should learn to meaningfully predict the depth for camera parameters between those discrete samples. We show that our approach learns to generalize more effectively than the baseline, which means that the network successfully learns geometric relationship between embedded camera parameters $\mathbf{G}$ and scene depth. To that end, our approach is feasible to be utilized with real-world data, since the collection of data with distribution similar to $\mathcal{D}_{\alpha,h,f_v}$ is feasible. However, increase in accuracy is not as prominent as in Table II b), which means that the semantic cues, such as the horizon level, when varied throughout the training dataset provide useful information for training of camera invariant depth estimation, even when the network is supplemented with embedded camera parameters $\mathbf{G}$.

Finally, we assess the performance of various fusion methods for embedded camera parameters $\mathbf{G}$ in Table III. Early and mid fusion are similar to [25] and [24] respectively, but with different choice of embedded camera parameters and embedding function. The most meaningful result is the difference between performance of late fusion within convolutional layers and our CADE module. Even though the fusion happens at the same point in the network, CADE achieves better results by taking advantage of the global receptive field of self-attention, and by strict enforcement of estimating depth $\mathbf{D}$ as a function of $\mathbf{G}$.

V. Conclusion and Further Work

In this paper we have presented an approach for camera invariant monocular depth estimation for automotive scenarios. After detailed examination on the effects of varying camera parameters on depth estimation performance, we designed a novel camera parameters embedding procedure in order to supplement the network with useful information about the perspective geometry and to force the network to learn depth estimation as a function of embedded parameters, thus effectively enabling learning of camera invariant features. The proposed embedding is fused with image features within

a novel module that exploits the global receptive field of the self-attention. We assess the accuracy of the proposed approach on various datasets with different camera extrinsic and intrinsic parameters distributions, collected within a simulated automotive environment. We show that the embedding provides useful information about the perspective geometry and enables better generalization on unseen data. Our method is specifically designed for further work in domain adaptation, where we aim to achieve camera invariant depth estimation on real-world data.

REFERENCES

[1] T. v. Dijk and G. d. Croon, "How do neural networks see depth in single images?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2183–2191.

[2] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *Advances in neural information processing systems*, vol. 27, 2014.

[3] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 239–248.

[4] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[5] A. CS Kumar, S. M. Bhandarkar, and M. Prasad, "Depthnet: A recurrent neural network architecture for monocular depth prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 283–291.

[6] M. Mancini, G. Costante, P. Valigi, T. A. Ciarfuglia, J. Delmerico, and D. Scaramuzza, "Toward domain independence for learning-based monocular depth estimation," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1778–1785, 2017.

[7] D. Xu, W. Wang, H. Tang, H. Liu, N. Sebe, and E. Ricci, "Structured attention guided convolutional neural fields for monocular depth estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3917–3925.

[8] B. Li, C. Shen, Y. Dai, A. Van Den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1119–1127.

[9] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5162–5170.

[10] Y. Cao, Z. Wu, and C. Shen, "Estimating depth from monocular images as classification using deep fully convolutional residual networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3174–3182, 2017.

[11] H. Jung, Y. Kim, D. Min, C. Oh, and K. Sohn, "Depth prediction from a single image with conditional adversarial networks," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 1717–1721.

[12] K. G. Lore, K. Reddy, M. Giering, and E. A. Bernal, "Generative adversarial networks for depth map estimation from rgb video," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2018, pp. 1258–12 588.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[15] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4009–4018.

[16] A. Agarwal and C. Arora, "Attention attention everywhere: Monocular depth prediction with skip attention," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 5861–5870.

[17] Z. Li, Z. Chen, X. Liu, and J. Jiang, "Depthformer: Exploiting long-range correlation and local information for accurate monocular depth estimation," *arXiv preprint arXiv:2203.14211*, 2022.

[18] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 340–349.

[19] R. Li, S. Wang, Z. Long, and D. Gu, "Undeepvo: Monocular visual odometry through unsupervised deep learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 7286–7291.

[20] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3828–3838.

[21] R. Li, S. Wang, Z. Long, and D. Gu, "Undeepvo: Monocular visual odometry through unsupervised deep learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7286–7291.

[22] K. Koledić, I. Cvišić, I. Marković, and I. Petrović, "Moft: Monocular odometry based on deep depth and careful feature selection and tracking," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 6175–6181.

[23] L. He, G. Wang, and Z. Hu, "Learning depth from single images with deep neural network embedding focal length," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4676–4689, 2018.

[24] J. M. Facil, B. Ummenhofer, H. Zhou, L. Montesano, T. Brox, and J. Civera, "Cam-convs: Camera-aware multi-scale convolutions for single-view depth," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 826–11 835.

[25] Y. Zhao, S. Kong, and C. Fowlkes, "Camera pose matters: Improving depth prediction by mitigating pose distribution bias," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 759–15 768.

[26] S. Saha, A. Obukhov, D. P. Paudel, M. Kanakis, Y. Chen, S. Georgoulis, and L. Van Gool, "Learning to relate depth and semantics for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8197–8207.

[27] A. Atapour-Abarghouei and T. P. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2800–2810.

[28] S. Zhao, H. Fu, M. Gong, and D. Tao, "Geometry-aware symmetric domain adaptation for monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9788–9798.

[29] K. PNVR, H. Zhou, and D. Jacobs, "Sharingan: Combining synthetic and real data for unsupervised geometry estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 974–13 983.

[30] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[31] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[32] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[33] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[34] M. A. Islam, S. Jia, and N. D. Bruce, "How much position information do convolutional neural networks encode?" *arXiv preprint arXiv:2001.08248*, 2020.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Synthetic Data-based Detection of Zebras in Drone Imagery

Elia Bonetto[*,†] *Student Member, IEEE,* and Aamir Ahmad[†,*] *Senior Member, IEEE*

***Abstract*— Nowadays, there is a wide availability of datasets that enable the training of common object detectors or human detectors. These come in the form of labelled real-world images and require either a significant amount of human effort, with a high probability of errors such as missing labels, or very constrained scenarios, e.g. VICON systems. On the other hand, uncommon scenarios, like aerial views, animals, like wild zebras, or difficult-to-obtain information, such as human shapes, are hardly available. To overcome this, synthetic data generation with realistic rendering technologies has recently gained traction and advanced research areas such as target tracking and human pose estimation. However, subjects such as wild animals are still usually not well represented in such datasets. In this work, we first show that a pre-trained YOLO detector can not identify zebras in real images recorded from aerial viewpoints. To solve this, we present an approach for training an animal detector using only synthetic data. We start by generating a novel synthetic zebra dataset using GRADE, a state-of-the-art framework for data generation. The dataset includes RGB, depth, skeletal joint locations, pose, shape and instance segmentations for each subject. We use this to train a YOLO detector from scratch. Through extensive evaluations of our model with real-world data from i) limited datasets available on the internet and ii) a new one collected and manually labelled by us, we show that we can detect zebras by using only synthetic data during training. The code, results, trained models, and both the generated and training data are provided as open-source at https://eliabntt.github.io/grade-rr.**

I. INTRODUCTION

A large dataset that includes realism and diversity in features is a fundamental building block for obtaining any working and reliable deep-learning model. This is especially true when dealing with visual tasks such as detection, semantic segmentation and shape estimation. For these, variability in both visual appearances and environmental conditions as well as a high number of instances are required. A variety of datasets have been introduced during the last decades to address various image-based tasks like MNIST [1], COCO [2], and PASCAL-VOC [3]. These have historically been based on real-world data, be this either images or videos, manually labelled by humans. Apart from being time-consuming and costly, this introduces errors such as missing and wrong labels [4]. Examples of these can be visualized in Fig. 2 and Fig. 3. Furthermore, ground truth for specific tasks may not be available either because it is hard to obtain, e.g., shape or skeletal information, or because it requires costly manual labelling procedures. These limitations impede the usage of these datasets in problems such as aerial human pose estimation [5], animal pose estimation [6], [7], or aerial wild-animal detection. For these reasons, methodologies to generate synthetic data became more ubiquitous since the advent of rendering engines such as Unity, Blender, Unreal Engine and IsaacSim. These are advantageous in multiple aspects since they allow generation and automatic labelling of ground truth data through full customization possibilities and with minimum human effort [8]. Indeed, synthetic data has been used in a variety of tasks such as human detection [8], [9], pose and shape estimation of humans [5], and semantic segmentation [10]. However, they usually lack the visual realism necessary to generalize well to real-world data if used alone. Thus, a combination of real and synthetic data is often utilized [8], [9]. Moreover, these datasets are usually application-specific and hardly generalize to different scenarios, tasks, or data. For example, wild animals are widely under-represented in datasets such as COCO or PASCAL-VOC [11], [12], [13]. Indeed, apart from a limited number of labelled images and videos of uncommon animal species such as zebras, hippopotami, and giraffes, there is also a general lack of variety of scenarios in which those are recorded. Taking zebras as an example, there are only 1916 training and 85 validation images containing at least one instance of them in the COCO dataset. To solve this problem, we generate a new synthetic dataset using our GRADE [8] framework, a publicly available animated zebra model and environments from the Unreal Engine

Fig. 1: An example image of our synthetically generated zebras in a Savanna environment.

[*]Max Planck Institute for Intelligent Systems, Tübingen, Germany. firstname.lastname@tuebingen.mpg.de

[†]Institute of Flight Mechanics and Controls, University of Stuttgart, Germany. firstname.lastname@ifr.uni-stuttgart.de

The authors thank Eric Price, Nitin Saini, Egor Iuganov, Chenghao Xu, and Benedikt Schwämmle for their help in collecting and processing the data.

We would like to extend our deepest gratitude to the Wilhelma Zoo in Stuttgart, and in particular to Ms. Ulrike Rademacher, for permitting us to record videos of Grévy's zebras on the premises of the Wilhelma Zoo.

The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Elia Bonetto.

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

marketplace. An example of the generated data can be seen in Fig. 1 and Fig. 4. We use this data to train a YOLO-based detector and perform evaluations with an extensive dataset of zebras captured by drones consisting of 104K images and the APT-36K [14] dataset. With this, we show that training with our synthetic data outperforms the baseline models trained on real-world datasets. In this paper, we take zebras as an example as it is an endangered species, which is greatly under-represented in currently available datasets. However, the proposed method can be generalized to other animals as well.

The rest of the paper is structured as follows. In Sec. II we review the current state-of-the-art in simulated worlds and animal datasets. Our method is described thoroughly in Sec. III. In there, we give a general overview of the system and explain how we generated our data. We then present the results of our experiments in Sec. IV and conclude the work with Sec. V with comments on known limitations and possible future work.

II. Related Work

In this section, we focus on two areas: animal-based datasets, and simulation engines.

Animals. There are not many animals-based datasets available in the literature [15], [12], [11], especially considering full 3D-vertices information and precise segmentation. This is clearly related to the difficulties of collecting and labelling ground truth data in outdoor scenarios. Various approaches have been applied to overcome this problem, ranging from using toy models [6], [7], merging different datasets [12], or using synthetic data [15], [16]. However, all of them fall short in some aspects like lack of animal species variability, size, pose and shape information, skeletal joints location, or limited capturing settings. For example, Horse-10 [17] has only horses moving left-to-right. The authors of the SMAL model [6] do not release the generated data. The Grévy's zebra dataset [18] consists only of 900 low-resolution images that do not contain either correct bounding boxes or labels for all animals. An example of that is provided in Fig. 2. AnimalPose [11] focuses on a limited set of animals, in which zebras are not included. The 4DComplete dataset [15], although it contains various animal animations, it fails on releasing textures or textured FBX files making it impossible to customize. They do provide rendered RGB+D images and scene flow but, still, the renderings are provided without any background information. Other synthetic datasets, such as the one from Mu et al. [19], contain data which cannot be used to train a successful detector since they are generated with unrealistic backgrounds and textures [19]. These also suffer from the low viewpoint variability and diversity of scenarios, such as the data from COCO. We must also note that, as shown in Fig. 3, COCO is not exempt from wrong or imprecise labelled data.

Simulation engines. Gazebo [20] is currently the standard for robotic simulation. High reliable physics and tight integration with ROS are its key advantages. However, the lack of visual realism and customization possibility, makes it unusable for generating visual data to be used in learning tasks. Indeed, alternatives emerged in the last years, such as [21], [22], [23], [24], [25], [26], along with several datasets [5], [9] that use Unreal Engine, Unity, and Blender for rendering. The combination of AirSim and Unreal Engine has been widely explored to generate multiple datasets focused on specific tasks such as human pose estimation [5] and visual odometry [27]. Simulators focused on robotics are usually limited by the type of the environment, e.g. indoor [24], [22], or the task, e.g. self-driving car [26]. Clearly, generalizing them to outdoor scenarios with animated animals is not trivial. GRADE [8] is a recently introduced method to generate synthetic data built directly upon Isaac Sim. It is a framework that includes both data generation and general robotic testing capabilities thanks to its integration with ROS and the use of ray- and path-tracing. In this work, we leverage the flexibility of GRADE to generate new synthetic data of outdoor scenarios with randomly placed zebras.

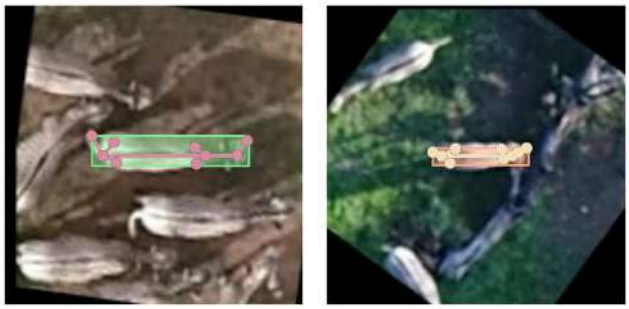


Fig. 2: Examples of missing bounding boxes and keypoints from the Grévy's zebra [18] dataset. Image ids 869 (left) and 882 (right).

(a) ID: 20164, missing bounding boxes **(b)** ID: 22149, toy labeled

(c) ID: 32206, wrong bounding box **(d)** ID: 533961, imprecise bounding box

Fig. 3: Four examples of wrongly labelled zebras from the COCO [2] dataset.

III. Approach

Using the system introduced in our previous work GRADE [8], we generate an outdoor-environment dataset

focused on zebras. GRADE is our synthetic data generation framework based on Isaac Sim. Thanks to the flexibility of GRADE, this approach will be easily applicable also to other animal species or setups. The details about the GRADE framework and the simulation management are thoroughly described in [8]. We proceed here highlighting any major difference with respect to the already introduced system.

A. Synthetic data

1) Environments: We selected nine commercial and one freely available environments from the Unreal Engine marketplace. We used the Unreal Engine Omniverse connector[1] to convert them to the USD file format. We list the environments with the corresponding shortened URL in Tab. I. For each environment, we used directly the available demos and pre-built scenarios. Then, we proceeded to remove the original sky sphere and fix textures when necessary. The connector indeed does not yet support full export of the terrains from Unreal Engine, resulting in a lower level of detail, e.g. missing 3D grass, some textures, and level of details. We replaced the textures with some taken from IsaacSim itself, resembling the *color* of the grass.

Environment Name	URL
Bliss	https://bit.ly/3HD3zYP
Forest	https://bit.ly/3mYQv8Z
Grasslands	https://bit.ly/3HD3zYP
Iceland	https://bit.ly/3Ax8zKi
L_Terrain	https://bit.ly/3V6H7MU
Meadow	https://bit.ly/3Hgxk1n
Moorlands	https://bit.ly/3oHT1ku
Rural Australia (Free)	https://bit.ly/3i5j6Hi
Windmills	https://bit.ly/3AvVTDK
Woodland	https://bit.ly/3mYQv8Z

TABLE I: Names and shortened URLs of the used environments.

2) Dynamic assets: We use a freely available zebra model from SketchFab [28]. This model consists of 34 different in-place animation sequences, i.e. without root translation or rotation movements, for a total of 888 animation frames. We converted each animation sequence to the USD format using Blender and its Omniverse connector. Then, we post-processed the sequence to obtain per-frame vertices position and skeletal information. This allows us, for every generated frame, to have corresponding ground truth information about these two characteristics. The vertices are used to compute oriented bounding boxes that are then employed for the placement procedure.

3) Placement of zebras: Zebra placement is based on an ad-hoc procedure that is repeated every time a frame is generated. For every environment we select a specific mesh as 'terrain', which represents the area in which we will then place the zebras. The placement consists then of four main steps: i) selecting a random rectangular area of the terrain, ii) randomly selecting a set of zebras, iii) for each zebra select frame of its animation sequence, a scaling factor and a global orientation of the zebra, iv) place the zebra in the rectangular area considering the bounding-box occupancy. The sides of the rectangle are randomly selected to be between 40 and 120 meters, while the scaling factor ranges between 40% and 100% and allows us to obtain a higher degree of variability. The placement is an iterative procedure that considers one zebra model after the other. Any model that cannot be placed following a detected collision is removed from the simulation. The final results depend mostly on the resolution of the terrain mesh for both collisions between meshes and contact of the zebras with the ground. In general, we noted that collisions are rare and that contacts with the ground are good. Note that, as opposed to [8], we do not consider the full animation sequence since we lack any root translation information.

4) Data collection methodologies: Contrary to what is done for indoor environments in [8], here we focus on image generation rather than video sequences. We also perform a series of randomization for each captured frame, i.e. the i) time of day, ii) number of zebras in the environment, iii) their scaling factor, iv) their specific animation frame, and the v) placement of three cameras that will record the scene. Specifically, given any environment, we set up three aerial cameras and randomly pre-load 250 zebras at the beginning of each experiment. We then uniformly select the number of zebras that will be placed in the next frame. This number is set to be between 2 and 250. Note that this is *not* the number that will appear in each frame, nor is the final number of zebras that are actually placed. As explained above, the placement strategy may remove some of the zebras and the camera may not observe all the zebras given a point of view. Once the placement happened, we randomize the location of the cameras and the time of days three times. The time of day will be 90% of the time between 5 am and 8 pm, which results in good lighting conditions given our current settings, and 10% of the time in the remaining hours, resulting in dusk-to-night light settings. This further randomizes the appearances of both the generated frames and the shadows. Cameras are placed using the average location of the zebras as a pivot point. For the placement of the cameras, we distinguish between two slightly different image-generation procedures: one more general and one more focused on capturing zebras from a nearer viewpoint. We first describe the former and then identify the minor modifications that we applied to the latter. From the pivot point, we randomize the distance in the x-y plane and the height of the camera. The height is set to be between 5 and 20 meters more than the average of the zebras, while the x and y are set to be between -/+ 100 meters. Once the position is fixed, we can compute and randomize roll, pitch, and yaw. Roll is set to be within $[-10,10]$ degrees, yaw is set to be the ray that connects the camera and the pivot point with an additional random $[-30,30]$ degrees. Pitch is computed as $\theta = atan2(pivot_z - cam_z, d(pivot, camera)) + 15$ degrees, where $d(pivot, camera)$ is the distance in the x, y plane. The modifications applied to this methodology during the second image-generation procedure are as follows: the x and y positions are set to be within 5 meters of the virtual

[1] https://bit.ly/3X82sph

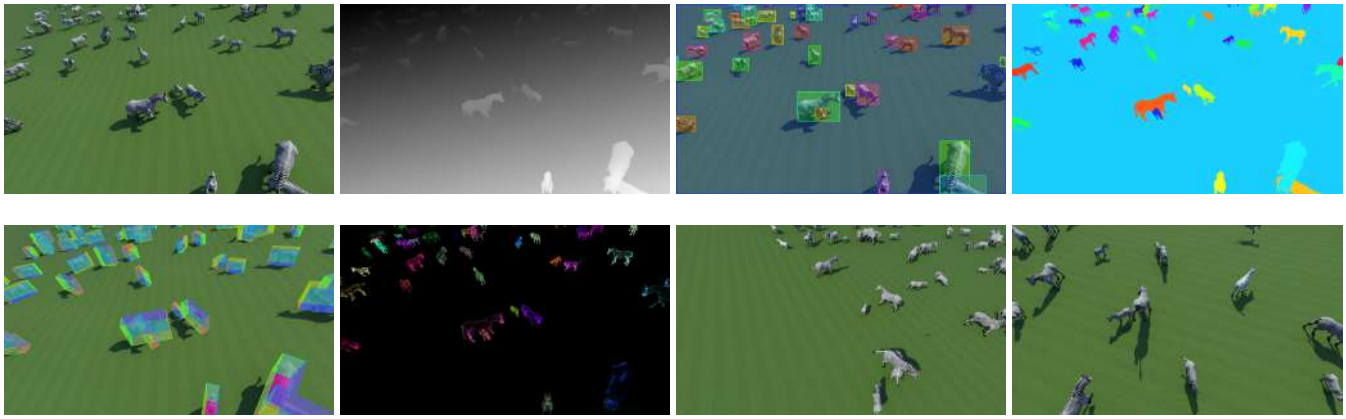


Fig. 4: An example of the generated data following the pipeline described in Sec. III. On the first row, from the left, we can see the rendered RGB image, depth data, 2D bounding boxes, and semantic instances. On the second row, from the left, we can see 3D-oriented bounding boxes, the vertices of each mesh drawed over a black background, and the second and third views of the same scene as taken from the other drones. Best viewed in color.

bounding box containing all zebras, the yaw has an additional $[-15, 15]$ degrees component instead of the $[-30, 30]$. This results in images that are closer to the zebras than the ones obtained from the former camera placement strategy.

For each environment, we randomly place the zebras 200 times, resulting in 600 frames per experiment per camera, i.e. 1.8K frames in total. After the generation process is complete for all ten environments, this totals to 18K frames captured with the first camera placement strategy, and 18K with the camera set to be more nearby, totalling 36K frames. For each frame, we save the pose of the cameras, zebra skeletal pose and meshes vertices, ground truth depth and instance segmentation.

B. Real-world data

We performed several data collection experiments in a controlled scenario within the Wilhlema Zoo in Stuttgart (DE). An example of the collected data can be found in Fig. 5. We used two manually flown DJI Mavic 2 Pro drones, recording images at 29.97 fps at a resolution of 3840×2160, and three GoPro Hero8, also with a resolution of 3840×2160 at 59.94 fps. None of the GoPros had fixed locations between experiments. The data has been manually synchronized by using a recorded light signal visible by all cameras at the same time. We then extracted one frame every five seconds from all the videos. Out of these, 905 images were randomly selected and annotated manually and precisely. These annotations were then used to train an SSD multibox [29] detector, which, with *Smarter-Labelme* [30], allowed us to obtain bounding boxes on our video sequences with ease. Out of all the data available, we finally selected three collections during which the zebras were visible by both drones. The boxes on those sequences were then manually refined in a final step. This procedure thus resulted in 905 precisely annotated images, and 104K frames annotated with [30]. Within this work, we release the data used during our training experiments, i.e. the 905 precisely annotated images and 200 automatically labelled ones from one of the experiments (see Sec. IV for details).

IV. EXPERIMENT AND EVALUATIONS

Here we seek to demonstrate that the synthetic data generated by our method can be used effectively for a vision-based task which is highly related to image features and context, such as the detection of zebras in outdoor wild environments from an aerial point of view. Our hypothesis is that, by training a model using only synthetic data acquired in a realistic simulation environment, we can achieve detection performance on real images comparable to a model trained on a manually and very-precisely labelled set of real images. Our goal is to prove that synthetically generated data *alone* can be used to train a network capable of detecting zebras with high accuracy in real-world images. To that end, we decided to perform various tests on YOLOv5s. We train the networks from scratch and with mixed datasets to test the performance and provide a complete overview. All the training runs are made from scratch with the default hyperparameters and for the standard 300 epochs. We do not introduce any additional data augmentation technique different from the one applied by default by the YOLOv5 code. This consists of some randomization in the scale, horizontal flip, translation and HSV colour space factors. We do not modify these values to have a fairer comparison across the models that would not require parameter grid searches or other steps when compared to the baseline pre-trained model. We save the best model, as evaluated on the specific validation set, and compare it over multiple datasets. We evaluate the performance with the COCO standard metric (mAP@[.5, .95], AP in this work) and the PASCAL VOC's metric (mAP@.5, AP50 in this work). We also report the average and weighted average of these two metrics. We weigh based on the cardinality of each one of the evaluated datasets. With these comparisons, we demonstrate that, with our synthetic data, we can successfully capture real-world features. This, while also obtaining trained models which show, in general, improved performance when compared to the pre-trained ones.

Synthetic Full dataset (SF) is the dataset containing all the 36K synthetically generated images. These are then

(a) R1 **(b)** R2

(c) R3 **(d)** RP

Fig. 5: An example of our real-world collected images used for testing. Three aerial and one 'ground-level' views. Best viewed in color.

randomly shuffled and split into 80/20 train/validation sets. Synthetic Closeby (SC) is the synthetic data generated only by the second strategy, as described in Sec. III-A.4, i.e. 18K images for which the camera is within 5 meters of the bounding box containing all the zebras. This data is also divided randomly with an 80/20 ratio. With COCO we refer to the images of the COCO dataset [2] which contains zebras, i.e. 1916 training and 85 validation examples. Due to the small size of the validation set of the COCO dataset, we do not perform any training on this data alone. With APT-36K we indicate the set of images from the APT-36K [14] dataset which contains zebras, i.e. 1.2K samples. We then have, R1, R2, and R3 which are three sets of real-world data which is not precisely labelled, as described in Sec. III-B. To distinguish between which drone captured the given sequence, we use the suffixes *_D1* and *_D2*. R1 consists of 19.7K images, R2 of 23.4K, and R3 of 8.8K, for each drone. Finally, we use RP to indicate the set of the 905 real-world images precisely labelled by us, sampled from representative images from the previous Rx datasets and additional images captured with the GoPros as described in Sec. III-B. Of them, 720 are randomly used in training and 185 for validation. An example of the bounding boxes of our real-world data is provided in Fig. 5. We also provide two zoomed-in examples of imprecise labels in Fig. 6. Note that also other datasets, e.g. COCO (see Fig. 3), present such approximations.

Our baseline for comparison consists of the network pre-trained on the full COCO dataset. We perform some training

Dataset	Description	Train imgs.	Val imgs.
SF	Our full synthetic data	29K	7K
SC	A subset of our synthetic data focused on camera poses closer to the zebras	14.5K	3.5K
R1	Aerial capture experiment 1	—	19.7K
R2	Aerial capture experiment 2	—	23.4K
R3	Aerial capture experiment 3	—	8.8K
$R3_{100}$	100×2 images randomly chosen from R3	100	100
RP	Precisely labelled images from R1/2/3 and GoPros	720	185
COCO	COCO-zebras	1916	85
Rx_D1,2	Either 1st or 2nd drone capturing during experiment Rx		
∗-1920	Same dataset but using 1920 image size during training		
∗ + ⋄	Trained by merging ∗ and ⋄ corresponding train and validation sets		

TABLE II: Zebras datasets legend and training/validation sizes

Fig. 6: Two zoomed-in examples of imprecisely labelled data. The bounding boxes can either be slightly too loose or too tight on the zebras.

tests on both the default 640×640 image size and the increased 1920×1920, identified in our table with the '-1920' suffix. Once established that the bigger image size yields better results, we trained the network with mixed datasets. These are i) SC+COCO-1920, which combines SC training and validation sets with images from COCO's corresponding splits, ii) SC+COCO+$R3_{100}$-1920, which adds 100 train and 100 validation images randomly sampled from R3, and iii) RP+COCO-1920, which merges RP and COCO

	APT-36K [14]		R1_D1		R1_D2		R2_D1		R2_D2		R3_D1		R3_D2		RP (validation)		Weigthed avg.		Avg.	
Training Dataset	mAP50	mAP	mAP50	mAP	mAP50	mAP	mAP50	mAP	mAP50	mAP	mAP50	mAP	mAP50	mAP	mAP50	mAP	mAP50	mAP	mAP50	mAP
SF	0.072	0.029	0.770	0.488	0.756	0.490	0.224	0.130	0.597	0.393	0.142	0.092	0.203	0.127	0.104	0.074	0.498	0.318	0.359	0.228
SF-1920	0.103	0.055	0.853	0.568	0.958	0.646	0.873	0.540	0.957	0.616	0.608	0.443	0.651	0.484	0.287	0.191	0.853	0.563	0.661	0.443
SC	0.121	0.046	0.714	0.455	0.830	0.529	0.147	0.084	0.513	0.316	0.156	0.097	0.375	0.202	0.092	0.061	0.482	0.299	0.369	0.224
SC-1920	0.150	0.053	0.907	0.605	0.971	0.664	0.939	0.593	0.968	0.652	0.649	0.476	0.819	0.580	0.331	0.228	0.901	0.604	0.717	0.481
RP	0.260	0.092	0.865	0.487	0.935	0.550	0.808	0.479	0.946	0.593	0.772	0.380	0.922	0.548	0.805	0.453	0.873	0.512	0.789	0.448
RP-1920	0.161	0.066	0.937	0.615	0.980	0.663	**0.989**	0.653	0.982	0.666	0.801	0.532	**0.986**	0.680	**0.914**	**0.636**	0.950	0.636	0.844	0.564
SC+COCO-1920	0.709	0.386	0.943	0.624	0.976	0.676	0.932	0.599	0.977	0.659	0.637	0.481	0.867	0.635	0.350	0.253	0.918	0.621	0.799	0.539
RP+COCO-1920	0.837	0.526	0.967	0.639	0.984	0.681	0.980	**0.656**	0.968	0.684	0.768	0.493	0.981	0.674	0.911	0.626	0.956	0.650	**0.925**	**0.622**
SC+COCO+$R3_{100}$-1920	0.704	0.378	**0.975**	**0.655**	**0.994**	**0.707**	0.963	0.636	**0.991**	**0.691**	**0.986**	**0.733**	0.961	**0.708**	0.432	0.308	**0.975**	**0.676**	0.878	0.602
SC+COCO+RP-1920	0.705	0.383	0.988	0.688	0.994	0.714	0.988	0.652	0.990	0.709	0.869	0.614	0.988	0.756	0.921	0.639	0.976	0.685	0.930	0.644
Pretrained-COCO	**0.879**	**0.566**	0.576	0.376	0.529	0.354	0.421	0.274	0.379	0.258	0.331	0.215	0.551	0.390	0.173	0.123	0.469	0.312	0.480	0.320

TABLE III: Results of the evaluations of the trained models. We report mAP50 and mAP for each dataset as well as both the average and weighted average of these metrics. We divide between models trained on mixed datasets, vanilla ones, and the model pre-trained with COCO. In bold the best results. We underline the best model not using the RP dataset during training in the corresponding validation column.

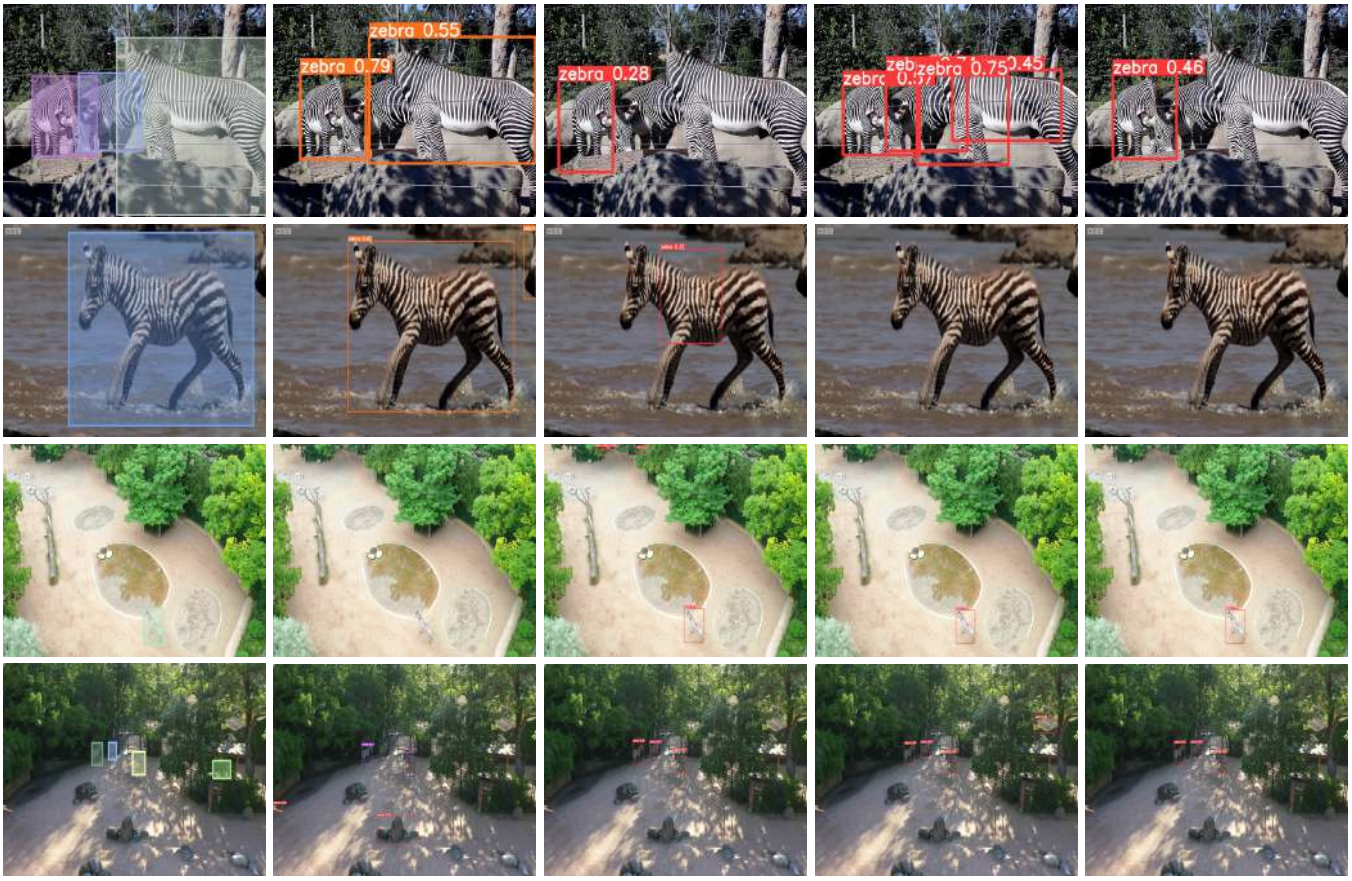
sets. Note that all models trained with RP have been exposed to representing data coming from Rx_Dx, giving them an advantage in these evaluations. The full description of the datasets, including the training a validation set sizes, is reported in II.

All our results are reported in Tab. III. Additionally, we present randomly sampled images from the COCO, APT-36K, R2, and RP datasets for the main models in Fig. 7. Now, we proceed to analyse the results that we report in the table. First, we can notice that the models trained on the bigger image size show higher performances across all datasets and metrics. This is true both for synthetic, i.e. SF and SF-1920, SC and SC-1920, and the real data, i.e. RP and RP-1920. The only exception is the model trained with RP which in the APT-36K performs $\sim 10\%$ better than RP-1920. Overall, the model pre-trained on the COCO dataset works well only on the APT-36K dataset with a mAP50 of $\sim 88\%$, further showing the low variability of these datasets and the incapability to generalize to both different points of view or scenarios. Indeed, the YOLO model pre-trained on COCO achieves at most $\sim 58\%$ accuracy on our data, with an overall weighted average of $\sim 47\%$. The fact that the COCO data is representative of the APT-36K dataset can be evinced also by the performance obtained by the model trained with RP+COCO-1920 dataset. Considering now the synthetic data, i.e. SF-1920 and SC-1920, we can see that the best model overall is SC-1920 which achieves $\sim 5\%$ higher mAP and mAP50 across all tests, with a peak of $\sim 15\%$ on the R3 dataset. This is probably related to the first of the two generation procedures, which resulted in long distances between the zebras and the cameras (see Sec. III-A.1). Our real data instead comprises mostly zebras that are reasonably nearby the drone as seen from the pictures in Fig. 5 and Fig. 7 in the third and fourth rows. The synthetic models may perform poorly on RP validation set and APT-36K due to the generation process. These sets have diverse images, including zebras near the camera in a side view or hidden behind bushes and trees, e.g. second and fourth row in Fig. 7. Moreover, by comparing SC-1920 with the model pre-trained on COCO, we can see how, across all data excluding APT-36K, we obtain higher performances on both metrics of considerable amounts, ranging between $\sim 20\%$ and $\sim 45\%$.

We can now compare the differences between the models trained on synthetic data and real data. For this, we will focus on comparing SC-1920 and RP-1920. The weighted average gap is only 4.9% in the mAP50 and 3.2% on the mAP. The big difference in the simple average is mostly linked to the results obtained in the validation set of RP, which was to be expected. Indeed, we can notice how the model trained on synthetic data performs considerably worse in the RP dataset, with a $\sim 58\%$ reduction in mAP50 and $\sim 41\%$ on mAP. A similar result is depicted when we consider tests on the R3_Dx data, with reductions of $\sim 16\%$ and $\sim 6-10\%$ for the two considered metrics. Nonetheless, with all other datasets, the model trained on synthetic data is comparable to the one trained on real-world captured images of just $1-5\%$. Recall that the RP model was trained on the RP dataset itself, composed of images from the Rx experiments and additional images from point-of-views not generated by our procedure. This clearly demonstrates that, on the considered datasets, the model trained solely with the synthetic data generated using the pipeline described above is perfectly capable of detecting zebras by achieving similar performance on all but two datasets when compared with RP, and significantly overcoming the model pre-trained on COCO in all but APT-36K dataset.

Finally, we consider the mixed models. Unsurprisingly, the one based only on real data, i.e. RP+COCO-1920, performs well on all datasets. The slight reduction in performance in the R2 and R3 datasets is well compensated by the generalization in the APT-36K. This is also the model with the highest average mAP and mAP50. We believe that this is mostly linked to how the dataset was built, with RP that contains data from all Rx experiments combined with the 1916 training images of COCO. Despite that, it is interesting to notice how the models trained with a mixture of synthetic and real data are capable of generalizing across all the datasets as well. Specifically, combining SC and COCO, i.e. SC+COCO-1920, resulted practically in a significant improvement of the performance solely in the APT-36K dataset. Minor improvements are noticeable in the other datasets as well, If to this we add 100 samples from R3, i.e. SC+COCO+$R3_{100}$-1920, we then achieve considerable improvements in the performance w.r.t. SC-1920 on all datasets.

The most noticeable are the ones on APT-36K, of around 55%, and on the RP dataset, of around 10%. The improvement in the R3 is to be expected since we mixed 100 images from that set. Nonetheless, it is remarkable that just a small change in the data brought a $\sim 34\%$ increase in the mAP for this validation test. The SC+COCO+$R3_{100}$-1920 is the model with the highest weighted average precisions



(a) Ground-truth **(b)** COCO **(c)** RP-1920 **(d)** SC-1920 **(e)** SC+COCO+R3_{100}-1920

Fig. 7: Sampled detections. We show in column (a) the ground truth and then the results obtained from (b) the default model (pre-trained on COCO), and the ones trained on (c) RP-1920, (d) SC-1920, and (e) SC+COCO+R3_{100}-1920. The images are randomly taken from the COCO (first row), APT-36K, R2, and RP (last row) datasets. Best viewed in color and zoomed-in.

and is the second best when considering the average mAP and mAP50. We believe that this model would be further improved by having more samples from the COCO dataset in the validation set or, overall, a better-balanced set of samples. Considering that SC is made of 18K images, and both COCO and R3_{100} make up for 2K training images and only 300 validation ones, we can expect an 'overfit' of the final selected model towards scenes which are strongly represented by the synthetic images. Also, in this case, the significant difference in the average mAP and mAP50 is mostly linked to the gap in the results in the RP validation set. For completeness, we also trained the SC+COCO+RP-1920 model, i.e. using the closeby synthetic data, the coco data, and the small set of real data which was precisely labelled. As expected, this is the model which performs best in the majority of the tests, excluding the APT-36K dataset, where the pretrained model performs best, and in R3_D1. However, we must note that RP contains data from all R1, R2 and R3 datasets in both the training and validation sets. Thus, the results in these case are clearly driven by this information. What is interesting to notice is that all mixed models perform similarly in the APT-36K dataset, with $\sim 70\%$ of mAP50 and $\sim 38\%$ mAP, further indicating that a better balancing in the validation set might further boost the performance of these models. Alternatively, a more representative generation strategy could be employed, by including camera locations relative to the zebras more similar to the ones that we can find in the APT-36K or in the COCO dataset. The results suggest that such an approach would be effective as well, perhaps in conjunction with a minimal amount of annotated real data. Finally, considering that zebra stripes are notoriously specific to the individual, it is interesting to notice how, despite the fact we use the same texture for all our generated zebras, we are still able to generalize to different individuals well. This suggests that the network does not focus and learn specifically the pattern it is shown, but rather the general appearance of the animal itself.

V. CONCLUSIONS

In this work, we first demonstrated that the currently available datasets do not generalize well to the task of detecting zebras captured from an aerial point of view. To solve this, we generated a large-scale synthetic dataset of zebras by using GRADE, a state-of-the-art framework for synthetic data generation. The dataset, which is the first of its kind both in terms of size and visual realism, has been released for the benefit of the community. By using that, we performed extensive evaluations by training and testing YOLO with a wide range of combinations of real

and synthetic data. This provides strong evidence that the visual realism of the data generated is very high, because our models showed performances which are as good as the one obtained by a detector trained on real-world labelled data alone. Using synthetic information we can surpass the process of collecting and labelling data in controlled scenarios, thus avoiding the probable introduction of errors. Further testing by using combined synthetic and a small amount of real data showed that we can successfully generalize to a wide variety of scenarios. A known limitation that we need to address is the realism of the adopted environments and more precise placement strategies, which we believe could solve both the generalization problem and the usage of high-resolution images by the network. Future works include the generation of videos instead of just static images, testing with different network architectures like SSD [29] or RCNN [31], and using the synthetic data for different tasks such as keypoints detection.

REFERENCES

[1] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[2] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll'a r, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[4] M. Rottmann and M. Reese, "Automated detection of label errors in semantic segmentation datasets via deep learning and uncertainty quantification," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 3214–3223.

[5] N. Saini, E. Bonetto, E. Price, A. Ahmad, and M. J. Black, "Airpose: Multi-view fusion network for aerial 3d human pose and shape estimation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4805–4812, 2022.

[6] S. Zuffi, A. Kanazawa, D. Jacobs, and M. J. Black, "3D menagerie: Modeling the 3D shape and pose of animals," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[7] S. Zuffi, A. Kanazawa, and M. J. Black, "Lions and tigers and bears: Capturing non-rigid, 3D, articulated shape from images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018.

[8] E. Bonetto, C. Xu, and A. Ahmad, "GRADE: Generating realistic animated dynamic environments for robotics research," *arXiv preprint arXiv:2303.04466*, 2023.

[9] S. E. Ebadi, S. Dhakad, S. Vishwakarma, C. Wang, Y.-C. Jhang, M. Chociej, A. Crespi, A. Thaman, and S. Ganguly, "Psp-hdri+: A synthetic dataset generator for pre-training of human-centric computer vision models," in *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*, 2022.

[10] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa, "Learning from synthetic data: Addressing domain shift for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3752–3761.

[11] J. Cao, H. Tang, H.-S. Fang, X. Shen, C. Lu, and Y.-W. Tai, "Cross-domain adaptation for animal pose estimation," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[12] S. Ye, A. Mathis, and M. W. Mathis, "Panoptic animal pose estimators are zero-shot performers," *arXiv preprint arXiv:2203.07436*, 2022.

[13] H. Yu, Y. Xu, J. Zhang, W. Zhao, Z. Guan, and D. Tao, "AP-10k: A benchmark for animal pose estimation in the wild," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=rH8yliN6C83

[14] Y. Yang, J. Yang, Y. Xu, J. Zhang, L. Lan, and D. Tao, "Apt-36k: A large-scale benchmark for animal pose estimation and tracking," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 17 301–17 313. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/6e566c91d381bd7a45647d9a90838817-Paper-Datasets_and_Benchmarks.pdf

[15] Y. Li, H. Takehara, T. Taketomi, B. Zheng, and M. Nießner, "4dcomplete: Non-rigid motion estimation beyond the observable surface." *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[16] C. Li and G. H. Lee, "From synthetic to real: Unsupervised domain adaptation for animal pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 1482–1491.

[17] A. Mathis, T. Biasi, S. Schneider, M. Yuksekgonul, B. Rogers, M. Bethge, and M. W. Mathis, "Pretraining boosts out-of-domain robustness for pose estimation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2021, pp. 1859–1868.

[18] J. M. Graving, D. Chae, H. Naik, L. Li, B. Koger, B. R. Costelloe, and I. D. Couzin, "Deepposekit, a software toolkit for fast and robust animal pose estimation using deep learning," *eLife*, vol. 8, p. e47994, oct 2019. [Online]. Available: https://doi.org/10.7554/eLife.47994

[19] J. Mu, W. Qiu, G. D. Hager, and A. L. Yuille, "Learning from synthetic animals," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[20] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[21] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017. [Online]. Available: https://arxiv.org/abs/1705.05065

[22] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, A. Kembhavi, A. Gupta, and A. Farhadi, "AI2-THOR: An Interactive 3D Environment for Visual AI," *ArXiv*, vol. abs/1712.05474, 2017.

[23] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D'Arpino, S. Buch, S. Srivastava, L. P. Tchapmi, M. E. Tchapmi, K. Vainio, J. Wong, L. Fei-Fei, and S. Savarese, "igibson 1.0: a simulation environment for interactive tasks in large realistic scenes," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, p. accepted.

[24] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[25] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4cv: A photo-realistic simulator for computer vision applications," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 902–919, Mar. 2018. [Online]. Available: https://doi.org/10.1007/s11263-018-1073-7

[26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 1–16.

[27] W. Wang, Y. Hu, and S. Scherer, "Tartanvo: A generalizable learning-based vo," in *Conference on Robot Learning (CoRL)*, 2020.

[28] "Zebra motions. Free 3D model by Kapi777," https://sketchfab.com/3d-models/zebramotions-2546097d0ea94ba88452ce62c041fb87, [Accessed 11-Apr-2023].

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37.

[30] E. Price and A. Ahmad, "Accelerated video annotation driven by deep detector and tracker," in *Intelligent Autonomous Systems 18*, 2023, to appear.

[31] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Navigating in 3D Uneven Environments through Supervoxels and Nonlinear MPC

Fetullah Atas [1] Grzegorz Cielniak [2] Lars Grimstad [1]

***Abstract*—Navigating uneven and rough terrains presents difficulties, including stability, traversability, sensing, and robustness, making autonomous navigation in these terrains a challenging task. This study introduces a new approach for mobile robots to navigate uneven terrains. The method uses a compact graph of traversable regions on point cloud maps, created through the utilization of supervoxel representation of point clouds. By using this supervoxel graph, the method navigates the robot to any reachable goal pose by utilizing a navigation function and Nonlinear Model Predictive Controller (NMPC). The NMPC ensures kinodynamically feasible and collision-free motion plans, while the supervoxel-based geometric planning generates near-optimal plans by exploiting the terrain information. We conducted extensive navigation experiments in real and simulated 3D uneven terrains and found that the approach performs reliably. Additionally, we compared resulting motion plans to some state-of-the-art sampling-based motion planners in which our method outperformed them in terms of execution time and resulting path lengths. The method can also be adapted to meet specific behavior, like the shortest route or the path with the least slope route. The source code is available in a GitHub repository. [1].**

***Index Terms*—Uneven Terrain Navigation, Outdoor Robotics, Motion Planning.**

I. Introduction

The use of robots in large outdoor environments with uneven terrain has become increasingly popular, with applications ranging from delivery robots to legged robots navigating unstructured terrains and mobile robots performing agricultural tasks. While autonomous navigation in 2D indoor environments has been well-established, the navigation of outdoor uneven terrains poses various challenges that require further investigation. These challenges include the need for robust terrain traversability analysis to prevent stability issues such as tipping over, efficient motion planning that avoids collisions, and adaptability to dynamic environments. An additional challenge is that different robot platforms have different traversability capabilities, depending on their design and locomotion mechanisms. Thus, developing a unified and abstract navigation strategy for uneven environments that meets the requirements of different robot models without significant architectural or code modifications to the navigation framework is a nontrivial task.

To address the challenges mentioned above regarding uneven terrain navigation, this paper proposes a compact approach using pre-built or online point cloud maps. The approach first identifies traversable regions based on the geometrical features of the point cloud by constructing a *supervoxel* representation of the point cloud, with this, the approach determines the robot's geometric motion plan towards a desired reachable goal. The geometric plan is then executed by a Nonlinear Model Predictive Controller (NMPC) that adjusts the robot's actions based on real-time feedback and environmental changes (e.g. dynamic obstacles).

Fig. 1: The figure illustrates the Thorvald II robot's capability to operate in various rough uneven terrains. To fully utilize the robot's strengths in these terrains, it is crucial to have a reliable navigation strategy.

A. Related Work

1) General 3D Navigation: Several works for 3D navigation have been proposed based on 2D processing such as in Wang et al. [27] and Pütz et al. [21]. However, these methods are limited as they do not exclusively consider rapid elevation changes as shown in Atas et al. [2]. Pütz et al. [20] proposed a mesh-based navigation where they created triangulation of underlying point cloud. The idea is spiritually similar to what we propose, however, our approach works as anytime, meaning that the method calculates supervoxels for each new navigation request while still meeting real-time constraints as supervoxel construction is efficient. More recent work by Fan et al. [4] uses e Conditional Value-at-Risk (CVaR) based terrain traversability analysis to navigate in uneven terrains. Additionally, Jian et al. [8] proposes a plane-fitting-based navigation framework, PUTN for uneven terrains. Our work is similar to that of PUTN, a major difference is that for the global geometric path generation they propose plane-fitting-based RRT*, however, the proposed motion planning algorithm is not compared to more recent work in the sampling-based planners that are highlighted in the next subsection of related work. Additionally, Atas et al. [2] proposed a surfel-

[1] Norwegian University of Life Sciences {fetullah.atas, lars.grimstad} @nmbu.no
[2] University of Lincoln gcielniak@lincoln.ac.uk
[1]https://github.com/NMBURobotics/vox_nav

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

based framework for uneven terrain navigation; however, the work did not deal with dynamic obstacles and assumed static environments.

2) Geometric Motion Planning: Sampling-based motion planning has been demonstrated as an effective strategy to deal with planning for robots in uneven environments in various works such as in Jian et al. [8] and Atas et al. [2]. In the following, we provide a brief review on sampling-based motion planning. The PRM Kavraki et al. [10] and RRT Lavalle and Kuffner [12] algorithms and their dozens of variants have paved the way for sampling-based motion planning. The authors in Karaman and Frazzoli [9] investigated sampling-based planners' *completeness* and *optimality* properties to understand formal guarantees further. Some sampling-based planners were later improved to account for non-holonomic constraints Palmieri et al. [18]. A library called Open Motion Planning Library (OMPL) Sucan et al. [26] contains implementations of many sampling-based planners. The majority of these planners are based on RRT* and PRM* Karaman and Frazzoli [9] planners (e.g., RRTX Otte and Frazzoli [17], LazyPRM* Bohlin and Kavraki [3], InformedRRT* Gammell et al. [5], and so on), but a newer collection of planners uses a structure that contains both graphs and trees (e.g., BIT* Gammell et al. [6], ABIT* Strub and Gammell [24], AIT* Strub and Gammell [25]). Some methods utilize parallelized approaches (e.g., CFOREST Otte and Correll [16], AnytimePartShortening (APS) Luna et al. [13]), in which many planners execute concurrently on different threads while planners inform each other of milestones reached, leading to better performance overall.

3) NMPC for Collision-Free Control: Numerous studies have established the effectiveness of Nonlinear Model Predictive Control (NMPC) in achieving optimal control of diverse robotic systems including mobile robots Salimi Lafmejani and Berman [22], unmanned aerial vehicles (UAVs) Mansouri et al. [15], and autonomous vehicles Yu et al. [28]. These systems share a common trait, which is the presence of multiple constraints that must be accounted for in each control cycle. These constraints may include limitations on state dynamics, non-collision requirements, and limits on state variables such as velocity or acceleration, among others. The approach we propose shares similarities with the work of Yu et al. [28], who also present an NMPC scheme for obstacle-aware uneven terrain navigation. However, our method offers two distinct advantages over their work. Firstly, we validate our approach through experiments on an actual robot, in addition to simulations. Secondly, our method demonstrates a faster run cycle of approximately 20Hz, whereas their reported control loop runs at approximately 3Hz. The NMPC scheme proposed in PUTN Jian et al. [8] also holds relevance. However, the authors of this scheme have used a 2D kinematic model and the speed of the control loop has not been explicitly reported.

B. Contributions

The specific contributions of our work include the following:

- We introduce a new navigation strategy that utilizes supervoxels and NMPC, which have been shown to be effective in real uneven terrain navigation.
- Through a series of experiments in simulated and real-world uneven terrains, we demonstrate that the proposed method's planning module outperforms existing state-of-the-art sampling-based planners in terms of efficiency and flexibility.
- We provide an open-source implementation of our approach.

II. Approach

A. Problem Statement

A successful navigation task will produce a discrete motion plan defined by state $\tilde{s}$ and control $\tilde{u}$ sets at the final stage t. The state and control sets are as follows:

$$\tilde{s} = (s_1, s_2, ..., s_t), \tilde{u} = (u_1, u_2, ..., u_t). \quad (1)$$

Based on Eq. 1, we define the following components for feasible navigation:

1) A finite state space X.
2) A control space $U(s)$ for each state $s \in X$.
3) A state transition function f that produces next state $f(s, u) \in X$.
4) A set of stages denoted by t that begins at $t = 0$ and continues indefinitely, and a goal set $X_G \in X$.

A successful navigation task is achieved by the function Υ that maps every state to control; $X \longrightarrow U$.

B. General Overview

Our approach is divided into three stages. First, we develop a method for evaluating the traversability of a point cloud map by analyzing the geometric characteristics of local terrain patches. The second stage involves the construction of supervoxels on point cloud maps with quantified traversability values, the supervoxels are then used to calculate minimal cost paths either with the Dijkstra or A* graph search algorithms. The Final stage utilizes the calculated geometrical path to determine the optimal and kinodynamically feasible control commands until a termination condition is met (e.g. goal reached, collision detected, etc.). In the following sub-sections, we will delve into the specifics of each stage of our approach.

C. Traversability Assessment via Local Geometric Features

Suppose that an environment is represented with a point cloud map. We discretize this point cloud map through uniform sampling with a voxel size of d_s, resulting in a subset of j points referred to as the sampled point set P_S. From this sampled point set, we extract local terrain features such as roughness, tilt, and maximum height by analyzing the geometrical relationship between the sampled points and their neighboring points from the original point cloud map.

To obtain geometric features, we search for the radial nearest neighbors of each sampled point in the original point

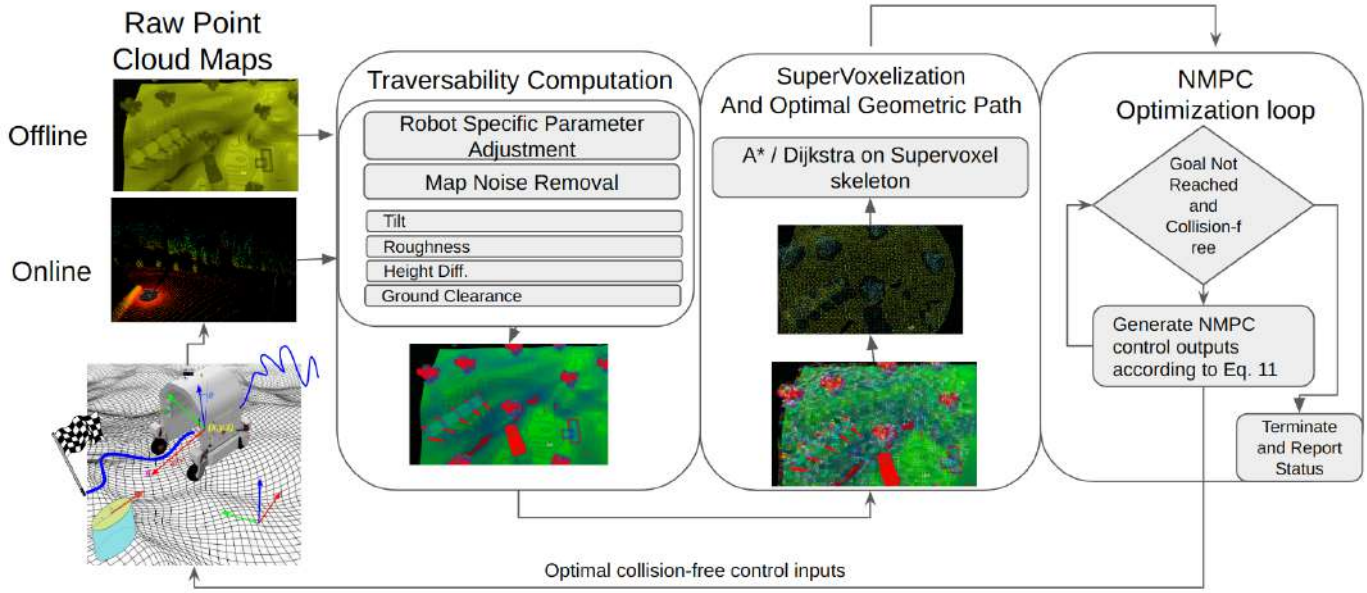


Fig. 2: A comprehensive overview of our approach is presented. The first step involves the calculation of the traversability of a given point cloud map, taking into account the specific capabilities of the robot (e.g. max tilt angle, min distance for ground clearance, etc.). Secondly, utilizing the obtained traversability map, the traversable regions are segmented into compact supervoxels. Finally, the NMPC optimization process is employed for path tracking by producing optimal control values that are kinodynamically feasible.

cloud, which forms disk-like structures. Points within each disk are then used to extract several geometric features.

The normal vectors for these disk-like structures are determined by fitting a plane to its points using RANSAC plane fitting. This plane fitting is expressed in Eq. 2.

$$\begin{aligned} Ax + By + Cz + D = 0, \\ n_s = (n_{sx}, n_{sy}, n_{sz}) = (A, B, C) \end{aligned} \tag{2}$$

The coefficients A, B, C, D represent the scalar equation of a plane. The normal vector for this plane (of a disk) is given by n_s.

We use the following cost critics to assign a traversability cost to each disk.

- Tilt of the slope within the disk, represented by t_d.
- Average deviation of points from the plane of the disk, represented by pd_d, to measure roughness.
- Maximum height difference of points within the disk, represented by hd_d.
- Ground clearance for the bottom chassis of the robot, represented by gc_d.

These criteria aim to evaluate the terrain patches based on the robot's physical limitations, such as the maximum tilt (roll or pitch) angle, the roughness of the terrain, ground clearance, etc. The resulting cost values will reflect these constraints and indicate the disk's suitability for traversal. The cost values for each disk are computed using the following equations, which are based on the geometrical properties of the points within disks. The cost values are then distributed to all points in the global point cloud map.

$$t_d = \arg\max(|\arctan(n_{dx}, n_{dz})|, |\arctan(n_{dy}, n_{dz})|) \tag{3}$$

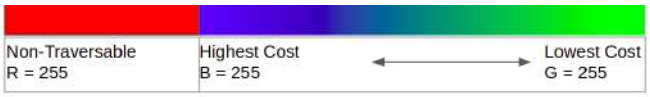


Fig. 3: Cost values are encoded with RGB channels to the terrain point cloud maps.

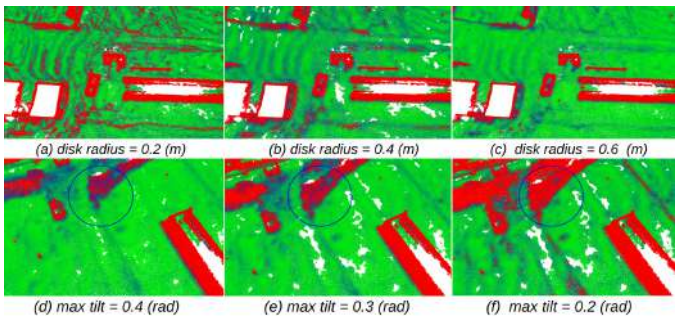


Fig. 4: The impact of varying parameters on the traversability map is shown. In (a), (b), and (c), the effect of using different disk radii is depicted. In (d), (e), and (f), the effect of choosing different max ranges for the tilt cost is depicted. Just as the tilt cost critic can be adjusted, the range and weight of other cost critics can also be customized, providing flexibility to accommodate robots with different traversability capabilities. The RGB values encode traversability cost values. See Fig. 3.

here, $\arctan(n_{dx}, n_{dz})$ and $\arctan(n_{dy}, n_{dz})$ represent the roll and pitch of the disk, respectively.

$$pd_d = 1/n \sum_{k=1}^{n} |\frac{AP_x + BP_y + CP_z + D}{\sqrt{A^2 + B^2 + C^2}}| \tag{4}$$

where n denotes the number of points within the disk. The roughness of a disk is determined by the average point deviation from the disk plane, pd_d. This value is calculated

by summing up the distances from each point within the disk to the disk plane and then dividing the result by the number of points in the disk.

$$hd_d = \arg\max_{0:n}(P_{z1}, .., P_{zn}) - \arg\min_{0:n}(P_{z1}, ..., P_{zn}), \quad (5)$$

The max height difference hd_d is computed as the difference between the highest and lowest points in the disk.

$$gc_d = \forall k \in \{1, .., n\}, max(\frac{AP_{kx} + BP_{ky} + CP_{kz} + D}{\sqrt{A^2 + B^2 + C^2}}). \quad (6)$$

Ground clearance of each disk, gc_s, is calculated as the maximum distance from the disk plane to any point within the disk, along the normal vector direction. This value must be less than the distance from the bottom of the robot chassis to the ground to ensure safe navigation. We combine the value of each critic with their corresponding weight factors α_t, α_{pd}, α_{hd}, α_{gc} to produce the final traversability cost value c_s for each disk:

$$c_s = (\alpha_t \cdot t_d + \alpha_{pd} \cdot pd_d + \alpha_{hd} \cdot hd_d + \alpha_{gc} \cdot gc_d) \quad (7)$$

where α_t, α_{pd}, α_{hd}, α_{gc} are the weighting factors that determine the relative importance of each cost critic in the final traversability cost value. The weighting factors are adjustable parameters to weigh each critic according to the desired behavior. Refer to Fig. 4 to see the impact of some cost critics and other parameters (e.g. disk radius) on the traversability map.

D. Supervoxels for Connected Safe Regions

Our approach explores *supervoxel* phenomena of point clouds as a way of marking reachable regions for navigation over uneven 3D terrains.

Supervoxels were proposed as an intermediate representation for a dense underlying point cloud to lower the computational requirements by segmentation algorithms, which are analogous to superpixels for image segmentation. Supervoxels are suitable for navigation as they provide a connectivity graph over the underlying point clouds. It is ensured that the supervoxels are evenly distributed within the actual observed space rather than being confined to the projected image plane. This is achieved through the implementation of a seeding methodology based in 3D space and a flow-constrained local iterative clustering algorithm that incorporates both color and geometric features. Furthermore, the supervoxels can be utilized directly on raw point clouds that are generated by combining multiple calibrated RGB+D cameras or LIDARs, thereby providing a full 3-dimensional supervoxel graph, which is capable of meeting the demands of robotic applications in terms of speed. We refer you to Papon et al. [19] for further details on supervoxel construction for point cloud segmentation.

In the previous subsection, a traversability measure was regressed across the point clouds, allowing reachable portions (non-red areas) of the point cloud to be derived. The traversability of each local terrain patch was quantified by regressing the cost values to each patch.

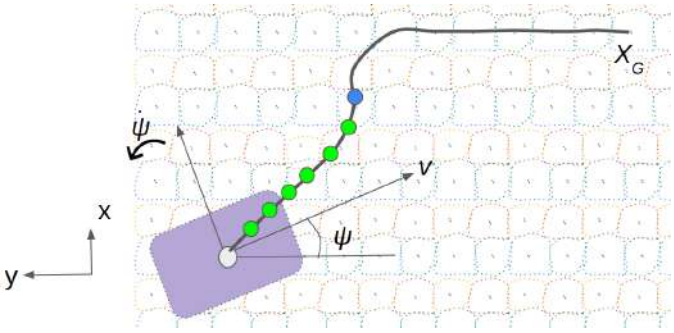


Fig. 5: Top view of path tracking over point cloud supervoxels. The local reference path, which is used as input for the NMPC algorithm, is composed of linearly interpolated path segments, with a time step of T between consecutive states in the path.

In Fig. 4, red points on the left picture are designated as non-traversable regions, and no supervoxels are created on these regions. Given the connectivity graph of supervoxels, it is straightforward to compute cost-optimal paths over state space X with graph traversal algorithms such as Dijkstra or A*. The area covered by a supervoxel is controlled with two parameters, seed resolution p_{sr} and resolution p_r. These two parameters determine the resolution of the resulting geometric path.

E. NMPC for Optimal Control Policy

In this study, we employ the NMPC-based method to track trajectory while ensuring compliance with dynamic constraints and addressing potential collisions with stationary or moving obstacles that are represented with 3D ellipsoids. It is important to note that detecting obstacles is not a focus of this study.

The state dynamics are represented with a typical discrete dynamic system;

$$s_{k+1} = f(s_k, u_k), s_k \in X, u_k \in U \quad (8)$$

where the state vector is $s_k = [x_k, y_k, z_k, \psi_k, \theta_k, \phi_k, v]$. This includes the robot's x, y, z position, the robot's yaw, pitch, and roll denoted by ψ_k, θ_k, ϕ_k, and finally v as the robot's linear velocity along $x - axis$ of its local frame. The control vector, $u_k = [\dot{v}_k, \dot{\psi}_k]$, includes linear acceleration $\dot{v}$, and angular velocity $\dot{\psi}$. The continuous kinematic model is described by the following equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \\ \dot{\delta} \\ \dot{\phi} \\ \dot{v} \end{bmatrix} = v \begin{bmatrix} cos(\psi)cos(\theta) \\ sin(\psi)cos(\theta) \\ cos(\theta) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{\psi} \end{bmatrix} \quad (9)$$

The Fig. 5 illustrates the path tracking over the terrain. The goal of the NMPC strategy is to keep the system as close as possible to the reference path while ensuring kinematic and dynamic constraints.

The NMPC optimization is formulated as;

$$u^*_{0:h-1} = \arg\min_{0:h-1}(J_t(s_{0:h}, u_{0:h-1}) + J_{ir}(u_{0:h-1}) + J_{obs}(s_{0:h}, O)) \tag{10}$$

subject to,

$$\begin{aligned} s_{k+1} &= f(s_k, u_k), \\ B(s_k) \cap O &= \varnothing, \\ U_{min} \leq U_{0:h-1} &\leq U_{max} \end{aligned} \tag{11}$$

- The cost term $J_t(s_{0:h}, u_{0:h-1})$ is used to ensure that the system follows the global path as closely as possible to the Euclidean path between supervoxels. This cost term also guarantees that the system's kinematic constraints are met since $s_{k+1} = f(s_k, u_k)$ is a hard equality constraint.
- The term $J_{ir}(u_{0:h-1})$ represents the input rate cost, which aims to minimize the jerky movements of the control outputs.
- The term $J_{obs}(s_{0:h}, O)$ takes into account the cost associated with the proximity of the system to obstacles, which are represented as time-varying ellipsoids. Collision-free control policies are guaranteed with $B(s_k) \cap O = \varnothing$.

Specifically, the tracking cost J_t and the input rate cost J_{ir} are defined as follows:

$$\begin{aligned} J_t(s_{0:h}) &= \sum_{i=0}^{h-1}(s_i - s_i^{ref})Q(s_i - s_i^{ref})^T, \\ J_{ir}(u_{0:h}) &= \sum_{i=0}^{h-1}(u_i - u_i^{dv})R(u_i - u_i^{dv})^T \end{aligned} \tag{12}$$

Q and R diagonal matrices are used to penalize specific state errors and input jerks.

Finally, the obstacle costs are defined as follows:

$$\begin{aligned} D_{obs}(s_{0:h}) = &\sum_{i=0}^{h-1}\sum_{j=0}^{n}(x_i - x_j^{obs})^2/(a_j^{obs}/2.0 + r)^2, \\ &+ \sum_{i=0}^{h-1}\sum_{j=0}^{n}(y_i - y_j^{obs})^2/(b_j^{obs}/2.0 + r)^2, \\ J_{obs}(s_{0:h}) &= e^{(1.0/D_{obs}(s_{0:h}))} \end{aligned} \tag{13}$$

As shown in Eq. 13, the cost is inversely proportional to the distance between the robot and obstacles. The variable n represents the number of obstacles in O, h is the time horizon considered in the NMPC setup and r is the robot radius. Each obstacle O_j is represented by a vector $[x_j^{obs}, y_j^{obs}, a_j^{obs}, b_j^{obs}]$, which are used in Eq. 14 to define the ellipse, see Fig. 2.

$$\frac{(x^{obs})^2}{(a^{obs})^2} + \frac{(y^{obs})^2}{(b^{obs})^2} = 1 \tag{14}$$

We implement the NMPC using CasADi Andersson et al. [1], a software framework that specializes in nonlinear optimization and optimal control. Although the NMPC algorithm can handle an arbitrary number of obstacles, to improve the speed of the control loop, we limit the number of obstacles considered by the algorithm to those that are within 20 meters of the robot's current position.

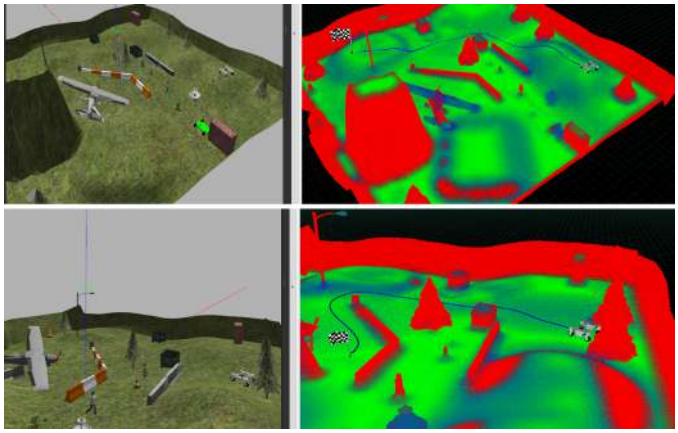


Fig. 6: The figure presents the sample navigation scenes in a simulated environment with uneven terrain and obstacles. The starting position of the robot is indicated by a checkered flag.

III. Experiments

We conduct experiments in 3D uneven terrains with varied slopes and roughness in both simulated and real environments. Buildings, poles, trees, and other items are examples of objects in the environment. The maps are around 300x300 meters in size. The simulated environment consists of more steep hills and more significant variance in terms of inclination. Our method relies on point cloud maps for the occupancy information of environments. Hence we construct point cloud maps of real environments with a SLAM method named LIO-SAM by Shan et al. [23]. For building maps in real environments, we use the Ouster OS1-64 LiDAR and MTi-30-2A8G4 Xsens IMU. In the simulation, we rely on the Gazebo simulator and simulated sensors and extract point cloud maps from the simulated environment through a software plugin Koenig and Howard [11]. The onboard localization was performed through a combination of ICP-based LIDAR, wheel odometry, and IMU. The autonomy system is run on a ZOTAC ZBOX with an Intel Core i7 CPU and an Nvidia RTX 2060.

For the robot platform, the Thorvald II modular robot was used Grimstad and From [7]. The software specifically designed for Thorvald runs on an Intel NUC computer equipped with a Core i7 CPU, and the communication between all components is managed by ROS 2 Macenski et al. [14]. In evaluating the motion plan generated, two metrics are considered: the length of the path and the computation time. Tab. I is created based on these two evaluation metrics.

A. Qualitative Performance Evaluation

In Fig. 8a, the robot arrives at the destination successfully; the path depicted in blue is the resultant feedback plan, the start pose is marked with a checkered flag. Similarly, in Fig. 8a the robot navigates from a similar start pose (as in Fig. 8b) but to a closer goal pose, resulting in a near-optimal path depicted in blue color. For Fig. 8a and Fig. 8b, we refer to *optimality* in sense of Euclidean distance. However, with the availability of traversability costs (represented with RGB colors in point cloud maps, seeFig. 3) we can re-adjust the sense of optimality

to refer to the traversability cost in point cloud maps. Hence the method can be configured such that it can navigate the robot with one of the following properties:

- least inclined path.
- shortest path.
- least rough path.

In Fig. 7 (a), we configure the cost term such that it accounts for traversability costs. The resulting motion plans avoid higher-cost regions, resulting in an optimal motion plan where less-inclined, less-rough terrain segments are preferred rather than a short distance. Hence the resultant motion plans tend to navigate through greener regions. This allows the robot to navigate through inclined terrains safely by avoiding steep hills and rough terrain patches.

B. Tuning the Method

The tuning of our algorithm is straightforward as it involves adjusting only a small number of configuration parameters. Specifically, two parameters, known as seed resolution (p_{sr}) and resolution (p_r), are used in the construction of supervoxels. The seed resolution (p_{sr}) is responsible for identifying the radius of nearest neighbors that are used to compute the supervoxel boundaries, while the resolution (p_r) determines the actual size of the supervoxel. Smaller values of p_r result in finer resolution of the global path, but we recommend using values in the range of $[0.1, 0.25]$ for large-scale outdoor robot navigation. In our reported results, we set $p_{sr} = 1.0$ and $p_r = 0.2$. The general rule is that p_r should be large enough to cover a number of points.

$$\begin{aligned} Q = Diag(10, 10, 10, 0.1, 0, 0, 0.1), \\ R = Diag(10, 100) \end{aligned} \tag{15}$$

In Eq. 12, we introduced tracking cost and input rate gain matrices Q and R. In this paper, we do not propose a methodological approach to tune these gain matrices optimally. However, from the experimental observation, we realize that the angular rate needs to be penalized higher as otherwise, the robot oscillates. From the optimality perspective, it is best to control the robot as close as possible to the coarse plan since the optimal property is explicitly contained within graph traversal by Dijkstra or A*. Therefore in Q, we penalize positional errors on x, y, z heavier than the controllable heading ψ and velocity v. In the presented results, Q and R were chosen as in Eq. 15.

We also provide customizable *objective* selection in our software implementation to optimize ensuing feedback plans, allowing us to achieve plans with the various characteristics itemized in Subsec. III-A.

C. Comparison to Sampling-based planners

In this subsection, we evaluate the geometric plan produced by our approach. As baselines, we choose some of the state-of-the-art sampling-based planners. We compare the proposed method to various planners such as AnytimePathShortening (APS), CFOREST, AIT*, and others. We create a benchmark of planning problems consisting of 20 unique planning problems in the map depicted atFig. 7. Each planner is requested to solve 20 different planning problems five times, resulting in a total of 100 runs. The results of 100 runs are presented in Tab. I.

Our method produced shorter paths in significantly less time. Optimal sampling-based planners require a timeout parameter where they use all allowed time to construct a valid path with minimal cost, in this case, the shortest length. To observe the behavior of sampling-based planners with different time constraints, we set the timeout to 10 and 20 seconds consecutively. The planners can improve their performance due to the available time in the 20-second setup. To establish a baseline, we run the APS planner for 60 seconds; our method almost achieves APS's performance in less than one second. Based on the results in Tab. I, our technique distinguishes itself from sampling-based planners by its short execution time and deterministic nature. Compared to sampling-based planners, our method has the lowest final plan length deviation in both setups (10 and 20 seconds), indicating consistency in the obtained results.

Planners	Length Mean (20 sec.)	Length Mean (10 sec.)
RRT*	65.2 ± 9.6	75.7 ± 13.9
PRM*	51.2 ± 2.2	67.0 ± 12.0
AIT*	52.0 ± 3.7	58.8 ± 8.3
CFOREST	51.6 ± 3.2	58.3 ± 6.9
APS	50.7 ± 1.8	54.1 ± 6.1
APS (60 sec.)	48.7 ± 0.5	
Ours (0.65 sec.)	**49.5** ± 1.3	

TABLE I: Means of acquired path lengths with 10 sec—timeout from 100 runs. A lower mean value indicates that planners achieve better performance for both metrics (shorter path in a shorter time).

The results in Tab. I show that the proposed approach is able to produce shorter paths in a significantly shorter amount of time compared to other planners. Specifically, the approach generated 8.56% shorter paths than the next best planner (APS) while only taking 6.68% of the time consumed by APS. It was also observed that the best-performing baseline planner (APS) required more than 40 seconds to surpass the proposed method. Additionally, it should be noted that the proposed method is an anytime planner that re-computes supervoxels efficiently for each planning request and the reported times also include the time spent for supervoxel creation.

Our method is able to generate plans at a significantly faster rate than sampling-based planners. This is because our method leverages the underlying terrain information in an active manner, whereas sampling-based planners do not take advantage of this information by default. As a result, generating a feasible plan for uneven terrain (e.g. no jump-overs above buildings) for a ground robot in uneven terrain can be a time-consuming process using sampling-based planners. This highlights the efficiency of the proposed method.

IV. Conclusion

In this paper, we presented a novel approach for navigating uneven terrains based on geometric traversability analysis,

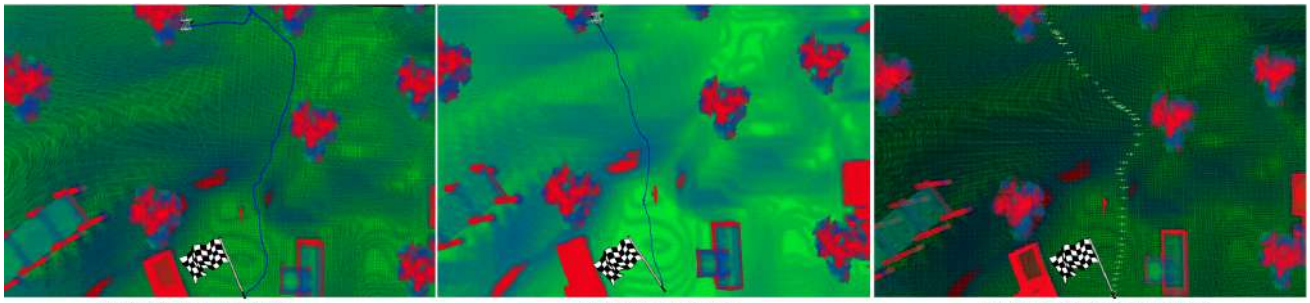
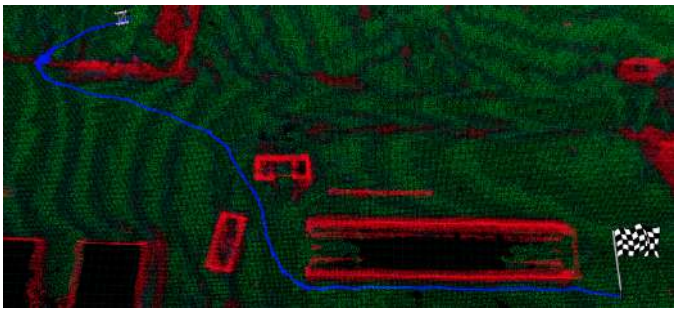
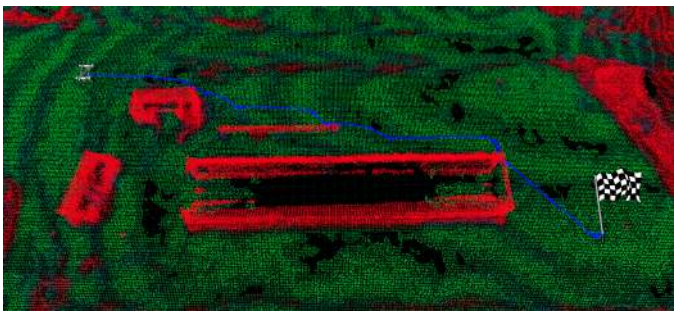


Fig. 7: A comparison of the performance of the proposed approach with an objective set to the inclination cost (a) and Euclidean distance (b), to that of the APS planner (c). The APS planner provides a longer path than the proposed approach despite taking more than ten times longer to compute.



(a)



(b)

Fig. 8: Example optimal motion plans resulting from the proposed the approach in a real uneven environment. The chequered flag indicates the robot's start pose, while the robot's icon indicates the robot's final pose.

supervoxels, and NMPC. Our experimental results on a real robot operating in a 3D environment demonstrate the method's efficiency. Specifically, our approach generated better-quality geometric plans in less time than state-of-the-art sampling-based planners on a benchmark. Additionally, the method enables a robot to navigate uneven 3D terrains with various objectives, such as finding the shortest or least-inclined path while dynamically avoiding obstacles.

One potential limitation of our proposed method is that it does not explicitly model uncertainty that may arise from the interaction between the robot and the uneven terrain. This is due to the lack of sensors on the robot platform that can perceive terrain parameters, such as soil compaction and slippage estimation. In future work, we plan to extend our approach to address this limitation and handle uncertainties that arise from the robot and uneven terrain interaction.

References

[1] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1): 1–36, 2019. doi: 10.1007/s12532-018-0139-4.

[2] Fetullah Atas, Grzegorz Cielniak, and Lars Grimstad. Elevation state-space: Surfel-based navigation in uneven environments for mobile robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5715–5721, 2022. doi: 10.1109/IROS47612.2022.9981647.

[3] Robert Bohlin and Lydia Kavraki. Path planning using lazy prm. volume 1, pages 521 – 528 vol.1, 02 2000. ISBN 0-7803-5886-4. doi: 10.1109/ROBOT.2000.844107.

[4] David D. Fan, Kyohei Otsu, Yuki Kubo, Anushri Dixit, Joel Burdick, and Ali-Akbar Agha-Mohammadi. STEP: stochastic traversability evaluation and planning for safe off-road navigation. *CoRR*, abs/2103.02828, 2021. URL https://arxiv.org/abs/2103.02828.

[5] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Informed rrt*: Optimal incremental path planning focused through an admissible ellipsoidal heuristic. *CoRR*, abs/1404.2334, 2014. URL http://arxiv.org/abs/1404.2334.

[6] Jonathan D Gammell, Timothy D Barfoot, and Siddhartha S Srinivasa. Batch informed trees (bit*): Informed asymptotically optimal anytime search. *The International Journal of Robotics Research*, 39(5):543–567, 2020. doi: 10.1177/0278364919890396. URL https://doi.org/10.1177/0278364919890396.

[7] Lars Grimstad and Pål Johan From. Thorvald ii - a modular and re-configurable agricultural robot. *IFAC-PapersOnLine*, 50(1):4588–4593, 2017. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2017.08.1005. URL https://www.sciencedirect.com/science/article/pii/S2405896317314830. 20th IFAC World Congress.

[8] Zhuozhu Jian, Zihong Lu, Xiao Zhou, Bin Lan, Anxing

Xiao, Xueqian Wang, and Bin Liang. Putn: A plane-fitting based uneven terrain navigation framework, 2022.

[9] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *CoRR*, abs/1105.1186, 2011. URL http://arxiv.org/abs/1105.1186.

[10] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. doi: 10.1109/70.508439.

[11] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004. doi: 10.1109/IROS.2004.1389727.

[12] Steven Lavalle and James Kuffner. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: New directions*, 01 2000.

[13] R. Luna, I. A. Şucan, M. Moll, and L. E. Kavraki. Anytime solution optimization for sampling-based motion planning. In *2013 IEEE International Conference on Robotics and Automation*, pages 5068–5074, 2013. doi: 10.1109/ICRA.2013.6631301.

[14] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022. doi: 10.1126/scirobotics.abm6074. URL https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[15] Sina Sharif Mansouri, Christoforos Kanellakis, Björn Lindqvist, Farhad Pourkamali-Anaraki, Ali-akbar Aghamohammadi, Joel Burdick, and George Nikolakopoulos. A unified nmpc scheme for mavs navigation with 3d collision avoidance under position uncertainty. *IEEE Robotics and Automation Letters*, 5(4):5740–5747, 2020. doi: 10.1109/LRA.2020.3010485.

[16] M. Otte and N. Correll. C-forest: Parallel shortest path planning with superlinear speedup. *IEEE Transactions on Robotics*, 29(3):798–806, 2013. doi: 10.1109/TRO.2013.2240176.

[17] M. Otte and Emilio Frazzoli. Rrtx: Real-time motion planning/replanning for environments with unpredictable obstacles. In *WAFR*, 2014.

[18] Luigi Palmieri, Sven Koenig, and Kai O. Arras. Rrt-based nonholonomic motion planning using any-angle path biasing. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2775–2781, 2016. doi: 10.1109/ICRA.2016.7487439.

[19] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Wörgötter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, 2013. doi: 10.1109/CVPR.2013.264.

[20] Sebastian Pütz, Thomas Wiemann, Jochen Sprickerhof, and Joachim Hertzberg. 3d navigation mesh generation for path planning in uneven terrain. *IFAC-PapersOnLine*, 49(15):212–217, 2016. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2016.07.734. URL https://www.sciencedirect.com/science/article/pii/S2405896316310102. 9th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2016.

[21] Sebastian Pütz, Jorge Santos Simón, and Joachim Hertzberg. Move base flex a highly flexible navigation framework for mobile robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3416–3421, 2018. doi: 10.1109/IROS.2018.8593829.

[22] Amir Salimi Lafmejani and Spring Berman. Nonlinear mpc for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots. *Robotics and Autonomous Systems*, 141:103774, 2021. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2021.103774. URL https://www.sciencedirect.com/science/article/pii/S0921889021000592.

[23] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Rus Daniela. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142. IEEE, 2020.

[24] Marlin P. Strub and Jonathan D. Gammell. Advanced bit* (abit*): Sampling-based planning with advanced graph-search techniques. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020. doi: 10.1109/icra40945.2020.9196580. URL http://dx.doi.org/10.1109/ICRA40945.2020.9196580.

[25] Marlin P Strub and Jonathan D Gammell. Adaptively Informed Trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3191–3198, 31 May – 31 August 2020. doi: 10.1109/ICRA40945.2020.9197338.

[26] I. A. Sucan, M. Moll, and L. E. Kavraki. The open motion planning library. *IEEE Robotics Automation Magazine*, 19(4):72–82, 2012. doi: 10.1109/MRA.2012.2205651.

[27] Chaoqun Wang, Lili Meng, Sizhen She, Ian M. Mitchell, Teng Li, Frederick Tung, Weiwei Wan, Max. Q.-H. Meng, and Clarence W. de Silva. Autonomous mobile robot navigation in uneven and unstructured indoor environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 109–116, 2017. doi: 10.1109/IROS.2017.8202145.

[28] Siyuan Yu, Congkai Shen, and Tulga Ersal. Nonlinear model predictive planning and control for high-speed autonomous vehicles on 3d terrains. *IFAC-PapersOnLine*, 54(20):412–417, 2021. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2021.11.208. URL https://www.sciencedirect.com/science/article/pii/S2405896321022527. Modeling, Estimation and Control Conference MECC 2021.

Ant Colony Optimization for Retail based Capacitated Vehicle Routing Problem with Pickup and Delivery for Mobile Robots

Agha Ali Haider Qizilbash[1] Anoj Kumar Yadav[2], Kevin Bregler[3], Werner Kraus[4]

***Abstract*— Mobile Robots have been the key for automation in various applications including picking and placing items in a retail store. Capacitated Vehicle Routing Problem with Pickup and Delivery(CVRP-PD) is widely used in similar applications like package delivery vehicles and mobile robots in retail, where mobile robots have a capacity limit such as weight and have to pickup and drop multiple items during their tour in an optimized manner. However, Retail application comes with more challenges where there could be multiple fixed deposit locations for particular pickup items such as packing counters and after delivering some items mobile robots can again regain capacity and be able to pickup more items during the same run. In this paper, we consider these constraints for retail applications and optimize retail orders for all mobile robots present in the environment, where the order requests for pickup and delivery of products at various locations while mobile robots have different maximum load capacities and robots can regain their capacity once they have dropped some items at their particular delivery locations. In this paper, we propose a method to solve this retail based CVRP-PD using Ant Colony Optimization(ACO). We take an industrial use-case and test the method with different order sizes and robot parameters. The results have been promising and used to solve the use-case under consideration. In addition, we also evaluate the results and propose future prospects.**

I. INTRODUCTION

Automated Guided Vehicles (AGV) and Autonomous Mobile Robots (AMR) have entered in almost all areas of applications whether it is warehouse logistics or agriculture precision farming. Among them is Retail, which has one of the most automation potential and since covid-19 pandemic, one of the most affected ones in terms of manual labor. This has paved its need of AGVs and AMRs in retail applications. Use of these mobile robots allows humans to reduce their efforts by having items picked up and delivered to deposit locations in retail environments similar to close warehouse logistics. This application of mobile robots fill the labor shortage gap for warehouses and retail applications as well as overcome the limitation of humans to work for long periods of time thereby minimizing efforts to maximize work. But even these mobile robots have some limitations such as battery life, collision-free movements, high costs and maximum payload. These mobile robots also demand high investments and may also require skilled expertise for the installation and further development of software for the mobile robots in any application. In research by Sabattini et al., describe the adoption of mobile robots, to make it cost-effective with accurate localization systems, better routing plans, and traffic management systems (e.g., traffic caused by other robots and the ability to work interactively with humans). As a result, it is critical to have optimized technology for mobile robots in any industrial application in order for them to be cost-effective [17]. So to efficiently use a fleet of mobile robots in retail or warehouse applications, optimized routing is of foremost important. So considering the capacities of the robot, the orders are to be completed while optimizing for the distance traveled by the robots. This will ensure orders being completed in shorter time and in result, be able to obtain return of investment faster.

The main objective of the paper is to efficiently use the fleet of mobile robots in a retail application such that the orders placed are completed by robots by minimizing the total traveling distance and minimizing the number of robots required to complete the order. Orders placed for pickup and delivery in a retail applications, where multiple items are going to be picked from various locations within an environment and delivered to fixed delivery locations. Every item that is going to be picked up will have a specific delivery location where it should be dropped off and have some specific weight. Whereas mobile robots also have a maximum limit to how much they can carry. Randomly allocating the orders to robots would lead to redundancy where the same location is visited multiple times, or visiting a location by a robot where another robot has just passed by. Also, it might lead to circumstances where some robots are overloaded or worse, moving mostly without much load. Manually calculating the possible routes and assigning the orders to the robots demands high calculation and time. Approaches like greedy algorithms are often used for smaller similar applications where nearest pickup item is assigned to nearest robot. But this is also not useful as it takes no account of fixed delivery location and weight thresholds. Therefore, this paper aims to optimize routes for mobile robots so that the distance traveled is minimized and all the orders are completed considering all the constraints related to the order and robot capacities in a retail setting.

II. BACKGROUND

Ant Colony Optimization (ACO) is an optimization technique which is inspired by the foraging behavior of of ants. Ants do not have developed sense of sight so they do not

[1]Agha Ali Haider Qizilbash is Research Associate at Fraunhofer IPA, Stuttgart, Germany. qizilbash_ali@yahoo.com

[2]Anoj Kumar Yadav Ravensburg-Weingarten University of Applied Science anojkumar.yadav10@gmail.com

[3]Kevin Bregler is Group Leader Field Robotics at Fraunhofer IPA, Stuttgart, Germany.

[4]Werner Kraus is Department Leader Robotics and Assistive Systems at Fraunhofer IPA, Stuttgart, Germany.

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

use it for navigation. Instead they use pheromones which is a chemical substance having a very strong scent which the ants deposit while traveling in search of food and way back home and later other ants can follow the smell to trace the food. Bülent Catay et al.[1] explains the behavior of ant colonies to find food sources and also communicate with other ants by laying a chemical on the trail called pheromone. The pheromones are laid by the moving ants on the path they take. However, not every ant will take the path on which they sense the pheromone. Some ants take random paths to generate different solutions and also to escape the local optimality. The ants normally take the path that has the higher pheromone level. The shorter the path, the higher the pheromone deposited on the path because the path is visited by more ants compared to a path that is longer. The pheromone deposited will evaporate with time from the paths if they are not revisited by the ants anymore.

The ACO algorithm was first introduced by Marco Dorigo in his thesis [4]. Since then this approach has been used widely in publications for solving various problems in diverse applications of science and technology. Researchers used ACO in routing problems such as the Traveling Salesman Problem (TSP) [3] and [19] and Vehicle Routing Problem (VRP) [5] and [10]. It has also been used in multi-robot task allocation and path finding problems by Kulatunga et al. [7] and Qizilbash et al. [14].

The Routing Problem is one of the most important applications of combinatorial optimization, in which the goal is to find the best routes with the constraints or demands of several geographically scattered customers/nodes, with the least total distance or minimizing the overall costs [13]. The most known routing problem is the Traveling Salesman Problem (TSP). The problem is to find the shortest route to visit all of the cities at different locations and return to the starting point. TSP is said to be a NP-Hard problem, i.e., it cannot be solved in polynomial time [16]. The extension of the common TSP with multiple vehicles is the Vehicle Routing Problem (VRP). Dantzig and Ramser [2] proposed the original version of the VRP, formulated as the "Truck Dispatching Problem," to calculate the best routes for a fleet of gasoline delivery trucks. The classical VRP has a set of delivery customers and a central depot. An extension of VRP is Capacitated Vehicle Routing Problem (CVRP) and, as the name suggests, in this problem, each vehicle in the fleet has some capacity and correspondingly each city or customer has a certain non-negative demand weight. A route for all particular vehicles need to be constructed such that the total demand from all customers or pickup locations should not exceed the vehicle capacity limit. In our paper we address Capacitated Vehicle Routing Problem with Pick and Delivery (CVRP-PD). As suggested by Min [12], we can see VRP with Simultaneous Delivery and Pick-up (VRPSDP) variant of the VRP problem where the pickup and delivery are independent to each other. Eksioglu et al. discusses other variants and sub-variants of VRP problems in detail in its Taxonomic review [6].

ACO in particular has been used to solve CVRP has previously been applied by Mazzeo et al. [11], Stodola et al. [18], Ky et al. [8] and Tan et al. [20]. ACO has also been used for cumulative capacitated vehicle routing problems where minimizing the sum of arrival times is the objective function [9]. This type of variant of CVRP is useful in applications like supply chain in disaster areas where it is important for the load to reach quickly rather than how much vehicles are traveling. other applications such as waste disposal and collection vehicles routing has given rise to compartment vehicle routing problem as in the works of Reed et al. [15]. But as we will be considering ACO to solve CVRP-PD version with an added constraint that the capacity of robots can be regained during the tour once an item is dropped. This is not mostly applicable for package delivery vehicles as in most cases vehicles are filled up at a depot location and do not need to refilled with packages again. Although in a retail setting, this is of utmost usage for mobile robots as they can choose to deliver some items to delivery locations while still continuing their tour and continuously picking more objects once some capacity is regained. This is what we call Retail based Capacitated Vehicle Routing Problem and will be using the term further on.

III. PROBLEM FORMULATION

The objective function for this problem will be minimum total distance traveled by all fleet of robots. Also being optimized here would be number of vehicles. The constraints for the problem are mentioned as following:

- The current capacity of the robot should not exceed total load on the robot.
- Each pickup node is visited by the robot only once.
- Each pickup location have some weight constraint.
- Pickup and deposit of the same item should be performed by the same robot.
- All robots need to return to the pool/same station.

In figure 1, a brief illustration of the problem can be seen. On the left section of the figure, green circles represent item locations (pickup) and yellow number indicates weight of the item. These items needs to be dropped at grey triangles (depots) which is also indicated using red arrows, which directs towards fixed particular drop locations for each item. In the middle you can see the blue pentagon which represents pool of robots where the robots start from. The Capacity for the yellow robot is 12, capacity of orange robot is 15 and capacity of green robot is 15. Finally, on the right section

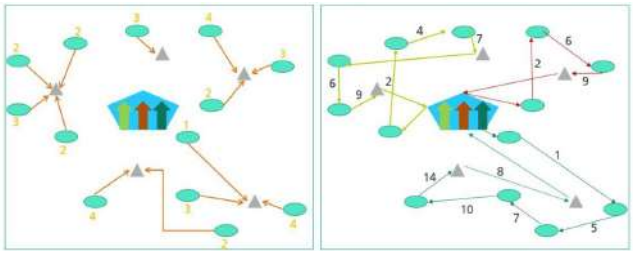


Fig. 1. Capacitated Vehicle Routing Problem with multiple pickups and multiple fixed deposit locations

you can see the routes traced for each robot in specific color along with the updated total capacity.

The fleet of robots is represented by **R**, set of all robots with equation below and M is the fleet size of the robots i.e. $\|\mathbf{R}\| = M$

$$\mathbf{R} = \{r_1, r_2, ...r_n\} \tag{1}$$

$$\mathbf{C} = \{c_1, c_2, ...c_n\} \tag{2}$$

And **C** represents set of all capacities of mobile robots: Where $c_1 \epsilon r_1$, $c_2 \epsilon r_2$ and $c_n \epsilon r_n$

Each of the pickup nodes has a certain demand or weight. Pickup nodes are represented by **P** and deposit nodes are represented by **D**. Total number of delivery nodes is represented by $\|\mathbf{D}\| = N$.

$$\mathbf{P} = \{p_1, p_2, p_3...p_n\} \tag{3}$$

$$\mathbf{D} = \{d_1, d_2, ...d_n\} \tag{4}$$

D´ represents list of pairs of pickup and deposit locations, corresponding to the directed edges in the figure 1.

$$\mathbf{D'} = [[p_1, d_1], [p_2, d_2], ...[p_n, d_n]] \tag{5}$$

In equation 5, p_n is the list of all pickup locations ϵP, which is to delivered to particular deposit locations d_n and $n \epsilon [1, 2, 3, ...N]$. l_i represents the demand of item to be picked from the list $p_n, p_n \epsilon P$. $d_i j$ represents the distance from the location i to j, where $i, j \epsilon D$. for the distance calculation refer to Implementation. And **W** represents weight of items corresponding to pickup locations such that p_i for $p_i \epsilon \mathbf{P}$. And as already mentioned in problem formulation, there would be multiple pickup nodes corresponding to each deposit location.

$$\mathbf{W} = \{w_1, w_2, w_3, ...w_n\} \tag{6}$$

Where $w_1 \epsilon p_1$, $w_2 \epsilon p_2$ and $w_n \epsilon p_n$

Further modeling involves to ensure one node is visited exactly once, pickup and delivery of a request placed is carried out by the same robot or the same route constructed. Robots are also ensured to start from pool station and come back to the same station once the order is completed and that no robot exceeds its capacity.

IV. IMPLEMENTATION

As ACO is a population-based algorithm, there is an initial parameter called "colony size." This parameter represents the size of the colony, i.e. how many solutions are to be built in each iteration of the algorithm. There is another important initial parameter called the "step size" of the algorithm. This parameter tells us how many iterations the algorithm should run. In each step, there are a number of solutions equal to the colony size. In ACO algorithm, based on the initial condition, the first solution is built, which is not the desired optimal solution. The best solution is found as iteration proceeds, based on heuristics and pheromone level on each edge. The ACO algorithm mimics the simple rules followed by the ants to find the food source. Initially, each artificial ant randomly generates paths based on the initial parameters. As the algorithm proceeds, ants generate the new solution based on the amount of the pheromone deposited on each edge. The algorithm selects the next node to visit, i.e., (i+1) from the node (i) based on the probabilistic function. Once an individual ant in the colony has constructed a tour, pheromones are updated on that tour, called local pheromone update. The edge that is taken by more ants in their tour has a high pheromone level. A global pheromone rule is applied in the algorithm to avoid the algorithm convergent towards a sub-optimal region. This process is done by evaporating the level pheromone after every step of the algorithm, i.e. when all the ants in the colony have constructed the tour. In other words, the pheromone intensity on an edge that is no longer taken by the ants will decrease with time unless it's revisited by the ants again.

The optimization problem is transformed into the problem of finding the best or shortest path on a weighted graph represented by arcs. The length of the arc is the distance between the two nodes or cities, of the arc. The artificial ants, in our algorithm, build solutions incrementally based on artificial pheromones and heuristic distance. For simplicity, this distance is calculated based on euclidean distance between the two cities which is given by equation 7. While testing the scenario, this distance can easily be replaced in the algorithm by the global plan distance calculated by the robots using the simulation environment.

$$d_{i_j} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{7}$$

In our approach, we will not directly use the probability function to find the next best fit node, but rather use the roulette wheel selection process. The roulette wheel selection process generates randomness to prevent the solution to get stuck in the local optimum. In the literature, the roulette wheel selection process for the nearest node is used in the genetic algorithm to solve the VRP problem. In this paper, we use this approach to construct the tour. The selection process will make use of inverse distance between two nodes and pheromones present on the edge. Naturally, the pheromone evaporates over time as a result, reducing the probability of other ants taking the path. The evaporation of pheromone is directly proportional to the distance, i.e. the longer the distance, the greater the pheromone evaporation. Therefore, the shorter path has a higher pheromone level. So in our approach we used the pheromone update process at each step size. We have used $(1 - \rho)$ as a factor to control the pheromone update, where ρ is pheromone evaporation constant. As we are dealing with capacity constraints of items as well so we make sure at every selection that total load on the robot does not exceed the maximum load a robot can carry. The flow of algorithm can be understood using the pseudo-code in algorithm 1.

V. EXPERIMENTATION

In this section, we perform multiple experiments to solve the pickup and delivery problem in retail application for a fleet of mobile robots with different capacities. These experiments are performed for different order sizes and different

Algorithm 1 CVRP-PD with ACO

```
CVRP (ColonySize, StepSize, α,β, RobotParameters,
Nodes, PickDropList)
SequenceList = (RobotParameters)
for each Robot in SequenceList do
    for each Step in StepSize do
        for each Ant in ColonySize do
            while RobotCapacity ≥ weight of pickup do
                Node = SelectNode (TourList, α, β )
                TourList.Append(Node)
                AddPheromone(Tour, Distance)
                Update.RobotCapacity(TourList.ItemDrop)
            end while
            GlobalPheromone(All Nodes)
        end for
      return BestTourist, BestTotalDistance for each Robot
    end for
end for
```

scenarios where there are multiple delivery locations and for different capacities of the robots. For the setup of these experiments, an order list is created which has pickup and delivery requests. In the order list coordinates of pickup and delivery is specified (x, y), the demand or weight of an item to be picked is assigned at random between 1 to 5 and as exception item number 2 is assigned 15. There can be single or multiple items to be dropped at delivery location. Since we are dealing with mobile robots with different capacities, the size of the fleet and each robot capacities in the fleet is specified as the initial parameter of the algorithm. ACO algorithm, as discussed previously, has its own parameters such as colony size, step size, alpha and beta for the probabilistic function, rate of evaporation of pheromone, initial pheromone level and distance matrix of robot pool station, pickup points and delivery points. In experiments we will discuss the results of different order list length, and varying parameters of ACO algorithm.

A. Experiment 1

In this experiment the algorithm constructs the tour for order list of size equal to 9 as shown in table I. In this experiment the robot capacity is not regained after dropping the picked items with certain load. For the experiment in consideration, the Number of robots used is **3**, the step size is **50** and colony size is **5**. Weight capacities of the first robot is **15**, second robot is **20** and third robot is **30**.

Results of the first experiment can be seen from the figure 2. The number of robots used were **2** and the Total distance traveled was **307.9**. This is one of the simplistic experiments

TABLE I
ORDER LIST FOR EXPERIMENT 1

Pickup Nodes	Delivery Nodes	Pickup and Delivery list
[1,2,3,4,8,9]	[5,6,7]	[1,9,7],[2,3,8,6],[4,5]

TABLE II
ORDER LIST FOR EXPERIMENT 2, 3 AND 4

Pickup Nodes	Delivery Nodes	Pickup and Delivery list
[1,2,3,4,8,9,11,12,13,..28]	[5,6,7,10]	[26,22,19,17,16,14,**5**], [25,21,20,11,8,3,2,**6**], [24,23,15,13,9,1,**7**], [28,27,18,14,12,**10**]

with very manageable number of pickups for each drop off so the figure 2 is easy to understand and one can trace robots easily. In this figure 2, first robot is in blue color trace and starts with 0 and continues to 4 to pickup the item, continues to 9 and 1 for similar action and then proceeds to 7 for drop off items picked from 1 and 9 and then proceeds to drop the rest to drop location 5. Similarly we can see robot 2 in orange trace also starting from 0 and proceeding to 2, 3 and 8 and then proceeding to 6 for drop off and finally returning to starting point.

B. Experiment 2

In this experiment the algorithm constructs the tour for order list of size equal to 28 as shown in table II. In this experiment the robot capacity is not regained after dropping the picked items with certain load. The number of robots used along with the step size and colony size are same as in the first experiment also specified subsection V-A. Weight capacities of each robot are same as the first experiment also specified in subsection V-A.

Results of the second experiment can be seen from the figure 3. The number of robots used were **3** and total distance traveled was **1025.16**. But in this experiment some of the pickup requests were **not completed** because the demand weights of items exceeded the total robot capacities constraints in total. In this experiment, we did not allow robot to regain capacity and, therefore, robots had no choice to deliver the ones they already picked and leave some items unpicked. Therefore, in next experiments we allow for the robot capacity to be regained when items are deposited at delivery locations during the tour.

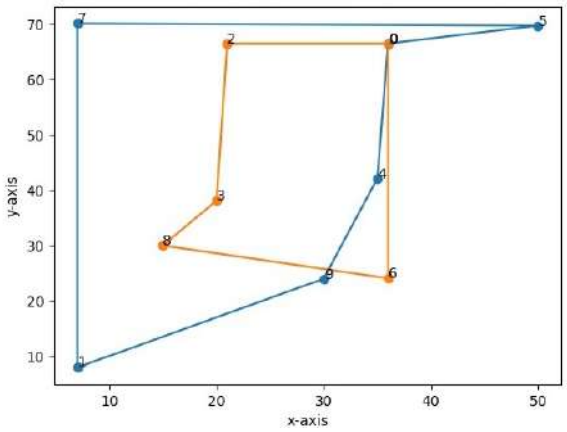


Fig. 2. Tours constructed for the robots in Experiment 1

C. Experiment 3

In this experiment the algorithm constructs the tour for order list of size equal to 28, similar to previous experiment, as shown in table II. In this experiment the robot capacity is regained once the item is dropped i.e. the weight or demand of the items delivered is removed from the robot carrying the total load. Number of robots used along with the step size and colony size along with Weight capacities of each robot are same as in the previous experiment also specified subsection V-A.

Results of the third experiment can be seen from figure 4. The number of robots used were **3** and Total distance traveled was **1034.37**. In this experiment, all pickup requests were completed **successfully** as robots were regaining capacities after dropping items to the respective delivery locations.

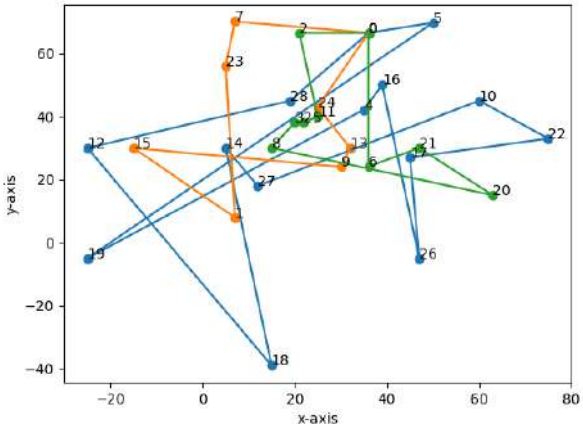


Fig. 4. Tours constructed for the robots in Experiment 3

D. Experiment 4

In this experiment the algorithm constructs the tour for order list of size equal to 28, similar to previous experiment, as shown in table II. In this experiment the robot capacity is again regained once the item is dropped i.e. the weight or demand of the items delivered is removed from the robot carrying the total load. Number of robots used along with the step size and colony size are same as in the previous experiment also specified subsection V-A. For this experiment, weight capacities of the first robot is **15**, second robot is **28** and third robot is **15**.

Results of the forth experiment can be seen from the figure 5. The number of robots used were only **2** and total distance traveled was **932.22**. In this experiment, all pickup requests were completed **successfully** and it was observed that in comparison with experiment 3, this experiment was completed by using only 2 robots and the total distance traveled by the robots is less than experiment 3.

VI. Evaluation

We observed after multiple experiments that sequence of robots also impacts the tours of the robot. Reason for this is that the algorithm iteratively generates the tour for each robot in the given robot sequence and, therefore, is biased towards the sequence of robots. It also makes it easy to get stuck in local minima which is a common ACO problem. In order to solve this, we propose two methods. First solution is sorting, where we sort the sequence of robots in descending order of capacities. It will make sure that the robot with bigger capacity is assigned to pickup more items and hence, reducing the total distance traveled by all robots. Second method we propose is shuffling of the robot sequence and running the algorithm for all combinations of fleet sequences and returning the minimum total distance tour among these combinations. This will ensure algorithm not to get stuck in local minima. The con of this approach is that computation time will increase by the factorial of fleet size.

We tested the shuffling approach with an experiment keeping all the parameters same except that we shuffled robots in the fleet according in all combinations of capacities as we can see in table III. We can see that the two best tour results we achieved (row 4 and row 7 on table III), were from the sequences where largest capacity robot was first in the sequence. This validates our initial method of sorting as well. It can also be observed that the two best results use only one robot to complete the tour whereas in other sequences at-least two robots were used.

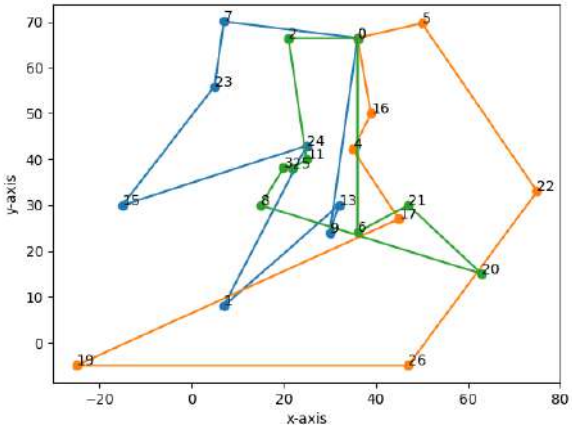


Fig. 3. Tours constructed for the robots in Experiment 2

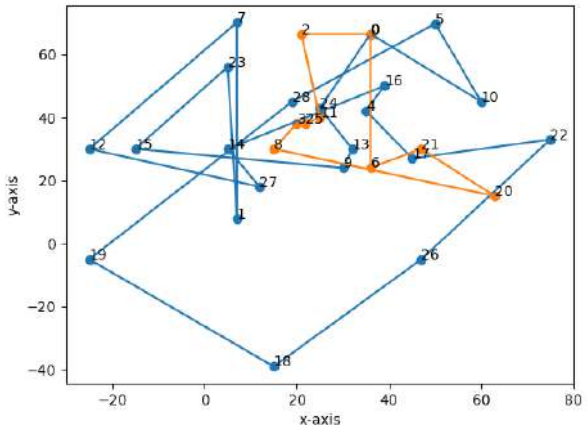


Fig. 5. Tours constructed for the robots in Experiment 4

TABLE III
SHUFFLING ROBOT CAPACITIES METHOD

R1	R2	R3	Total distance traveled	Total robots used
20	15	30	1031.85	2
15	30	20	964.42	2
30	15	20	731.6	1
20	30	15	938.08	2
15	20	30	1027.45	3
30	20	15	727.88	1

VII. CONCLUSION

In this paper, we propose a method derived from Ant Colony Optimization(ACO) to solve for retail based Capacitated Vehicle Routing Problem with Pickup and Delivery (CVRP-PD) for mobile robots. The method has been used in multiple experiments and has been successful. The method is also evaluated and has common ACO short-comings, of which improvements have been suggested and initial tests have been conducted. As a future work, we would like to work upon decentralized routing of the mobile robots using parallel computing.

REFERENCES

[1] Bülent Çatay. "Ant Colony Optimization and Its Application to the Vehicle Routing Problem with Pickups and Deliveries". In: *Natural Intelligence for Scheduling, Planning and Packing Problems*. Springer Berlin Heidelberg, 2009, pp. 219–244. ISBN: 978-3-642-04039-9.

[2] G. B. Dantzig and J. H. Ramser. "The Truck Dispatching Problem". In: *Management Science* 6 (1959), pp. 80–91.

[3] M. Dorigo and L.M. Gambardella. "Ant colony system: a cooperative learning approach to the traveling salesman problem". In: *IEEE Transactions on Evolutionary Computation* 1 (1997), pp. 53–66.

[4] Marco Dorigo. "Optimization, Learning and Natural Algorithms". PhD thesis. Politecnico di Milano, 1992.

[5] Marco Dorigo and Thomas Stützle. "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances". In: vol. 37. 2006, pp. 250–285.

[6] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. "The vehicle routing problem: A taxonomic review". In: *Computers Industrial Engineering* 57 (2009), pp. 1472–1483. ISSN: 0360-8352.

[7] A. K. Kulatunga et al. "Ant Colony Optimization based Simultaneous Task Allocation and Path Planning of Autonomous Vehicles". In: *2006 IEEE Conference on Cybernetics and Intelligent Systems*. 2006, pp. 1–6.

[8] Phan Nguyen Ky Phuc and Nguyen Le Phuong Thao. "Ant Colony Optimization for Multiple Pickup and Multiple Delivery Vehicle Routing Problem with Time Window and Heterogeneous Fleets". In: *Logistics* 5 (2021). ISSN: 2305-6290.

[9] Nikolaos A Kyriakakis, Magdalene Marinaki, and Yannis Marinakis. "A hybrid ant colony optimization-variable neighborhood descent approach for the cumulative capacitated vehicle routing problem". In: *Computers Operations Research* 134 (2021), p. 105397. ISSN: 0305-0548.

[10] Andrea Lodi, Silvano Martello, and Daniele Vigo. "Neighborhood Search Algorithm for the Guillotine Non-Oriented Two-Dimensional Bin Packing Problem". In: *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Springer US, 1999, pp. 125–139. ISBN: 978-1-4615-5775-3.

[11] Silvia Mazzeo and Irene Loiseau. "An Ant Colony Algorithm for the Capacitated Vehicle Routing". In: *Electronic Notes in Discrete Mathematics* 18 (2004), pp. 181–186. ISSN: 1571-0653.

[12] Hokey Min. "The multiple vehicle routing problem with simultaneous delivery and pick-up points". In: *Transportation Research Part A: General* 23 (1989), pp. 377–386. ISSN: 0191-2607.

[13] Jean-Yves Potvin. "A Review of Bio-inspired Algorithms for Vehicle Routing". In: *Bio-inspired Algorithms for the Vehicle Routing Problem*. Springer Berlin Heidelberg, 2009, pp. 1–34. ISBN: 978-3-540-85152-3.

[14] Agha Ali Haider Qizilbash, Christian Henkel, and Sanaz Mostaghim. "Ant Colony Optimization based Multi-Robot Planner for Combined Task Allocation and Path Finding". In: *2020 17th International Conference on Ubiquitous Robots (UR)*. 2020, pp. 487–493.

[15] Martin Reed, Aliki Yiannakou, and Roxanne Evering. "An ant colony algorithm for the multi-compartment vehicle routing problem". In: *Applied Soft Computing* 15 (2014), pp. 169–176. ISSN: 1568-4946.

[16] Roberto Roberti and Paolo Toth. "Models and Algorithms for the Asymmetric Traveling Salesman Problem: an Experimental Comparison". In: *Journal of Mechanical Systems for Transportation and Logistics* 1 (2012), pp. 113–133.

[17] Lorenzo Sabattini et al. "Technological roadmap to boost the introduction of AGVs in industrial applications". In: 2013, pp. 203–208. ISBN: 978-1-4799-1493-7.

[18] Petr Stodola et al. "Using the Ant Colony Optimization algorithm for the Capacitated Vehicle Routing Problem". In: *Proceedings of the 16th International Conference on Mechatronics - Mechatronika 2014*. 2014, pp. 503–510.

[19] Thomas Stützle and Holger H. Hoos. "MAX-MIN Ant System". In: *Future Gener. Comput. Syst.* 16 (2000), pp. 889–914.

[20] zhong Yang and Baozhen Yao. "An Improved Ant Colony Optimization for Vehicle Routing Problem". In: *European Journal of Operational Research* 196 (2009), pp. 171–176.

Direct Object Reconstruction on RGB-D Images in Cluttered Environment

Mikołaj Zieliński[1], Bartłomiej Kulecki[1], Dominik Belter[1]

***Abstract*— Robots have limited perception capabilities when observing the scene from a single viewpoint. Some objects on the scene might be partially occluded and their 3D shape is not fully available to the robot. Existing methods obtain object models through a series of observations using RGB-D sensors or the robot is trained to operate in the presence of occlusions. In this paper, we directly address object reconstruction in the presence of occlusions. We propose an image generation approach using a neural network architecture to remove occluding objects and other objects from RGB-D images and reconstruct the occluded object that the robot is interested in. The proposed method utilizes a cascade of neural networks trained to progressively remove occlusions and reconstruct the RGB-D images of the scene.**

I. Introduction

Objects pose estimation [1], grasping [2], [3] and manipulation [4] is challenging due to occlusions. In the typical scenario with the autonomous robot working in an unstructured environment, the robot has to detect and estimate the pose of the selected objects using the onboard perception system (Fig. 1). Most of the perception systems of the robots operating in the indoor environment utilize RGB and depth cameras. As a result, the full scene model of the scene is impossible to obtain from a single view due to the limited observation angle and occlusions in the scene. Careful scene scanning is time-consuming and very often impossible because the robot does not have access to poses that allow scanning of the occluded parts of the scene. Similarly, when the robot grasped the object or performs in-hand manipulation, the pose of the object should be estimated from a single view to improve the performance of the robot. However, this task is challenging because of the limited view of the RGB-D cameras and the strong occlusions of the selected object by the fingers of the robot.

Typical approaches to pose estimation and grasping objects based on the machine learning focus on training the system in the cluttered environment [1], [5]. In this case, the system is trained to solve various challenging tasks like grasping, pose estimation, and simultaneously deal with occlusions. Other approaches assume that the main task is performed based on a series of time-consuming observations [6], [7], [8] or the robot actively plans the sequence of actions and interacts with objects to remove occluding objects [9], [10]. However, these approaches will not help if the robot has to estimate the pose of the object in the hand occluded by the fingers.

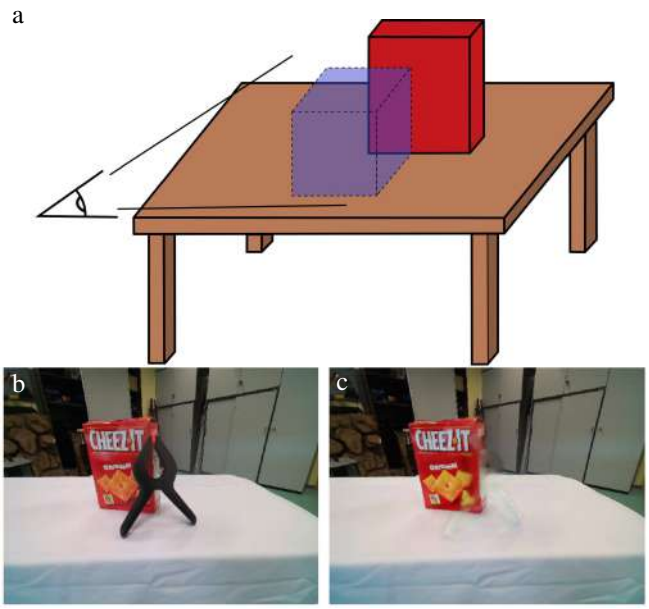


Fig. 1. Example application scenario of the proposed method (a). The goal of the proposed method is to remove the occluding object (black spring clamp) and other objects from the scene (b) and simultaneously reconstruct the occluded object (cracker box) (c) to improve the performance of robots operating in a cluttered environment.

In this research, we focus on direct scene reconstruction. However, instead of directly reconstructing the 3D layout of the scene and the 3D shape of the objects [11], [12], [13], we formulate the problem as an image generation task. In this task, the robot observes the scene using an RGB-D camera (Fig. 1b). The images contain the partially registered object, that is occluded by other objects on the scene or fingers of the robot during grasping (Fig. 1a). The goal of the proposed system is to generate RGB and depth images of the scene and remove objects from the images except for the object that the robot is interested in. The reconstructed images can be later used by the pose estimation framework [1] or during grasping objects [14].

Inspired by the methods that remove small objects from the images e.g. raindrops removal from the windshield of a car [15], we propose a neural network architecture that removes unwanted objects from the RGB-D images. Because the scene reconstruction problem is formulated in the 2-dimensional space, we utilize standard encoder-decoder architecture. The main challenge of this task is related to the size of the removed objects. Reconstructing large parts of

The work was supported by the National Science Centre, Poland, under research project no UMO-2019/35/D/ST6/03959.

[1]Authors are with Institute of Robotics and Machine Intelligence, Poznan University of Technology, 60-965 Poznań, Poland name.surname@put.poznan.pl

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

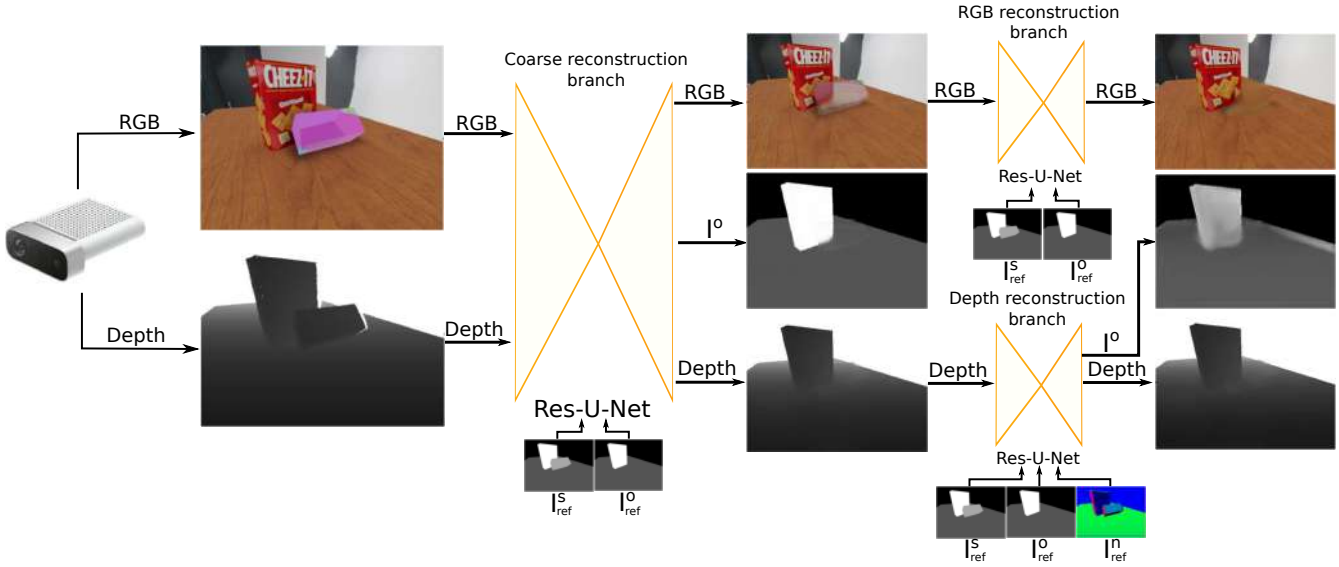


Fig. 2. Architecture of the proposed object reconstruction system on RGB-D images. The proposed system consists of three U-Net-based neural networks that gradually remove the occluding objects and other objects from the scene and simultaneously reconstruct the occluded object.

the images is significantly more challenging than removing small objects like raindrops. Thus, we focus on training a cascade of neural networks that gradually remove objects and reconstruct the images of the scene.

II. Related work

The primary goal of the proposed method is to reconstruct the images with the occluded object on the scene. The proposed solution allows "seeing through" occluding objects and reconstructing the RGB-D images of the occluded object. The problem that we formulate in this paper is different from problems commonly presented in the literature like unwanted objects removal from the images [15] or direct scene reconstruction [11], [12], [13]. The methods that remove the objects from the images are based on image segmentation with convolutional neural networks [16] or visual transformers [17]. Then the holes in the images are reconstructed using in-painting methods [18]. These approaches assume that we know the category of occluding objects. In this research, we assume that only the occluded object is known and this object can be occluded by any other type of object. For this reason, we can not directly utilize image segmentation and in-painting methods in this task. Also, the popular problem of raindrop removal is slightly different because the objects that are removed from the image are in the same category and they are small with respect to the image [15].

The proposed method reconstructs the depth and RGB image of the selected object. Most of the methods that reconstruct the scene operate directly in the 3D space. The methods utilize 3D convolutions for shape completion of the objects [13]. The methods that operate directly in 3D space are computationally-expensive and in some cases, the inference may take several minutes even if standard fully Mulitilayer Perceptron is applied to infer about the occupancy of the 3D space [19]. Another scene and object reconstruction method tries to match the 3D object CAD models from the database [20] to the observed scene. This approach requires estimating the poses of the objects on the scene. In contrast, the goal of our method is to directly reconstruct the RGB-D images. Thus, our method utilizes techniques that are popular in view-dependent scene reconstruction [21], [11]. On the other hand, our method does not reconstruct a full 3D model of the object.

The method from computer vision that segment the images of the 3D scene and represent them as a Layered Depth Image (LDI) is used in [22]. These methods utilize semantic segmentation. The method determines the spatial relation between layers and with this information, it is possible to remove the objects from the scene. The LDI determines which objects are closer and which ones are farther from the camera. Our method also segments the images but this information is used during training only to improve the convergence of the training. In contrast to our method, the LDI does not reconstruct the parts of the objects that are occluded on the images.

A. Approach and Contribution

In this paper, we propose a method that removes the objects from the images and leaves and reconstructs the surface and the texture of the object that is related to the task. The main contributions of this paper include the following:

1) view-dependent definition of the scene reconstruction problem that efficiently takes advantage of 2D convolutions,
2) training loss formulation and a new architecture of the neural network that utilizes three branches to remove objects and reconstruct the scene,
3) experimental verification of the proposed method on the images from the Kinect Azure and Asus Xtion Pro Live.

Fig. 3. Example random training scenes with the objects from the YCB dataset: cracker box, a mustard bottle, tomato soup can, power drill, and extra large clamp.

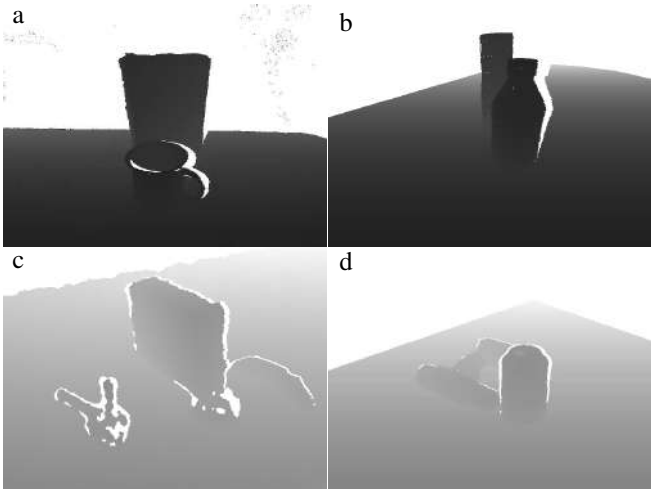


Fig. 4. Comparison between real depth images from Kinect Azure (a), Asus Xtion Pro Live (c), and the example depth images from the training set with artifacts generated using RGB-D camera model for Kinect Azure (b) and Asus Xtion Pro Live (d).

III. Objects Removal and Object Reconstruction

The main task of the proposed neural architecture is to remove the objects from the scene and reconstruct the occluded parts of the object that the robot is interested in. In contrast to neural networks that remove small and known objects from the images, the end-to-end solution based on the U-Net-like architectures does provide accurate results. Instead, we propose a cascade of neural networks that gradually reconstruct the occluded objects on the RGB-D images. The approach of training a cascade of separate modules has many advantages. We can train each subsystem independently using different loss functions. This strategy allows each module to learn a given task better than in end-to-end solutions because instead of training one network to solve one complex problem, we train neural networks to solve much simpler tasks.

The architecture of the proposed neural network is presented in Fig. 2. The proposed system consists of three main modules: coarse RGB-D reconstruction, accurate color image reconstruction, and accurate depth image reconstruction. Each of these subsystems is based on deep neural networks with a U-Net-type architecture [23] with the ResNet-34 encoder, named Res-U-Net, pre-trained on the ImageNet dataset. The coarse construction module takes a pair of RGB-D images as input with the reconstructed object occluded by other objects. The task of this branch is to initially remove the occluding objects, estimate the geometry of the reconstructed object, and initially recover the texture of the object. The generated RGB images are provided to the input of the accurate color image reconstruction branch. It aims to improve the quality of estimation of the actual appearance of the reconstructed object. In parallel, the depth images go to the input of the depth image reconstruction branch. This subsystem is designed to improve the quality of depth image reconstruction generated by the coarse processing module.

A. Training

Each neural network from the architecture presented in Fig. 2 is trained independently. The loss functions play a crucial role in the training process. We use the reference RGB image on the output with removed objects $\mathbf{I}_{\text{ref}}^{\text{RGB}}$, and corresponding depth image $\mathbf{I}_{\text{ref}}^{\text{d}}$, scene segmentation image $\mathbf{I}_{\text{ref}}^{\text{s}}$, scene segmentation image with removed objects $\mathbf{I}_{\text{ref}}^{\text{o}}$, and image that contains normal vectors to the surfaces of the objects $\mathbf{I}_{\text{ref}}^{\text{n}}$. Because we found the structural similarity index measure (SSIM) sensitive to local minima, we utilize the Mean Squared Error (MSE) to define the loss function for training each neural network:

$$MSE^{\epsilon} = e^{\epsilon} = \frac{1}{n}\sum_{i=i}^{n}(p_i - \hat{p}_i)^2, \tag{1}$$

where ϵ is related to the type of reference image and p_i is the i-th pixel of the image used to compute the loss value.

Even though we do not focus on segmentation, the coarse reconstruction branch returns also a segmentation image. In this case, we utilize the strategy defined in [24] that jointly trains multiple images on the output of the CNN to improve the consistency of the results compared to separate learning. The first branch of the CNN is trained using the following loss function $\mathcal{L}_c$:

$$\mathcal{L}_c = a_1 \cdot e^{RGB} + a_2 \cdot e^{d} + a_3 \cdot e^{s} + a_4 \cdot e^{r} + a_5 \cdot e^{u} + a_6 \cdot e^{o}, \tag{2}$$

where e^{RGB}, e^{d}, e^{s}, e^{r}, e^{u}, and e^{o} are MSE values computed for the RGB, depth, segmentation, depth of the reconstructed object, depth of the union of removed and reconstructed objects, and depth of the removed objects, respectively. The coefficients are set to $\mathbf{a} = [2, 1, 1, 1, 5, 5]$ to increase the role of the objects in the image with respect to the whole image.

The RGB reconstruction branch that returns the reconstructed RGB image and takes the output RGB image from the first branch is trained using the following loss function:

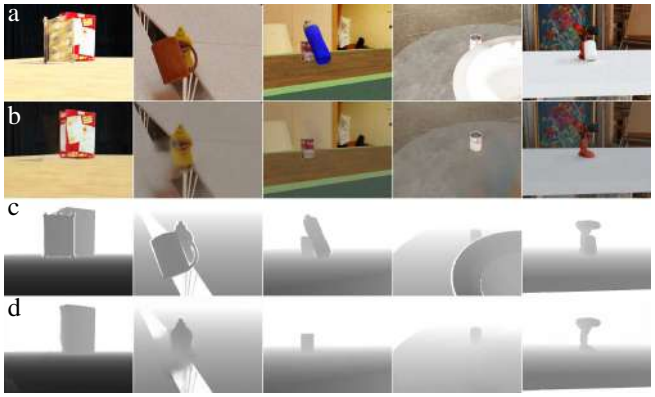


Fig. 5. Example reconstruction results of various objects on the synthetic dataset: input RGB (a), output RGB (b), input depth (c), and output depth images (d).

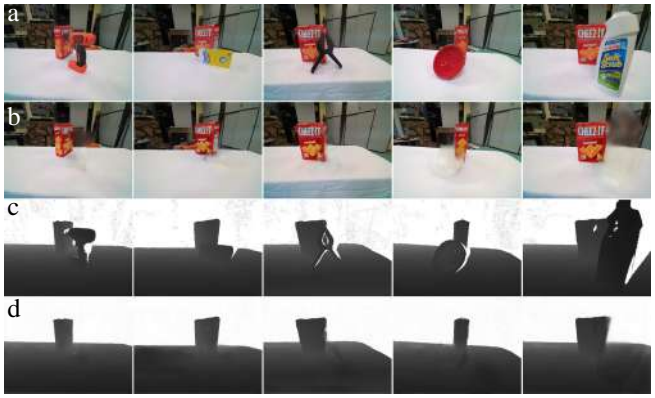


Fig. 6. Example reconstruction results of the `cracker box` on the images from the Kinect Azure: input RGB (a), output RGB (b), input depth (c), and output depth images (d).

$$\mathcal{L}_{\text{RGB}} = b_1 \cdot e^{RGB} + b_2 \cdot e^r + b_3 \cdot e^u + b_4 \cdot e^o, \quad (3)$$

where e^{RGB}, e^r, e^u, and e^o are MSE values computed for the RGB image, RGB image of the reconstructed object, RGB image of the union of all objects, and RGB image of the removed objects, respectively. The coefficients are set to $\mathbf{b} = [2, 1, 1, 1]$. The last branch that reconstructs the depth image and takes the depth image from the output of the coarse reconstruction branch is trained using the following loss function:

$$\mathcal{L}_{\text{d}} = c_1 \cdot e^s + c_2 \cdot e^r + c_3 \cdot e^u + c_4 \cdot e^o + c_5 \cdot e^n, \quad (4)$$

where e^s, e^r, e^u, e^o, and e^n are MSE values computed for the segmentation image, the depth image of the reconstructed object, the depth image of the union of all objects, a depth image of the removed objects, and image of the normal vectors, respectively. The coefficients are set to $\mathbf{c} = [1, 1, 20, 20, 0.1]$. Again, we found that forcing the neural network to return the segmentation image and using the normal vectors during training even though the system does not utilize these values improves the convergence of the training and consistency of the results.

We train the three neural networks for 14/14/6 epochs using the Adam optimizer. The obtained neural network reconstructs a specific instance of the object. To reconstruct various instances, we trained multiple CNNs.

B. Dataset

Learning a neural network using real data gives the best results. In this case, training and inference are performed on data that have similar characteristics. However, generating real training data for computer vision tasks is time-consuming. For this reason, a synthetic data generator is used in this work. The advantage of a synthetic dataset generator is the ability to quickly generate new data in a fully controlled environment. On the other hand, it is difficult to guarantee the same properties of the generated and real images. This can lead to the neural network not being capable to operate on real images. In this research, we use the Blender environment and objects from the YCB [25] and shapeNet [26] datasets to generate the training dataset. Moreover, we actively modify the camera pose and configuration of the objects on the scene to set the reference occlusion of the object. With this strategy, we can uniformly generate the samples with the occlusions changing from 10 to 90%.

We train our system to operate with Kinect Azure and Asus Xtion Pro Live RGB-D cameras. We found that the system performs significantly better if we model the properties of the depth images in the dataset generator. We mimic a strategy similar to the method presented in [27] to generate depth images for the Asus Xtion camera (Fig. 4). This model does not fit the images returned by the Kinect Azure because these sensors use different techniques to measure the distance to objects. We noted that the depth images from the Kinect Azure do not have data on the right side of the objects. We conclude that this phenomenon comes from the fact that the RGB and depth cameras are shifted, and the depth measurements are transformed to the RGB camera frame to have a direct correspondence between the RGB and depth pixels. We simulate the shift between RGB and depth cameras to mimic this phenomenon during generating the training samples. We transform the depth measurements into 3D space and reproject them back on the RGB camera frame. The results of camera modeling are presented in Fig. 4.

To show the results on the real images, we utilize the YCB video dataset that operates on the Asus Xtion Pro Live images [25]. To show the result on the Kinect Azure camera we collected 100 RGB-D images of the scenes containing YCB objects (cracker box) occluded by other objects[1]. We also registered the ground-truth images in the real environment by removing the occluding objects from the scene. These images are later used for qualitative and quantitative evaluation.

[1]The dataset is available at `https://drive.google.com/file/d/15zJp1F9ZQGhIRiyqc2pgAGCo1x1sQyWi`

Fig. 7. Example reconstruction results of the `cracker box` on the images from the Asus Xtion Pro Live (YCB video dataset): input RGB (a), output RGB (b), input depth (c), and output depth images (d).

TABLE I

MAE VALUE OBTAINED FOR THE OBJECTS FROM THE SYNTHETIC TEST SET WITH THE ASUS XTION IMAGES.

	cracker box	mustard bottle	tomato soup can	power drill	extra large clamp
MAE_r^d [mm]	12.52	18.52	27.84	18.26	18.52
MAE_o^d [mm]	24.93	21.26	19.27	23.88	21.26
$\mathrm{MAE}_r^{\mathrm{RGB}}$	12.74	12.09	14.25	12.04	12.09
$\mathrm{MAE}_o^{\mathrm{RGB}}$	25.32	25.50	21.60	23.44	25.50

IV. RESULTS

A. Qualitative evaluation

First, we performed a qualitative evaluation of the generated images. Example results presented on the synthetic dataset and the images from the real Kinect Azure and Asus Xtion, are presented in Fig. 5, Fig. 6, and Fig. 7, respectively[2]. The trained neural network properly recovers the texture of the selected object, and at the same time removes the remaining objects. Even though we pay more attention to the reconstruction of the objects (weights in loss (2)-(4)), the neural network also reconstructs the background and the surface of the table. The texture is slightly distorted but the details are well visible. More importantly, the neural network trained on the synthetic dataset also works on the data from the real sensor (Fig. 6 and Fig. 7). Qualitative results show that the proposed method works better when the images from the Kinect Azure are used. This result is caused by the higher quality of depth images from Kinect Azure that contains smaller numbers of artifacts and provides a more accurate model of the objects on the scene.

B. Quantitative evaluation

To provide the measurable quality of reconstruction, we compute the mean absolute error for the reconstructed (MAE_r) and for removed objects (MAE_o) for the depth MAE^d and RGB images $\mathrm{MAE}^{\mathrm{RGB}}$. The MAE values for the RGB images are computed for intensity values that are in the range between 0 and 255. The results are presented in Tab. I. The results are computed for the synthetic images generated for the Asus Xtion camera model so they contain artifacts similar to those introduced by the real sensor. The obtained MAE value for reconstructed objects changes from 12.52 mm for the cracker box to 27.84 mm for the power drill. These results suggest that the neural network better reconstructs regular shapes. The reconstruction error for the occluding objects is larger for RGB and depth images. This comes from the fact, that we prioritize the reconstruction of the occluded object. Moreover, a whole removed object is replaced by the pixels belonging to the background and objects behind the removed object while the CNN reconstructs only a part of the occluded object.

[2]The video is available at `https://youtu.be/EWMauSVTShA`

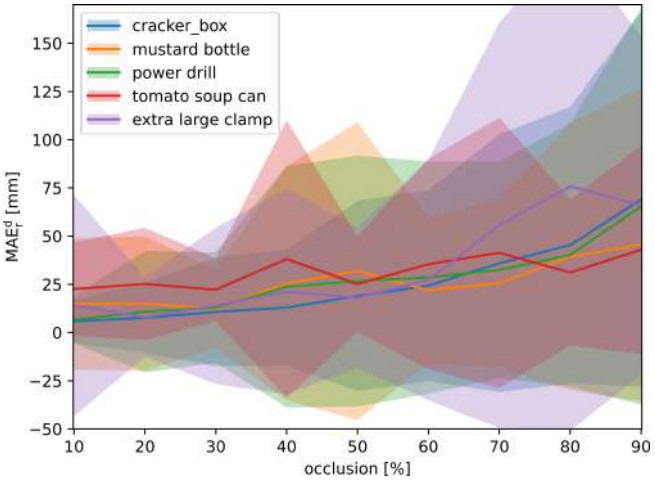


Fig. 8. Relation between the depth reconstruction error of the occluded object MAE_r^d and the percentage of the object occlusion.

We also check the dependency between the reconstruction error depends and the object occlusion. If the reconstructed object is slightly occluded, the task given to the neural network is easy. On the other hand, the reconstruction becomes more challenging when the occlusion increases. In the next experiment, we gradually increase the occlusion of five objects from 10% to 90%. The results for five objects from the YCB dataset are presented in Fig. 8. When the occlusion is small, the reconstruction error is almost three times smaller than the average errors presented in Tab. I. The presented experiments show that we can expect gradually increasing reconstruction error when the occlusion of the objects increases.

We also verified the proposed neural network on the data from the Kinect Azure (Fig. 6). The mean MAE value for the whole depth images is equal to 24.8 mm while the MAE value for RGB images is equal to 5.66. When we use the mask of the reconstructed object to compute the error, The MAE value is equal to 53.8 mm for the depth images and 19.7 for RGB images.

C. Ablation study

We performed an ablation study to justify our design choices. We have checked the popular Binary Cross Entropy

TABLE II

MEAN μ AND STANDARD DEVIATION σ OF THE RECONSTRUCTION ERROR FOR THE DEPTH $\mathbf{I}^d$ AND RGB $\mathbf{I}^{RGB}$ IMAGES OBTAINED FOR VARIOUS STRATEGIES OF TRAINING THE NEURAL NETWORK.

method	μ_r^d [mm]	σ_r^d [mm]	μ_r^{RGB} []	σ_r^{RGB} []
Binary Cross Entropy	52.67	56.04	14.25	20.09
End-to-end (2)	**12.01**	38.21	13.58	21.31
our (2)–(4)	12.52	24.93	**12.74**	25.32

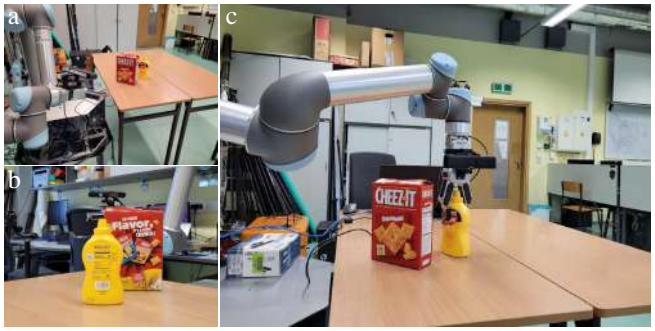


Fig. 9. Experiment with the robot grasping a mustard bottle occluded by the cracker box: initial robot configuration (a,b) and the robot grasping the object (c).

loss for evaluating the segmentation results instead of using MAE. We also verified the end-to-end approach to the problem. Because end-to-end training from scratch does not bring satisfactory results, we trained the neural network using the proposed loss function. Then, we continue training all branches of the neural network simultaneously using the proposed loss function (2). The results are presented in Tab. II. Training CNN using Binary Cross Entropy brings the worst results. These results might be caused by the unequal size of positive and negative masks on the images. Two-staged learning brings significantly better results. It even provides slightly better mean results for the depth images but with a larger standard deviation. At the same time, the proposed loss function returns good results for both RGB and depth images.

We also checked the influence of the proposed sensor model. The MAE value for the real images is equal to 29.8 mm when we use perfect synthetic data to train the neural network. The error is reduced by 17% to 24.8 mm when the model of the Kinect Azure camera is used to generate training samples [28].

D. Grasping with object reconstruction

Finally, we have performed the qualitative evaluation of the scene and object reconstruction on the real robot. In the experiment, the robot observes the scene with two objects on the table. The scene configuration is presented in Fig. 9. The robot is going to grasp the mustard bottle that is strongly occluded by the cracker box. The initial scene configuration captured by the RGB-D camera is presented in Fig. 10a. The robot utilizes the method presented in [29] to grasp the mustard bottle. To this end, the grasping method detects objects from the point cloud and determines the 3D bounding boxes by computing the covariance matrix for the points belonging to the object. Using the partial model of the object results in the incorrect reference pose of the gripper (Fig. 10b). In Fig. 10c, we compare the reconstructed mustard bottle with the initial bounding box that shows that the obtained grasp pose lies on the edge of the object and causes the grasp failure. Our method directly removes the cracker box and reconstructs the occluded mustard bottle that improves the grasp pose that lies close to the geometric center of the object (Fig. 10d). The robot grasping the mustard bottle is presented in Fig. 9c.

V. CONCLUSIONS

In this paper, we deal with the problem of the occluded objects reconstruction. We propose a method for direct removing objects from RGB-D images and reconstructing the occluded parts of the selected object to enhance the perception system of mobile-manipulating robots. We developed a novel approach for scene reconstruction that incorporates a view-dependent definition of the problem, leveraging the efficiency of 2D convolutions. We introduce a new neural network architecture and formulation of a training loss that utilizes a cascade of CNNs to effectively remove objects and reconstruct the scene. Finally, the paper presents the results of the experiments conducted to validate the proposed method using images captured from the Kinect Azure and Asus Xtion Pro Live devices. The obtained results show that we can reconstruct the given object on the scene without directly using an object detector or scene segmentation methods. The reconstruction accuracy depends on the percentage of occlusion and the average error for most of the reconstructed error is smaller than 20 mm.

The proposed method shows promising results in scene reconstruction and in the future, we are going to use the reconstructed objects to improve object detection, pose estimation, and grasping methods. We are also going to work on the neural model of the objects to separate the reconstruction task from the internal object representation stored in the CNN structure.

REFERENCES

[1] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *Robotics: Science and Systems (RSS)*, 2018.

[2] A. Saxena, L. Wong, M. Quigley, and A. Y. Ng, "A vision-based system for grasping novel objects in cluttered environments," in *Robotics Research*, M. Kaneko and Y. Nakamura, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 337–348.

[3] Y. Yu, Z. Cao, S. Liang, W. Geng, and J. Yu, "A novel vision-based grasping method under occlusion for manipulating robotic system," *IEEE Sensors Journal*, vol. 20, no. 18, pp. 10 996–11 006, 2020.

[4] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving Rubik's cube with a robot hand," *CoRR*, vol. abs/1910.07113, 2019. [Online]. Available: http://arxiv.org/abs/1910.07113

[5] W. Guo, W. Li, Z. Hu, and Z. Gan, "Few-shot instance grasping of novel objects in clutter," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6566–6573, 2022.

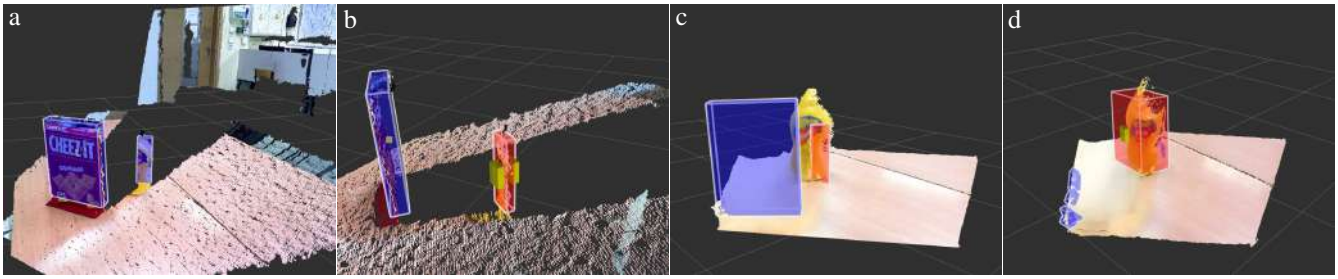


Fig. 10. Object and scene reconstruction during the experiment with the robot grasping an object (mustard bottle) occluded by the other object (cracker box): input point cloud from the sensor and 3D bounding boxes for the detected objects (a), failure position of the gripper during grasping the mustard bottle (b), reconstructed mustard bottle and removed cracker box (c), and the improved gripper pose after direct scene reconstruction (d).

[6] D.-C. Hoang, A. J. Lilienthal, and T. Stoyanov, "Object-RPE: Dense 3D reconstruction and pose estimation with convolutional neural networks," *Robotics and Autonomous Systems*, vol. 133, p. 103632, 2020.

[7] D. Son, "Grasping as inference: Reactive grasping in heavily cluttered environment," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7193–7200, 2022.

[8] A. Kanezaki, Y. Matsushita, and Y. Nishida, "Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5010–5019.

[9] Z. Liu, Z. Wang, S. Huang, J. Zhou, and J. Lu, "Ge-grasp: Efficient target-oriented grasping in dense clutter," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 1388–1395.

[10] D. Ren, X. Ren, X. Wang, S. T. Digumarti, and G. Shi, "Fast-learning grasping and pre-grasping via clutter quantization and q-map masking," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3611–3618.

[11] A. Nicastro, R. Clark, and S. Leutenegger, "X-section: Cross-section prediction for enhanced RGB-D fusion," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1517–1526.

[12] D. Shin, Z. Ren, E. Sudderth, and C. Fowlkes, "3D scene reconstruction with multi-layer depth and epipolar transformers," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2172–2182.

[13] A. Dai, C. Diller, and M. Niessner, "SG-NN: Sparse generative neural networks for self-supervised scene completion of RGB-D scans," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 846–855.

[14] M. S. Kopicki, D. Belter, and J. L. Wyatt, "Learning better generative models for dexterous, single-view grasping of novel objects," *The International Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1246–1267, 2019.

[15] S. Zini and M. Buzzelli, "Laplacian encoder-decoder network for raindrop removal," *Pattern Recognition Letters*, vol. 158, pp. 24–33, June 2022.

[16] R. Li, S. Zheng, C. Zhang, C. Duan, L. Wang, and P. M. Atkinson, "ABCNet: Attentive bilateral contextual network for efficient semantic segmentation of fine-resolution remotely sensed imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 181, pp. 84–98, 2021.

[17] A. Hatamizadeh, Z. Xu, D. Yang, W. Li, H. Roth, and D. Xu, "Unetformer: A unified vision transformer model and pre-training framework for 3d medical image segmentation," 2022. [Online]. Available: https://arxiv.org/abs/2204.00631

[18] W. Li, Z. Lin, K. Zhou, L. Qi, Y. Wang, and J. Jia, "Mat: Mask-aware transformer for large hole image inpainting," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 10 748–10 758.

[19] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 165–174.

[20] F. Engelmann, K. Rematas, B. Leibe, and V. Ferrari, "From points to multi-object 3D reconstruction," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 4588–4597.

[21] R. Staszak, B. Kulecki, W. Sempruch, and D. Belter, "What's on the other side? a single-view 3d scene reconstruction," in *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2022, pp. 173–180.

[22] Z. Jiang, B. Liu, S. Schulter, Z. Wang, and M. Chandraker, "Peek-a-boo: Occlusion reasoning in indoor scenes with plane representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[24] U. Kusupati, S. Cheng, R. Chen, and H. Su, "Normal assisted stereo depth estimation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2186–2196.

[25] B. Çalli, A. Walsman, A. Singh, S. S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols," *CoRR*, vol. abs/1502.03143, 2015. [Online]. Available: http://arxiv.org/abs/1502.03143

[26] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[27] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1524–1531.

[28] M. Zieliński and D. Belter, "On the importance of the RGB-D sensor model in the CNN-based robotic perception," in *Proc. of the 4th Polish Conference on Artificial Intelligence (PP-RAI 2023)*, 2023.

[29] B. Kulecki, K. Młodzikowski, R. Staszak, and D. Belter, "Practical aspects of detection and grasping objects by a mobile manipulating robot," *Industrial Robot: the international journal of robotics research and application*, Jan 2021. [Online]. Available: https://doi.org/10.1108/IR-10-2020-0242

Robust Perception Skills for Autonomous Elevator Operation by Mobile Robots

Steffen Müller[1], Benedict Stephan[1], Tristan Müller[1] and Horst-Michael Gross[1]

***Abstract*— Autonomous mobile service robots with transportation tasks are often restricted to work on a single floor, since remote access to elevators is expensive to integrate for reasons of safety certification. Therefore, already ten years ago first robots have been enabled to use the human interface for riding an elevator. This requires a variety of perception and manipulation capabilities as well as social skills when it comes to interaction with other people who want to use the elevator too. We summarize the progress in solving the specific tasks of detecting and localizing the required buttons to press robustly. A deep-learning approach for detecting buttons in images is combined with a verification based on predefined knowledge on button arrangements in the elevator's control panels. Also perception of the elevator's state and our realization of the robot's elevator riding capabilities are discussed.**

I. INTRODUCTION

Riding an elevator is a subconscious action for human beings, even if there is a lot of time to think about it while waiting for the cabin. For a mobile service robot in contrast the process has to be decomposed into clearly defined steps each requiring specific recognition and articulation skills.

To reach a destination on another floor, a robot must first implement a path planner capable of managing multiple floors. Already in 2007 Kang et al. [1] described navigation skills necessary to plan on multi-story maps. In general, a combination of topological and metric planning allows to solve the planning problem [2].

Once the plan shows that a transition between floors is needed, the robot can go to the lift on its current floor, and then the elevator procedure starts.

There exists research about robots without manipulation skills, which need the assistance of humans for operating the lift. [3] as well as [4] described how a robot seeks a helping hand and analyzes the people's intent in helping the robot. Other social studies [5], [6] analyze the effect of a robot's appearance and interaction behavior in a conflict situation while waiting for an elevator.

On the technical side, an autonomous robot first needs to call the elevator by pressing the respective button. For that purpose, a robust detection and localization of the button in the 3D operational area of the robot is needed. In the early days, this has been achieved using classical approaches in image space. For example SIFT-feature-based detection or template matching [7] have been used. Recent solutions rely on neural-network-based detectors. Zhu et al. [8] trained a Faster RCNN detector on elevator buttons. In order to also handle situations with unconventional buttons, they combined the Faster RCNN-based region of interest (ROI) detection with optical character recognition (OCR) to read the button labels. Once a 2D proposal for a button location exists, the utilization of depth data either from 3D cameras or LIDAR sensors can be used to project the image position into 3D space. A button detection with a neural network in combination with a LIDAR for 3D coordinate transform can be found in [9].

The detection of individual buttons alone may fail or the wrong label might be predicted in a real world application where partial occlusions or other artifacts may disturb the perception. Therefore, the geometry and arrangement of the complete control panel has been used to disambiguate the meaning and position of individual buttons. Abdulla et al. [10] for example detect button panels and correct the exact location by using artificial landmarks at the corners of the panel. We adopted the idea of using knowledge of the complete control panel to make button localization more resilient.

Assuming that the correct 3D coordinate of the desired button could be found, the next step is to push the button. This is a straight forward task utilizing the motion planner for the robotic arm. To decide if the cabin has arrived and whether the robot can safely enter the lift, further perception skills are necessary. Lee et al. [11] recognize the state of the elevator by laser, and a neural-network-based classification of the arrow signs is used to confirm success of the push button action. Once the cabin is entered, the goal floor has to be selected by means of another button operation. Then the robot has to recognize the correct floor to leave the lift. This can either be done by means of artificial landmarks for localization on the goal floor [11], or the robot can rely on acceleration sensor tracking [12] and reading the information panel of the elevator [13]. In all cases, the robot needs to be specialized on the present elevator, or the environment has to be adapted to the robot's needs.

In our work, we tried to avoid any modification to the environment and used pragmatic solutions for controlling the whole sequence only with feedback on the lift state extracted from the SICK laser range scanner of our robot. By reducing the number of different image-based detection systems to a minimum, the potential points of failure should decrease as well, which makes the whole system more robust.

In Literature typically only solutions for the individual

*This research has received funding from the thuringian project of innovation potentials as part of the thurAI project (grant agreement 2021 FGI 0008).

[1]Authors are with the Neuroinformatics and Cognitive Robotics Lab, Technische Universität Ilmenau, 98693 Ilmenau, Germany steffen.mueller@tu-ilmenau.de

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

Fig. 1: Robot platform *Zeus* as a combination of a SCITOS G5 base by MetraLabs GmbH with an arm by Kinova

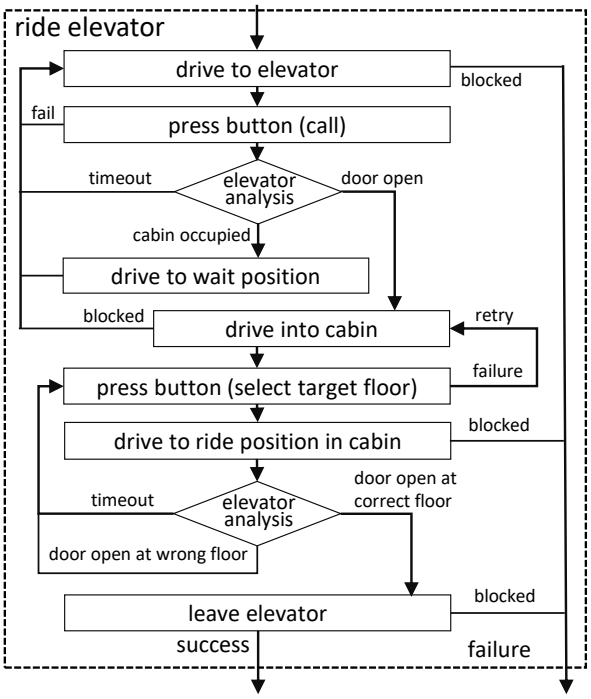


Fig. 2: Procedure of riding the elevator

skills of the robot can be found. In practical application there arise many challenges like short time spans to react when the lift doors open or unforeseen interactions with people occupying the robots way. We discuss these in the following as well.

II. Robot Platform and Background

This work is part of the research project RobInCare[1] dealing with basic capabilities of mobile service robots to enable autonomous navigation in nursing homes for the elderly. When deployed to deliver mail and other items or to pick up residents at their homes for group events, the robot has to open and close doors and needs to use elevators with its on-board manipulation skills.

The robot we use is a SCITOS G5 differential drive platform equipped with an additional Kinova Gen II 7 DoF arm for manipulation tasks. The wheels are big enough to avoid getting stuck in the 4cm gap at the lift cabin entrance. For perception of the environment, the robot has an Azure Kinect and an ASUS Xtion depth camera on top of a pan-tilt unit (PTU). Additionally, we use the three axes accelerometer of the SCITOS G5 for recognizing the vertical movement of the lift.

Navigation and localization of the robot platform relies on a SICK S300 laser range scanner mounted in a horizontal position. For self-localization in the environment, we apply a Monte Carlo Localization [14], which uses laser to map matching, and additional observations on button panel locations which have been mapped before. This will be explained in Sec. V-B in more detail.

The navigation skills rely on [15] which allows maneuvering in the lift cabin that is only a few centimeters wider than the robot's footprint. The planning of arm motions is done according to [16], which allows us to consider perceived obstacles in real-time.

III. Overview of the Elevator Riding Procedure

The whole elevator riding process is integrated in our behavior-based software architecture implemented in MIRA[2]. The higher order behavior decides to use the elevator when the plan to the goal requires changing the floor. Then the *ride elevator* procedure of Fig. 2 takes over. Here the robot at first navigates to a starting position in front of the elevator. Then the press *button procedure* is triggered which will be explained in Sec. IV. This returns control after the push force has been recognized and the robot's arm has been retreated to its home position allowing for safe navigation. An additional feedback on the success of the call button action is not implemented, since the robot will notice the effect when the elevator doors open within a reasonable time. So, the analysis of the elevator by means of the laser scan is continuously running in the background yielding information on the cabin door's state and the occupancy of the cabin. This is further explained in Sec. VI.

Once the door opened, the robot can enter the cabin if it is empty. Otherwise, it will give way for the people and retries the whole procedure. This is commented politely via specific voice outputs. In case that the robot reaches the goal position inside the lift cabin, again the *press button* procedure can take control. In our database, the semantics of the different buttons of the control panel have been mapped and, thus, the correct one can be selected according to the target floor. After

[1]https://www.robincare.de/

[2]https://www.mira-project.org

pressing the button, the robot places itself in front of the exit door, because the doors open for a short moment (6 sec) only. Again an explicit feedback on the success of the button action is not implemented. The robot will react on the opening door and counts floors to decide whether the target floor has been reached or an unexpected intermediate stop took place. If the later case, when the door is opening the robot comments to the people who have called the elevator that he is occupying the cabin and asks for patience. If there are no people waiting and the cabin is not going to move to the next stop, the button for the target floor is pressed again, which is also the case if the cabin does not start to move after the selection of the target. In the normal case, the robot simply drives to a point outside the cabin and control is returned to the higher level behavior responsible for the navigation to the actual target (e.g. the apartment of the resident to be visited). There are several possible points of failure, which in most cases can be handled by a retry. Only if the navigation path is blocked, or the whole system gets stuck after a collision, the procedure has to be aborted with an exception.

Challenges for the design of the procedure were the short time intervals for maneuvering the arm in a safe position for platform movements and the required time to plan the movements. If the elevator is already at the current floor when the robot presses the call button, he has only around six seconds to enter the cabin before the doors close again. This has been solved by finding an optimal position for the robot when pressing the button. It has to be nearly centered in front of the door, but on the other hand, this should not be too close to the door to allow for robust button panel perception and room for manipulation actions. Finally, the robot is slightly angled such that it can enter the cabin with a gentle arc movement. Additionally, the retreat motion of the robot arm could not be executed with the motion planner in the short time. Instead, we reversed the trajectory for the pushing action and executed that at a higher speed assuming that the scene is still free of obstacles in that region. The same timing issues are present when the elevator already is on the target floor, and the robot would presses the respective button inside, which causes the door to open immediately, and there is not enough time to leave the cabin. Luckily, this case only can happen when the path to the outside of the cabin is blocked by people and the robot has to retry to leave the cabin after verbal communication to the people outside. A solution for that problem is that if the robot is already on the goal floor the button for another floor is pressed, causing the elevator to take a detour, which gives time for maneuvering inside the cabin. The original target floor then is reached after another cycle of selecting the destination by button.

IV. Pressing the Button

The actual procedure of pressing a desired button is shown in Fig. 3. It starts with a navigation to a panel specific interaction position, which allows for a good camera view at the buttons as well as good reachability. Then the camera is pointed at the control panel, while button detection and panel localization run in background (see Sec. V-A). Once

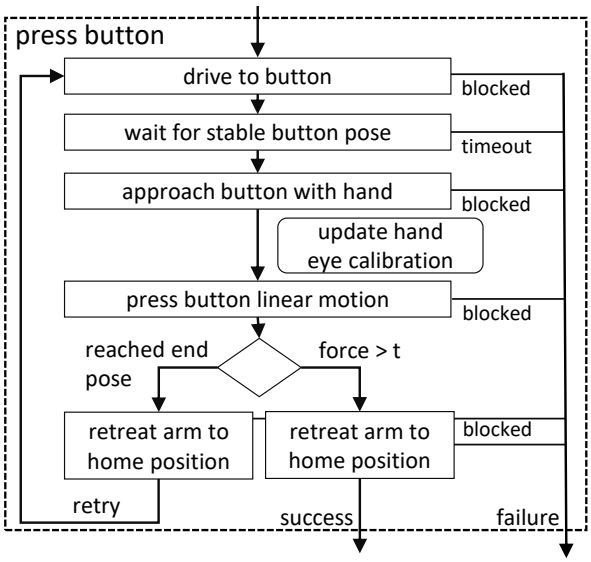


Fig. 3: Procedure of pressing a button

the pose estimation stabilized, the manipulation starts by bringing the robot's finger 10 cm in front of the estimated button. Since the target position of the button is estimated in camera coordinates, and the camera is on a PTU, and additionally the whole robot frame is not rigid, there is a calibration offset from the internal robot model and the actual position of the hand in respect to the camera. Depending on the position in the reachable area, this deviation can be up to 3cm, which is critical for hitting the button correctly. To solve that calibration problem, we included an estimation of the end-effector pose in camera coordinates by means of an ArUco marker [17] detection. A hexagonal marker arrangement has been mounted to the robot's wrist, allowing to see at least two non-co-planar markers at a time (see Fig. 4). By means of the calibrated camera, the 6D pose of the marker arrangement can be estimated, yielding an offset to the internal model of the robot. This offset then is used to correct the goal position for the next movement to be executed. This is a linear motion to a virtual point 2 cm behind the button of interest. During the execution, the collision check for the fingertip is disabled to allow for the contact with the button, which otherwise would be avoided because the wall is contained in the collision scene of the motion planner. The success of the operation is monitored by means of the end-effector force. If the force in movement direction exceeds a threshold t of 6 N, the button is supposed to be pressed and the arm can be retreated to the home position. If the force has not been noticed before the end point is reached, then the button is supposed to be missed and the whole process starts over again.

Possible points of failure for the whole procedure are unreachable positions. Either the position for the robot platform is occupied, or the robot's arm can not reach the desired start and end point of the push trajectory.

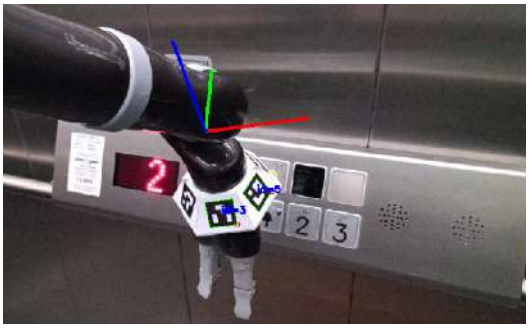


Fig. 4: Localization of the robot's end effector in camera coordinates by means of ArUco markers used for online hand-eye calibration.

V. Button Detection and Localization

Detecting and distinguishing individual buttons either with classical or machine learning methods always is prone to confusion of labels or complete misses due to occlusion for example (see Fig. 5). In order to make the button pressing process more robust, we decided to incorporate prior knowledge on the existing button arrangements in our operational environment. The raw detections of buttons are compared to previously mapped button panels in order to optimize their position and correct labels. The recording of panel configurations during installation is done using the same detector as for online recognition but under optimal recognition conditions without occlusions and from a perpendicular view.

Furthermore, to reduce the influence of occasional false detections, the recognition process during application is not only done once but runs in background at 4 Hz yielding a stream of button panel locations that are filtered for outliers by means of a geometric median.

On the first instance, the location of the button panel and therefore the exact position of the individual buttons is used for planning a push trajectory for the end effector of the robotic arm, but the deviation of the panel's relative position in robot coordinates to the mapped panel position is also fed into the MCL as additional cue for localization. This allows a reduction of the typical laser-based localization error from about 5 cm, which is related to the occupancy grid map resolution, to a value in the 1 cm range. Therefore, also static collision scenes can be considered by the robot during manipulation, which prevents contacts to walls and surfaces that are not visible in the depth camera directly.

In the following the realization of the neural-network-based button detection is described before the actual localization of button panels is explained, which uses the set of detected buttons.

A. Raw Button Detection

For the recognition of individual buttons in color images of our Azure Kinect, we trained a Faster-RCNN network on the dataset of [8].

The aim is to distinguish 16 classes of buttons, which are the special functions (open doors, close doors, alarm) as well as most prominent floors B (basement), L (lobby), G (ground floor), 1, ..., 9, and some wild card buttons with unrecognizable labels called 'button'. The exact semantics of the buttons is not necessarily to be detected by the network since it is mapped to the buttons in the button panel as stored in the robot's database as described in the next section.

The network architecture we used is more lightweight than in [8] since we use a ResNet-50 backbone instead of ResNet-101 in order to save resources on our mobile hardware (Nvidia Geforce RTX 2060). Additionally, we did not use the OCR-RCNN approach proposed in [8] as the number of floors and therefore the possible elevator buttons are within the set of 15 classes. The slightly worse AP_{50} value of 85.8% (compared to 90.1% of the ResNet-101) is acceptable due to the further processing in the panel matching process.

Once the bounding boxes of candidate buttons have been found, the next step is creating 6D poses for each of them by means of projecting the boxes onto a point cloud of a depth camera. We use the ASUS Xtion depth image due to more reliable geometric properties of that active stereo camera. The depth image of the Azure Kinect camera shows material dependent depth offsets and other artifacts due to the time of flight approach. From the 3D points inside the bounding boxes, the surface normal and the xyz-coordinate of the center can be extracted by means of a plane regression. Using the vertical axis as granted, the full rotation matrix can be easily computed for the normal and the z-unit vector by means of three cross product operations, which completes the 6D pose of the buttons.

B. Panel Matching

Having the incomplete set of button poses and respective predicted labels, the association to known button panels can be done. From the robot localization we can find the panel of interest in our database and for that the relative position of the buttons to the panel's origin (at its center) is known.

The detection of the button panel's pose in respect to the camera is done by a Maximum Consensus approach. For that, each pair of possible associated buttons, one from the known panel in the database p_i and one from the current detections d_j, is used to compute the relative 6D transformation between them. This in the following can be used to project all the detected buttons onto the known panel. For each detection, we then search for the closest button p^* on the known panel and accumulate the distances. Here the match of the predicted and the mapped label are taken into account. Label mismatch yields a penalty offset ρ to the distance causing a possible association to neighboring buttons that might be a better match.

From that accumulated minimal distances to associated buttons a matching score $m(p_i, d_j)$ is computed to rank the predicted transformation.

$$m(p_i, d_j) = e^{-\left(\sum_j min_i\left(|\mathbf{p}_i - \mathbf{d}_j| + l(p_i, d_j)\right)\right)^2/\sigma^2} \quad (1)$$

Here $\mathbf{p}_i$ and $\mathbf{d}_j$ are the xyz-coordinates of the panel buttons and projected detected buttons respectively. The term $l(p_i, d_j)$ has a value ρ (free parameter) if the labels do

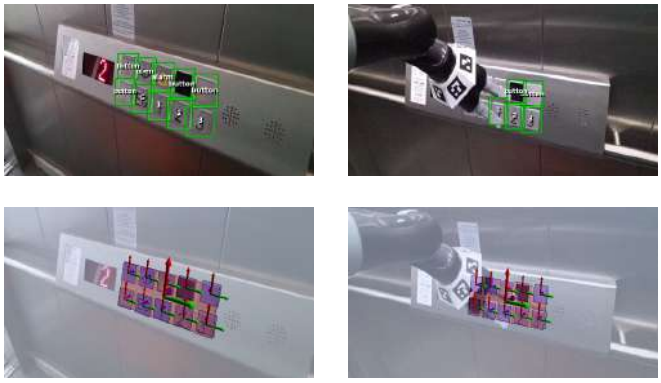


Fig. 5: Examples of exact button panel localization in case of occlusion (right) and in normal conditions (left). top: raw detections from the Faster RCNN; bottom: estimated 6D pose of button panels and respective buttons

not match and 0 otherwise. This ensures that geometrically ambiguous arrangements of buttons can be resolved by means of the associated labels. See Sec. VIII for selection of ρ. The parameter σ has been set to 0.03 m, which defines the matching radius to the order of a single button's size.

The pairing of detection and panel button with the maximum matching score yields the final hypothesis for the panel's pose estimation.

Fig. 5 shows some examples of the detection, which is robust even if more than half of the panel is occluded by the robot's gripper.

VI. Elevator Analysis

Once the lift call button has been pressed, the robot needs to detect when the cabin is ready for boarding. Therefore, the laser range scan is analyzed, while the robot is waiting in front of the door. Individual scan rays' line segments in map coordinates are intersected with a virtual line 5 cm behind and parallel to the door. Then the left most and right most intersection point on that door line define the traversable gap. A threshold on the gap's width allows to distinguish door open and door closed state.

This method also works from inside the cabin, but in this case the virtual line is offset 5 cm to the outside of the door in order to become tolerant against localization errors.

Once the door is open, the robot needs to decide whether the cabin is empty or occupied. Due to safety reasons, we do not allow the robot to enter an occupied cabin, as people could be blocked to leave the lift. To check the cabin, again the laser range scan can be used. If the number of scan end points inside the polygon of the cabin exceeds a threshold, the cabin is considered to be occupied. To compensate for small localization deviation the polygon has a safety margin of 10cm to the actual walls of the cabin. Also people in front of the elevator need to be recognized when the robot has to decide whether to wait outside the lift or when leaving the cabin. To that end a square of 1.5m by 1.5m in front of the elevator's door has been defined as the waiting area. This area

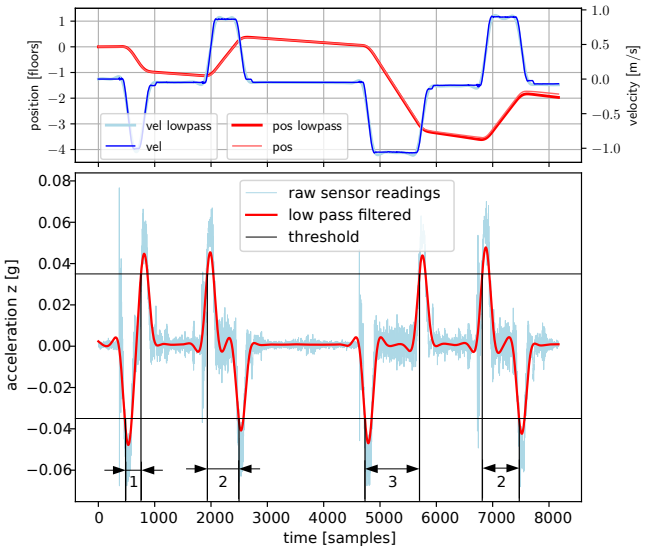


Fig. 6: Bottom: vertical acceleration sensor readings while traveling one floor up, two floors down, three up, two down. The time intervals between the acceleration peaks indicate the floor difference. Top: integrated acceleration (velocity) and integrated velocity (position) show a drift (red curve should end at start level).

can be checked in the laser range scan similar to the inside of the cabin. Usually, there are no other dynamic obstacles in that region other than people.

VII. Floor Recognition

After the robot finally has entered the cabin and the destination floor has been selected by means of another button action, the system needs to detect the current floor in order to leave the cabin at the right one.

While other implementations try to read the indicator screen inside the cabin [11], we rely on the built in acceleration sensor of the SCITOS G5 robot.

We found that simply integrating the vertical component of the acceleration two times is not reliable enough. Even after careful calibration, the resulting distances drift rapidly over more than the height of one story. Fig. 6 (top) shows that drift in the red curve, which in practice should end at the same level as it started. Instead, we found that in most elevators the duration for changing from one specific floor to each other are more or less constant (see Fig. 6 (bottom)). This is especially true, if the weight of the cabin is constant which is guaranteed since the robot will ride the elevator alone. Therefore, by using the acceleration sensor, we simply detect the acceleration event of the cabin at the start and the deceleration event at the end by a simple threshold on the low-pass-filtered vertical acceleration value. The low pass filter shown in Fig. 6 ensures that little bumps at the entrance of the lift do not trigger floor change recognition when the robot enters or leaves the lift. This approach may reach its limits, when the number of stories increases. The potential deviation of the travel time increases with the overall distance

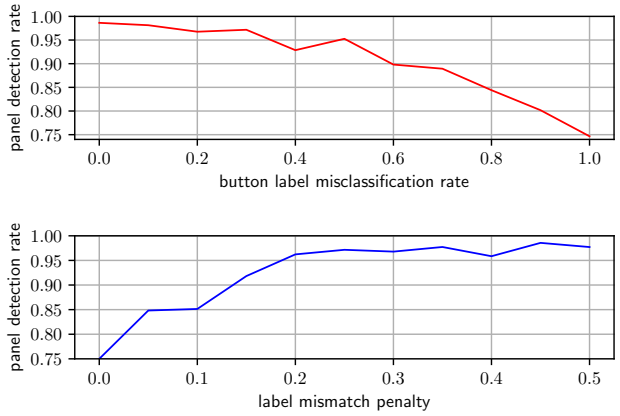


Fig. 7: Ratio of correct panel localization depending on the amount of misclassifications of the button labels (top) and in dependency of the label mismatch penalty parameter ρ (bottom). Both were evaluated over a sequence with heavy occlusions of two button arrays.

and finally reaches the duration of a single floor transition with an increasing number of stories. In our test facilities we only had access to a four story elevator. Therefore, during our real world application the floor recognition did not fail a single time.

VIII. Experimental Evaluation

We use the proposed methods in two installations of the autonomous assistance robot. One is operating in our university building and the other in the target facility, a nursing home for elderly in Ilmenau Germany. First, the results of a quantitative evaluation of the button panel localization is presented before the insights into the development process and the results of a one week application test are discussed.

A. Evaluation of the Button Panel Localization

In the following, we report the analysis of the button panel localization which is a prerequisite for the precise robot localization and button interaction.

In order to evaluate its robustness, we have recorded a video sequence of the robot observing the button panels of two different elevators. This dataset shows occlusions to parts of the panel due to the robot's hand operating on the buttons in 35% of the images.

First, we wanted to show the panel detection results in presence of misclassified buttons in the raw detections. Unknown buttons will effect the panel localization in the same way. This would be the case in a building with more than 10 stories since we only use the 16 button classes as described in Sec. V-A. For the experiment, the predicted labels in the video sequence have been artificially invalidated randomly for a variable amount of raw detections in each image. The label match parameter ρ has been set to 1.0 for that experiment. Fig. 7 (top) shows that for the given setup the panels have been localized correctly in more than 97% of the frames with label drop out rates of up to 30%. The position of the detected panels has been count as false if it snapped to another grid position, which means that there was a position offset of more than 5 cm in 3D world coordinates. Note, that this is not the accuracy of the localization in camera coordinates since it includes the localization error of the moving robot platform.

The position accuracy of the correctly detected panels in camera coordinates inherits directly from the accuracy of the raw button detection and has been evaluated as follows. Knowing the size s of the buttons in the real world, the position accuracy could be evaluated by comparing the offset d of the detected box to the ground truth box and the ground truth box dimensions w.

$$\bar{e} = \frac{1}{N}\sum_{n=1}^{N} s_n \frac{d_n}{w_n} \qquad \hat{e} = \max_{n\in[1,N]} s_n \frac{d_n}{w_n} \tag{2}$$

Using the test dataset of known buttons, the average and max position offset has been found to be $\bar{e} = 0.16$ cm and $\hat{e} = 0.33$ cm. This small offset is increased slightly when the 3D position is evaluated from the point cloud data but at the end it is reasonable for hitting the button with the robot's finger.

A further evaluation deals with the free parameter ρ of the panel matching algorithm. Although, it is possible to find the correct matching buttons pairs without any label information if the corners of the panel are not occluded, in cases with the arm in front of the panel the correct pairing based on the label makes the localization of the panel more robust. Fig. 7 (bottom) shows the influence of the parameter ρ. When label matches are not rewarded at all ($\rho = 0$), then due to occlusion 25% of the frames yield a panel detection that is snapped to the wrong grid position. When ρ reaches 0.2, the association mistakes can be reduced and the false panel detections drop to about 3%. The error that might be introduced by a ρ that is too high depends on the actual misclassification rate of the raw button detector. For our practical application that rate is so low that no negative effects of a big ρ could be observed.

B. Application Test and General Findings

The development of the elevator procedure took place in the said facilities leading to a functional transportation service that has been tested in our office building for a week. There were 96 transportation tasks that required changing the floor. In three cases a human intervention was necessary to recover the robot from a deadlock. In one case the robot got stuck when leaving the cabin. The drive system was not able to overcome the gap. In the other cases the robot stopped assuming to be in a collision with the static environment, which actually was not the case. This can happen when the platform gets moved passively or the localization system changes the position without an actual movement of the robot. In the later case the robot model virtually is placed inside e.g. a wall, which causes the collisions.

During the application no problems with the floor recognition could be recorded. The travel-time-based tracking of floors seems to be robust against all jerks happening.

There was a total number of 305 press button procedures, of which seven missed to hit a button at all. That means the force threshold has not been exceeded, which could be caused by a unfavorable positioning of the robot in relation to the button panel of a wrong localization of the panel (distance estimated too large). The unexpected high number of button operations is caused by repeatedly calling the elevator when it is occupied or busy. The timeout for a retry was only 20 sec. When the lift is stopping at other floors this often is not long enough.

During development, the robot occasionally pushed away itself while pressing the call button in front of the lift. Then the changed orientation caused difficulties with the following detection and cabin entering movements. This misbehavior could be mitigated by means of active breaking when the push trajectory is executed with the arm.

A further, more critical point of failure was related to the navigation while entering the cabin. When the robot struggles to get over the doorstep it can get stuck when the cabin doors close and hit the robots bumper. In that case the hardware needs 3 seconds until safety stop is released and the door closes again soon. A fallback strategy for this situation has been implemented, which consists of a manual drive command for going backwards for 10 cm in order to escape from that loop.

IX. CONCLUSIONS

In this paper we present a complete pipeline for automated operation of an elevator by a mobile robot. We introduce our button localization approach based on the complete panel, which makes it robust to detection errors made by the button detector. Additionally, by localizing the panel our approach can handle occluded buttons during push operation allowing us to continuously track the button's position.

We evaluated the panel localization and the complete pipeline through real world trials with two different elevators. While most of the problems remaining are caused by strict time constraints caused by the small time frame in which the elevator doors are open, our pipeline is effective enough to be used in real world applications.

REFERENCES

[1] J.-G. Kang, S.-Y. An, and S.-Y. Oh, "Navigation strategy for the service robot in the elevator environment," in *2007 International Conference on Control, Automation and Systems*. IEEE, 2007, pp. 1092–1097.

[2] H.-M. Gross, A. Scheidig, K. Debes, E. Einhorn, M. Eisenbach, S. Mueller, T. Schmiedel, T. Q. Trinh, C. Weinrich, T. Wengefeld, *et al.*, "Roreas: robot coach for walking and orientation training in clinical post-stroke rehabilitation—prototype implementation and evaluation in field trials," *Autonomous Robots*, vol. 41, pp. 679–698, 2017.

[3] J. Liebner, A. Scheidig, and H.-M. Gross, "Now i need help! passing doors and using elevators as an assistance requiring robot," in *Social Robotics: 11th International Conference, ICSR 2019, Madrid, Spain, November 26–29, 2019, Proceedings*. Springer, 2019, pp. 527–537.

[4] S. Rosenthal and M. Veloso, "Mobile robot planning to seek help with spatially-situated tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, 2012, pp. 2067–2073.

[5] F. Babel, P. Hock, J. Kraus, and M. Baumann, "Human-robot conflict resolution at an elevator - the effect of robot type, request politeness and modality," in *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2022, pp. 693–697.

[6] W.-t. Law, K.-s. Li, K.-w. Fan, T. Mo, and C.-k. Poon, "Friendly elevator co-rider: An hri approach for robot-elevator interaction," in *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2022, pp. 865–869.

[7] D. Troniak, J. Sattar, A. Gupta, J. J. Little, W. Chan, E. Calisgan, E. Croft, and M. Van der Loos, "Charlie rides the elevator – integrating vision, navigation and manipulation towards multi-floor robot locomotion," in *2013 International Conference on Computer and Robot Vision*, 2013, pp. 1–8.

[8] D. Zhu, T. Li, D. Ho, T. Zhou, and M. Q. Meng, "A novel ocr-rcnn for elevator button recognition," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3626–3631.

[9] P.-Y. Yang, T.-H. Chang, Y.-H. Chang, and B.-F. Wu, "Intelligent mobile robot controller design for hotel room service with deep learning arm-based elevator manipulator," in *2018 International Conference on System Science and Engineering (ICSSE)*. IEEE, 2018, pp. 1–6.

[10] A. A. Abdulla, H. Liu, N. Stoll, and K. Thurow, "A robust method for elevator operation in semi-outdoor environment for mobile robot transportation system in life science laboratories," in *2016 IEEE 20th Jubilee International Conference on Intelligent Engineering Systems (INES)*. IEEE, 2016, pp. 45–50.

[11] J. Lee, X. Cui, H. Kim, S. Lee, and H. Kim, "Elevator riding of mobile robot using sensor fusion," in *The 8th International Conference on Robotic, Vision, Signal Processing & Power Applications: Innovation Excellence Towards Humanistic Technology*. Springer, 2014, pp. 89–98.

[12] R. Stricker, S. Müller, E. Einhorn, C. Schröter, M. Volkhardt, K. Debes, and H.-M. Gross, "Interactive mobile robots guiding visitors in a university building," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2012, pp. 695–700.

[13] J. Krejsa, S. Vechet, K.-S. Chen, M. Havelka, and M. Černil, "Mobile robot in the elevator: What floor am i on?" in *2022 20th International Conference on Mechatronics-Mechatronika (ME)*. IEEE, 2022, pp. 1–5.

[14] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[15] S. Müller, T. Q. Trinh, and H.-M. Gross, "Local real-time motion planning using evolutionary optimization," in *Towards Autonomous Robotic Systems: 18th Annual Conference, TAROS 2017, Guildford, UK, July 19–21, 2017, Proceedings 18*. Springer, 2017, pp. 211–221.

[16] St. Mueller, B. Stephan, and H.-M. Gross, "Mdp-based motion planning for grasping in dynamic szenarios," in *Europ. Conf. on Mobile Robotics (ECMR), Bonn, Germany*. IEEE, 2021, p. 8 pages.

[17] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314000235

Scalable Evaluation Pipeline of CNN-based perception for Robotic Sensor Data under different Environment Conditions

Naeem Iqbal[1], Mark Niemeyer[1], Jan Christoph Krause[1], Joachim Hertzberg[1,2]

***Abstract*— Deep learning impacted a wide variety of perception applications for autonomous mobile robots. In classic computer vision benchmark tests, new algorithms keep appearing that outperform each other. However, these benchmark tests cannot be generalized, so that the specific application must be considered for the selection of sensors and algorithms. Especially in the agricultural domain, environmental conditions like weather and vegetation significantly influence the reliability of sensor systems. Therefore, it is essential to test different sensor modalities and algorithms in the operational design domain. This motivates the need for an evaluation framework which has the flexibility to compare and validate various perception algorithms, sensors suites, and data samples with a focus on different conditions.**

This paper proposes a pipeline combining a test environment (*AI-TEST-FIELD*), a semantic environment representation (*SEEREP*), and an inference server (Triton) for an automatic evaluation of different CNN-based perception algorithms under various environment conditions. Recurring and comparable recordings of raw sensor data with identical scenarios and objects can be performed on the test field, with the only difference being the environmental conditions. The inference results are inferred once and stored alongside the sensor data in *SEEREP*. Thus, they can be queried efficiently based on the environment conditions to generate (partially overlapping) subsets of the whole dataset. It is demonstrated how this pipeline can be used to apply the CNN-inference just once on the data, and how the queried subsets can subsequently be used to evaluate the performance in different environment conditions.

I. Introduction

Nowadays, various algorithms for semantic environment perception benchmark suits such as KITTI [1]–[3] and NuScenes [4]–[6] exist. For these benchmarks, different perception algorithms are evaluated against the corresponding fixed dataset. This results in an overall performance metric. Though, there is no option to test the same algorithm against different sensor suites or specific subsets of the datasets with specific environment conditions. However, these specific evaluations are of interest when designing new perception modules for autonomous systems. They enable an extensive search for the best combination of sensor setup, perception algorithms, and field conditions. Especially in the agricultural domain, this is needed because of the wide variety of field conditions such as dust, fog, clouds etc.

Similarly, it is also of interest to evaluate various perception models and compare their performance on different benchmarks to get the best model for a specific perception task (like human detection). A model evaluation can imply several kinds of evaluations, ranging from evaluation in terms of model robustness against adversarial attacks, inference speed, or in terms of accuracy on a carefully designed test dataset that covers all the corner-cases. While there are many datasets available for different perception tasks, they are mostly focused on developing state-of-the-art neural networks for those tasks. For example, how one network can better detect a set of objects in the given dataset compared to any other network. This helps in advancing neural network architecture research. Each dataset has its own accompanying benchmark as well as perception challenge, and all these datasets are either recorded with the same sensor setup [3] or acquired without any sensor information available [7]. In contrast to that, it is as well of interest how a neural network behaves in a specific perception task when the dataset is changed from one sensor to another (also known as domain gap) or when the field conditions change (see fig. 1).

(a) sunny (b) cloudy

Fig. 1: **Example images** from the *AI-TEST-FIELD* with different environment conditions (sunny and cloudy) demonstrating the qualitative differences due to environment conditions. The bounding boxes of the ground truth (green) and the detections of the YOLOv5m trained on the COCO dataset (red) are drawn in the images. Notice that the *person* here is a dummy and not a real person.

The project AI-TEST-FIELD is supported by funds of the Federal Ministry of Food and Agriculture (BMEL) based on a decision of the Parliament of the Federal Republic of Germany. The Federal Office for Agriculture and Food (BLE) provides coordinating support for artificial intelligence (AI) in agriculture as funding organization, grant number 28-D-K1.01A-20. The DFKI Niedersachsen (DFKI NI) is sponsored by the Ministry of Science and Culture of Lower Saxony and the VolkswagenStiftung.

[1]Plan-Based Robot Control Group, DFKI Niedersachsen, German Research Center for Artificial Intelligence, Osnabrück, Germany
Contact: naeem.iqbal@dfki.de

[2]Knowledge-Based Systems Group, Osnabrück University, Osnabrück, Germany

979-8-3503-0704-7/23/$31.00

This paper addresses the evaluation of CNN-based perception for robotic sensor data from different sensors and under different environment conditions. In contrast to the common evaluation workflow of (1) having a dataset, (2) performing the inference for the complete dataset, and (3) evaluating all inference results, this paper proposes a methodology which allows the creation of specific subsets of data based on user-defined parameters such as field conditions (e.g., rainy or clear weather), sensor modality (e.g., LiDAR or camera), and other spatio-temporal-semantic information in the data. For this, the sensor data is stored in the semantic environment representation *SEEREP* by Niemeyer et al. [8]. With the help of *SEEREP* one can use spatio-temporal-semantic queries to create subsets of data. This feature can be used to evaluate perception models on different subsets of a larger dataset with specific characteristics. This can be very beneficial in agricultural domains where the field conditions diverge by such a large degree (also known as domain gap) that neural networks do not generalize well enough.

In the next section, the related work is discussed, and our methodology is presented in section III. To demonstrate the benefits of our methodology, exemplary evaluations are demonstrated in section IV. Finally, this paper concludes and gives an overview of future work in section V.

II. Related Work

In this section, related work regarding perception datasets and benchmarks as well as the storage solutions for those datasets are discussed. An overview of CNN-based environment perception approaches and performance evaluation of the perception results is also given.

The development and evaluation of AI algorithms for environment perception are based on a broad set of sensor data equipped with ground truth. To provide a general overview of the performance of different algorithms, challenges, and benchmarks have been announced, e.g., ImageNet [9], PascalVOC [10], COCO [7] and KITTI [3]. In general, this includes datasets, that contains everyday scenes with a focus on urban and driving scenarios. The corresponding datasets, especially COCO and ImageNet, were often used to train AI Models like YOLO [11] or models from the Detectron Model Zoos [12], [13].

Running object detection algorithms on images results in 2D bounding boxes that mark the area in which an object was predicted. As pointed out in Zou et al. [14] a large number of object recognition algorithms like R-CNN [15], SSD [16] and YOLO [17] have been developed recently, so that the performance level is continuously surpassed. Such an object detection task can be evaluated both in terms of speed and accuracy. For human detection and tracking, Godil et al. [18] proposed a set of detection and tracking metrics. Besides the real-time capability of an object detection algorithm, Average Precision (AP) is the most known metric for evaluation. For the different benchmarks, there are comparable but essentially different ways of calculating the AP [19]. In general the authors of the benchmark suites provides tools or an API like the PyCoCoTools [7], KITTI's object development kit [3] and the PASCAL VOC2012 Development Kit [20]. The IoU describes the closeness of two bounding boxes. Based on a threshold, a prediction results in a True or False Positive. Missed object are counted as False Negatives. For multiple thresholds, the Precision-Recall-Curve is generated from these values. The area under that curve is called Average Precision (AP), a key indicator for the performance of object detection algorithms. For a universal and flexible evaluation workflow, it is essential that different metrics from different APIs are applicable.

Especially for safety reasons, a detection system has to operate reliably in their field of application. For autonomous vehicles, the definition of the operational design domain (ODD) describes all conditions, scenarios [21]. To evaluate the performance of a perception algorithm in certain domains like off-road and agriculture, datasets focused on that special domain are needed. Available datasets such as Marulan [22], RELLIS-3D [23], NREC [24] and FieldSAFE [25] in particular cover the typical harsh and changing environmental conditions, but are not comprehensive. Thus, a mixture of these datasets with a composition of self recorded data is needed.

MSeg [26] provides tools for accessing handling a selection of datasets as well as for fusing and mapping their corresponding class names. Although scripts for accessing the datasets are provided, a separate download for each user is still required. For handling the individual datasets, still different APIs are used.

Typically, the sensor data is provided in huge file archives with image files and attached *.txt*- [27], *.xml*- [10] or *.json*-files [7] that contain the annotations. Kragh et al. [25] provide time series data in rosbags and corresponding ground truth by masking different colors in orthophotos.

In datasets, different categories for filtering like occlusion or truncation [3], [24], object's motion or pose [24] or environmental conditions like day and night [22] are added to the attached descriptions files. Thus, the developer must first download the entire dataset for a specific performance evaluation and then generate the specific subset using the filter criteria provided. This effort and the related resource requirement is avoided by querying the data available on a centralized storage solution with filtering capabilities. Data Version Control (DVC) [28] is used for data pipelines and accessing the data, especially to track the change in datasets and model files [29]. DVC is a suitable tool for providing and fetching the data. This is very useful for the development of an AI algorithm with expanding data. For the evaluation, tracking changes of datasets in terms of versioning is not the focus. It is much more about calculating evaluation metrics for different sets and subsets. And the data to be used for this are to be composed by filtering.

A user can potentially fetch the data from a file server and feed it to an inference node that is, for example, a python program. For that a broad number of deep learning frameworks like PyTorch [30], TensorFlow [31], TensorRT [32], OpenVINO [33] etc.) exist. However, this workflow has the disadvantage that for each neural network trained

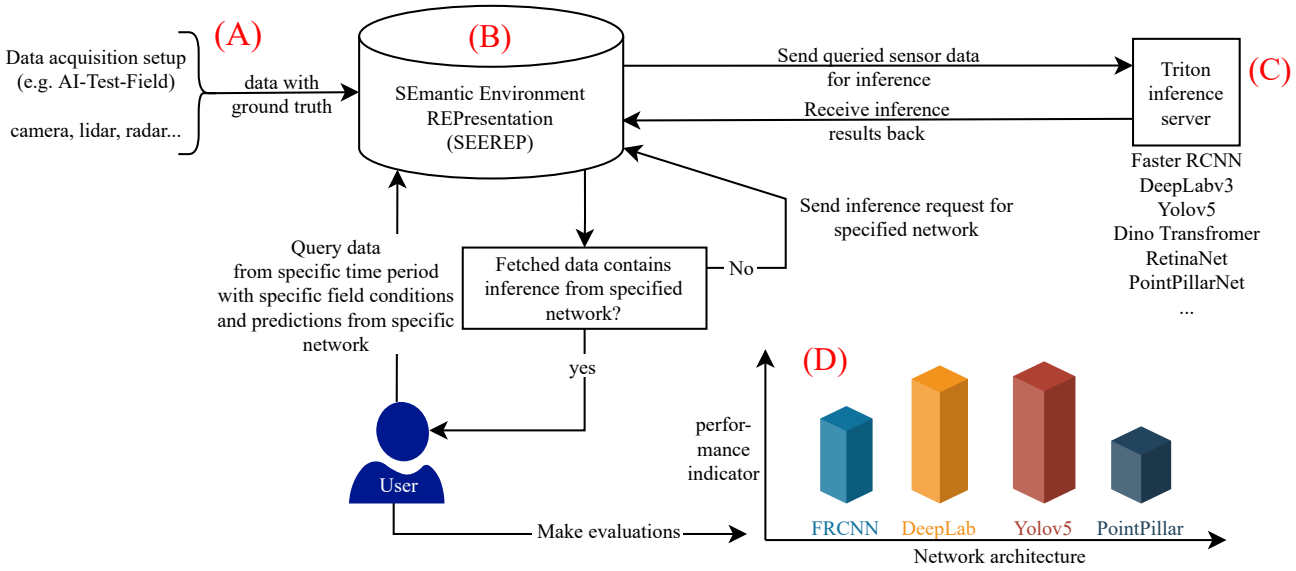


Fig. 2: **Architecture overview.** This diagram visualizes the architecture of the proposed methodology. It shows the data flow from the data acquisition setup (A) to *SEEREP* (B), the connection to the Triton inference server (C), and a data query for an evaluation from the user's perspective (D).

on a different deep learning framework, the pipeline has to be modified and version control gets increasingly complicated. This is where Triton Inference server proves very helpful. Triton [34] is one packaged open-source software by NVIDIA where the deep learning frameworks are served as backends. With this capability, Triton can digest models from all the major training frameworks without changing the inference API, making it more flexible and scalable. Among other benefits, it is possible to store multiple versions of the same model and play with input resolution vs. accuracy trade off. With Triton, one can easily maintain the version consistency with all the backends in an automated fashion.

For the evaluation of perception systems, flexible and freely composable datasets are needed as a basis. Different emphases in the evaluation goals are to be addressed by selection of special subsets. For evaluation purposes, a data holding system is needed that allows filtering and querying of evaluation data. It is mandatory that the queried evaluation data can be given to different algorithms for evaluation via an interface that is as identical as possible. Various metrics may be relevant in the evaluation. Therefore, an interface with a flexible integration of the metrics is relevant here as well. With the proposed Evaluation Pipeline, the benefits of each area are incorporated. The datasets that fit the evaluation task best, the algorithms that are most promising, and the metrics and tools that are appropriate for assessing viability are brought together here via a unified pipeline system.

III. Methodology

The proposed methodology which handles the sensor data, annotates the data with inference results, and serves everything for evaluation consists of four major parts (see figure 2):

(A) A data acquisition setup which captures sensor data and provides ground truth annotations. A system like the AI-TEST-FIELD carrier system [35] can be used for this. The carrier system can capture sensor data of the test field with a flexible sensor setup, and it can generate ground truth annotations automatically due to the known environment.

(B) A spatio-temporal-semantic environment representation (*SEEREP*) that stores the sensor data, the ground truth annotations, the inferred annotations, and further semantic information such as weather, lighting, sensor setup etc. [8].

(C) A Triton inference server with several neural networks (for object detection, instance segmentation, and the flexibility to add more) which can infer semantic annotations for the provided data on-demand.

(D) A user or in general a querying system that receives the sensor data with ground truth and inferred annotations for evaluation.

Given that the data acquisition setup provides sensor data with ground truth annotations, the semantic environment representation creates spatial (based on the sensor position and the spatial extent of the data), temporal (based on the point in time of the sensor reading) and semantic (based on the annotations) indices. Using the spatio-temporal-semantic information of the data, a user can query a specific subset of the complete dataset. This dataset can be of a specific, challenging region (spatial query), of a specific point of time (temporal query), or with a specific semantic annotation (semantic query). The semantic annotations can be, for instance, that there is a human visible in the data or that a specific weather condition is perceived in the data.

Additionally, combinations of the three modalities in one query are possible.

In addition, the user can define the neural network from which the inferred annotations should be present in the data. For the same data, multiple inference results from different models can be stored alongside the sensor data in *SEEREP*. If the data matching the query contains the annotations from the specified model, the query can be answered immediately. Though, if the data does not contain the annotations, they have to be inferred. For this, the data is sent to the Triton inference server. If the specified model is not deployed on the Triton server, no data can be used to answer the query. Though, if the specified model is deployed, it is used to infer the annotations. The inferred annotations are then stored alongside each data sample in *SEEREP*. If the same subset of the data and same model are requested in a subsequent run, the predictions will be fetched from the SEEREP server directly and the follow-up inference run will be skipped. This makes the reproducibility of the same query in subsequent runs more time efficient.

Due to the storage of the inference results in *SEEREP* alongside the data, the inference has to be performed only once for the queried data. If the same query is performed twice or if the results of a query intersect partially with the results of a prior query, the inference results stored in *SEEREP* can be used directly and there is no need to block computing power of the inference server. Only the remaining data, which matches the query and has no inference results yet, has to be sent to the Triton server.

Finally, the data with the ground truth annotations and the inference results are sent to the querier. Based on that, use-case dependent evaluations, like the intersection over union, can be conducted to assess the performance of a model. The scenario for each evaluation can be defined by the spatio-temporal-semantic query to fetch specific subset of a dataset from *SEEREP*.

IV. Application Example

For highly automated machinery, a reliable perception of the environment is crucial. In agricultural applications, in particular, not every object in the vicinity of the machine, like plants in front of a harvester, is necessarily an obstacle. Therefore, classification of objects in the working area is needed. In the research project *AI-TEST-FIELD*, sensor systems for human recognition are evaluated. The focus is on the influence of environmental conditions like weather and vegetation. As an application example, the proposed evaluation pipeline is used to evaluate algorithms that detect humans in images during agricultural processes. For that, three different models (YOLOv5m [36], RetinaNet [37], and Faster RCNN [38]) trained on the COCO dataset [7] are used in this application example. All models have a single-scaled input size of 800 x 1333. Any images that do not fit the model input aspect ratio, are resized and padded with zero values until the aspect ratio of the padded image fits.

For a first checkup, the algorithms will be proven using the COCO validation dataset from 2017. To enlarge the validation data, the KITTI dataset is considered afterwards and in order to show the performance in the agricultural application domain, our own test data is added. For the data acquisition of our own data, the rail-based *AI-TEST-FIELD* sensor carrier [35] is used. It is equipped with a ZED2i camera, and two 4activePA pedestrian test targets[1] (an adult and a child) are placed in surveyed positions on the field.

The COCO validation dataset and the KITTI dataset were fed to *SEEREP*. 2693 of the 5000 COCO images have at least one *person* in them. In the KITTI dataset, only 542 out of the 7481 images have a *pedestrian* in them. For this example, the *AI-TEST-FIELD* carrier system acquired images under the three weather conditions: 1) cloudy, 2) sunny, and 3) rainy. Two exemplary images can be found in figure 1. This small dataset was manually labelled with CVAT [39] and afterwards fed into *SEEREP* as well. The complete dataset with all weather conditions has 196 images in total, but only 108 images have a person visible in them.

In general, the *SEEREP* query interface enables the creation of various (partially overlapping) subsets of the overall dataset based on the spatial, temporal, and semantic indices. As a result, *SEEREP* only returns subsets of 2693, 542 and 108 images as an answer to the following queries:

1) `person and COCO`,
2) `pedestrian and KITTI`
3) `person and AITF`

These subsets are non-overlapping, as each query targets another input-dataset. Thus, all images returned by the three queries do not have any inference results stored during previous queries, yet. Therefore, the images have to be sent to the Triton inference server so that the deployed object detector models can infer the detections.

When the querier receives the images with the ground-truth and the inferred annotations, the evaluation can be performed. There are many evaluation benchmark suites available that can be used for the evaluation (like [3], [7]) and the evaluation API can be changed to any other API for other perception tasks such as 3D pose estimation, instance segmentation, etc. For this application example, the official COCO API [40] is used to evaluate the precision of bounding box based predictions. Table I shows the results of the same models evaluated on the COCO person class, the KITTI pedestrian class, and the *AI-TEST-FIELD* images with dummy persons respectively.

TABLE I: Average Precision (AP@IoU=0.5:0.95) of three Object detectors namely YOLOv5m, RetinaNet and Faster RCNN evaluated on the subsets of images by *SEEREP* queries 1) `person and COCO`, 2) `pedestrian and KITTI` and 3) `person and AITF`.

Query	Number of images	YOLOv5m	RetinaNet	FRCNN
1)	2693	48.7	44.9	46.4
2)	542	21.4	22.1	21.9
3)	108	74.4	65.1	67.0

[1]https://www.4activesystems.at/

Among many possible evaluations, an example evaluation investigating the influence of lighting and environmental conditions is carried out below, which can happen quite often in an agricultural context. Thus, the following *SEEREP* semantic queries are used to retrieve the subsets of the *AI-TEST-FIELD* dataset for the evaluation of the person detector under different field conditions:

1) `person and cloudy`
2) `person and sunny`
3) `person and rainy`

The inference results of the models already exist, because the previous query 3) contains all images with persons. Thus, all images were already sent to the inference server, the inference results are stored in *SEEREP* and the queries can be answered immediately. In the table II, the average precision is shown for the different environment conditions.

TABLE II: Average Precision (mAP@IoU=0.5:0.95) of the three Object detectors YOLOv5m, RetinaNet and Faster RCNN evaluated on the subsets of images showing a ***person*** dummy with three subsets of weather conditions. The different subsets of various weather conditions were obtained using the *SEEREP* queries 1) `person and cloudy`, 2) `person and sunny` and 3) `person and rainy`.

Query	Number of images	YOLOv5m	RetinaNet	FRCNN
1)	17	74.3	63.0	58.9
2)	18	54.1	51.3	49.4
3)	73	80.3	69.0	73.1

Based on these results, it can be seen that the models are capable of recognizing humans. A first consideration of the weather conditions shows that especially the sun has a significant influence on the Average Precision. Based on these results, the models might perform much worse in sunny conditions than in cloudy or rainy conditions. For the deployment of the algorithms in the agricultural context, a deeper investigation with a larger database is therefore essential, especially for this environmental condition. Even though the domain specific data presented here does not yet allow any in-depth conclusions about the applicability of the algorithms, the suitability of the pipeline for making such evaluation is shown.

V. Conclusion

This paper presented a new methodology for the evaluation of perception systems for robotic sensor data. The sensor data is stored in the semantic environment representation *SEEREP* which enables the creation of subsets with specific spatio-temporal-semantic properties. The ground-truth as well as the inferred annotations are stored alongside the sensor data in *SEEREP*, and they are used to create the semantic index, which enables the semantic queries. Not only the inference results from a single analysis algorithm, but from multiple can be stored alongside the same data. If the inference results of a specific analysis algorithm is not available in *SEEREP* yet, the data is sent to the Triton inference server. If the requested model is deployed there, the inference is conducted and the results are added to *SEEREP*. Therefore, the inference results are available to answer the current and any future queries. Due to the storage in *SEEREP*, the inference only has to be conducted once, which reduces unnecessary usage of the inference computing power.

The example in section IV demonstrates how the query interface can be used to retrieve subsets of the data to evaluate the deployed detector models under different environment conditions. Even though, no real conclusion can be drawn from the small dataset in this example, it shows the benefits of the presented methodology for the evaluation of a single machine learning model under various environment conditions. As multiple models can be deployed on the Triton inference server and multiple inference results can be stored alongside the data on the *SEEREP* server, this can easily be extended to compare multiple models under various environment conditions so that the best model for each environment condition can be found.

The presented results and the integration in the ongoing research project *AI-TEST-FIELD* motivate further work towards a fully automated data acquisition and analysis algorithm evaluation pipeline. The future work may include:

- Developing the automatic labeling of the sensor data of the *AI-TEST-FIELD* sensor carrier based on the surveyed objects and the carrier position.
- Extensively demonstrating the benefits of querying partially overlapping datasets using *SEEREP*.
- Adding the data type *pointclouds* to the evaluation pipeline to support perception based on laserscanners.
- Evaluating multiple machine learning models with regard to different environment conditions based on a bigger dataset.
- Using spacial queries to evaluate for example the change of accuracy with respect to the distances between sensor and object.

References

[1] J. Fritsch, T. Kuehnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[2] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[3] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[4] K. T. e. a. H. Caesar, J. Kabzan, "Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles," in *CVPR ADP3 workshop*, 2021.

[5] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[6] W. Fong, R. Mohan, J. Hurtado, L. Zhou, H. Caesar, O. Beijbom, and A. Valada, "Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking," in *ICRA*, 2022.

[7] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[8] M. Niemeyer, S. Pütz, and J. Hertzberg, "A spatio-temporal-semantic environment representation for autonomous mobile robots equipped with various sensor systems," in *2022 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 9 2022.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[11] G. Jocher, Ayush Chaurasia, A. Stoken, J. Borovec, NanoCode012, Yonghye Kwon, Kalen Michael, TaoXie, Jiacong Fang, Imyhxy, , Lorna, Zeng Yifu, C. Wong, Abhiram V, D. Montes, Zhiqiang Wang, C. Fati, Jebastin Nadar, Laughing, UnglvKitDe, V. Sonck, Tkianai, YxNONG, P. Skalski, A. Hogan, Dhruv Nair, M. Strobel, and M. Jain, "ultralytics/yolov5: v7.0 - yolov5 sota realtime instance segmentation," 2022. [Online]. Available: https://zenodo.org/record/3908559

[12] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018.

[13] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[14] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[18] A. Godil, R. Bostelman, W. Shackleford, T. Hong, and M. Shneier, "Performance Metrics for Evaluating Object and Human Detection and Tracking Systems," National Institute of Standards and Technology, Tech. Rep. NIST IR 7972, Jul. 2014. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7972.pdf

[19] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242.

[20] M. Everingham and J. Winn, "The pascal visual object classes challenge 2011 (voc2011) development kit," *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, vol. 8, 2011.

[21] On-Road Automated Driving (ORAD) committee, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," SAE International, Tech. Rep., 2021.

[22] T. Peynot, S. Scheding, and S. Terho, "The Marulan Data Sets: Multi-Sensor Perception in Natural Environment with Challenging Conditions," *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1602–1607, November 2010.

[23] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, "Rellis-3d dataset: Data, benchmarks and analysis," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 1110–1116.

[24] Z. Pezzementi, T. Tabor, P. Hu, J. K. Chang, D. Ramanan, C. Wellington, B. P. Wisely Babu, and H. Herman, "Comparing apples and oranges: Off-road pedestrian detection on the national robotics engineering center agricultural person-detection dataset," *Journal of Field Robotics*, vol. 35, no. 4, pp. 545–563, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21760

[25] M. F. Kragh, P. Christiansen, M. S. Laursen, M. Larsen, K. A. Steen, O. Green, H. Karstoft, and R. N. Jørgensen, "Fieldsafe: Dataset for obstacle detection in agriculture," *Sensors*, vol. 17, no. 11, 2017.

[26] J. Lambert, Z. Liu, O. Sener, J. Hays, and V. Koltun, "Mseg: A composite dataset for multi-domain semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2879–2888.

[27] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[28] R. Kuprieiev, skshetry, P. Rowlands, D. Petrov, P. Redzyński, C. da Costa-Luis, A. Schepanovski, D. de la Iglesia Castro, Gao, I. Shcheklein, B. Taskaya, J. Orpinel, F. Santos, D. Berenbaum, daniele, R. Lamy, A. Sharma, Z. Kaimuldenov, D. Hodovic, N. Kodenko, A. Grigorev, Earl, N. Dash, G. Vyshnya, maykulkarni, M. Hora, Vera, and S. Mangal, "Dvc: Data version control - git for data & models," Apr. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7848331

[29] A. Barrak, E. E. Eghan, and B. Adams, "On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects," in *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. Honolulu, HI, USA: IEEE, Mar. 2021, pp. 422–433. [Online]. Available: https://ieeexplore.ieee.org/document/9425888/

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[32] NVIDIA Developer, "NVIDIA TensorRT." [Online]. Available: https://developer.nvidia.com/tensorrt

[33] "OpenVINO™ Toolkit." [Online]. Available: https://github.com/openvinotoolkit/openvino

[34] NVIDIA Corporation, "Triton Inference Server: An Optimized Cloud and Edge Inferencing Solution." [Online]. Available: https://github.com/triton-inference-server/server

[35] J. C. Krause, J. Martinez, H. Gennet, M. Urban, J. Herbers, S. Menke, S. Röttgermann, J. Hertzberg, and A. Ruckelshausen, "AI-TEST-FIELD - An Agricultural Test Environment for Semantic Environment Perception with Respect to Harsh And Changing Environmental Conditions," in *2023 ASABE Annual International Meeting*, ser. ASABE Paper No. 2300757. St. Joseph, MI: ASABE, 2023, p. 1.

[36] G. Jocher, "YOLOv5 by Ultralytics," 5 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[37] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[38] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[39] CVAT.ai Corporation, "Computer Vision Annotation Tool (CVAT)," Sep. 2022. [Online]. Available: https://github.com/opencv/cvat

[40] "COCO API." [Online]. Available: https://github.com/cocodataset/cocoapi

Teach and Repeat and Wheel Calibration for LiDAR-equipped Omnidirectional Drive Robots

Sebastián Bedín[1], Javier Civera[2] and Matías Nitsche[1]

***Abstract*— In this work, we present a novel teach-and-repeat (T&R) method for omnidirectional-drive autonomous ground vehicles, based on the tightly-coupled fusion of LiDAR and odometry readings in a relative representation. In addition to robot localization, we perform online estimation of the platform intrinsic parameters, which significantly enhances the robustness and localization accuracy of the system. We demonstrate the effectiveness of our approach, including the performance of the platform parameters and pose estimation, as well as the general T&R method, through simulation.**

I. Introduction

Unmanned wheeled vehicles are utilized in various forms across numerous application domains, including autonomous vacuuming or transportation among others. Of particular relevance is their application in warehouses and storage facilities, where the goal is to achieve the highest degree of automation possible. In this context, omnidirectional drive robots possess a significant advantage due to their unrestricted 2D motion. Teach-and-repeat (T&R) methods also hold great importance, given the repetitive nature of the robots' tasks and the convenience of defining their trajectories through demonstration. These various aspects serve as motivation for our work.

Although LiDAR sensors stand out in robustness compared against visual ones, they are limited in certain geometric configurations such as long corridors or large scenes with objects further than their range, which are common in warehouses. Wheel odometry is also limited in omnidirectional drive robots due to slippage, and may present inaccuracies due to changes in the robot load, wear or variabilities in a large fleet.

In this work, we formulate a novel T&R method for omnidirectional robots that fuses, in a tightly coupled manner, LiDAR and wheel odometry measurements into a relative representation suited for the problem. By fusing the two sensors we outperform the limitations of both of them. We also include platform-specific parameters in the state, in order to self-calibrate the odometry in the cases mentioned above. In our simulation experiments, we demonstrate the effectiveness of our approach in several challenging scenarios.

II. Related Work

T&R navigation is a classical topic in autonomous robotics, that has been addressed over the years from several perspectives. Most research has focused on terrestrial robots equipped with either LiDAR [16], [28], [19], [2] or visual sensors [11], [23], [24], [7], [5], [15], [18], [6], [31], [17], [26]. Works that address other types of locomotion, such as underwater [14] or aerial [25], [30], [29], [9], [12], [21] vehicles, are scarcer.

From the perspective of the onboard robot sensors, a wide variety of them have been explored in the literature: monocular [31], stereo [11], LiDAR [16], [28], [19], [2], radar [27] and multimodal combinations such as LiDAR-radar [2] or stereo-inertial [13], [21]. Up to our knowledge, the approach we formulate in this paper using 2D LiDAR and wheel encoders has not been addressed in the literature. In particular, the simultaneous self-calibration of the platform parameters, that we propose in this work, is also novel with respect to previous work.

III. Method Overview

The proposed navigation method builds on the T&R method presented in [20], [21]. In T&R, a target path is first demonstrated during the *teach* phase and is later autonomously followed by a vehicle during the *repeat* phase. Incremental localization is used during both stages in order to build a map composed of keyframes. During the teach phase, these keyframes define the target path. During the *repeat* phase, the live map is continuously localized against the reference map built during the previous phase. The robot can then be controlled to follow the target path.

In this work we extend the base T&R framework to target ground robots featuring wheel encoders and a LiDAR sensor, while following the same relative formulation of the problem where the map is composed of chained relative transforms. In this context, odometric information arising from wheel encoders is included in the estimation following the well-known preintegration approach [10], [8], which seamlessly integrates with the relative formulation. Furthermore, to improve the odometric information we include online estimation of wheel radii. For the LiDAR sensor we choose a simple and effective approach based on the Iterative Closest Point (ICP) [1], [4] scan-matching algorithm. By computing the relative transform from a pair of LiDAR point-clouds corresponding to consecutive keyframes we obtain a relative-pose measurement. We also show how to obtain the corresponding uncertainty of the ICP algorithm.

IV. Incremental Localization

The goal of incremental localization is to estimate the current pose of the robot relative to the most recent keyframe in the map, while simultaneously recording a set of keyframes

[1]Instituto de Ciencias de la Computación (ICC-CONICET) Universidad de Buenos Aires, Buenos Aires, Argentina sbedin@dc.uba.ar, mnitsche@icc.fcen.uba.ar

[2]I3A, Universidad de Zaragoza, Spain jcivera@unizar.es

979-8-3503-0704-7/23/$31.00

that describe the robot's path, and calibrate the intrinsic platform parameters. As mentioned earlier, we utilize a relative approach, which means that the state to be estimated is defined as the relative pose between the current keyframe and the previous keyframe in the path

$$\mathbf{x} = \left\{ \left\{ \mathbf{T}_i^{i+1} \right\}_{i=1\ldots n-1}, \mathbf{r} \right\} \ , \tag{1}$$

where

$$\mathbf{T}_i^{i+1} = \begin{bmatrix} \mathbf{R}_i^{i+1} & \mathbf{t}_i^{i+1} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathrm{SE}(3)$$

stands for the relative transformation between two consecutive keyframes denoted as i and $i+1$, and $\mathbf{r} \in \mathbb{R}^4_{>0}$ stands for the unknown wheel radii in our case of a four-wheel omnidirectional drive robot.

A. Iterative Closest Point

Given a pair of point clouds $\mathbf{P}$ and $\mathbf{Q}$ in two different local frames, the ICP (Iterative Closest Point) algorithm enables us to find a relative transform $\mathbf{L} \in \mathrm{SE}(3)$ that aligns $\mathbf{P}$ with $\mathbf{Q}$ with minimal error. In other words, we aim to find the transform $\mathbf{L}$ that satisfies the following equation

$$\mathbf{L} = \underset{\mathbf{L'}}{argmin}\ C\left(\mathbf{Z}, \mathbf{L'}\right) \ , \tag{2}$$

where $\mathbf{Z} = \begin{bmatrix} \mathbf{P} & \mathbf{Q} \end{bmatrix}$ and

$$C\left(\mathbf{Z}, \mathbf{L}\right) = \sum_{j=1}^{n} \left[\left(\mathbf{R}\mathbf{p}_j + \mathbf{t} - \mathbf{q}_j \right) \mathbf{n}_j \right]^2 \tag{3}$$

corresponds to the *point-to-plane* error metric defined in [4]. The point clouds $\mathbf{P} = \left\{ \mathbf{p}_j \right\}_{j=1\ldots n}$, $\mathbf{Q} = \left\{ \mathbf{q}_j \right\}_{j=1\ldots n}$ contain 3D points $\mathbf{p}_j, \mathbf{q}_j \in \mathbb{R}^3$, and $\mathbf{n}_j \in \mathbb{P}^3$ stand for the normal of a plane built from the k nearest neighbors of $\mathbf{q}_j$.

B. LiDAR Residual

In the context of LiDAR-based robot localization, the ICP algorithm can be utilized to determine the robot's motion from a pair of scans acquired from two distinct positions. To establish the correspondence between the points in both point clouds, a matching process is carried out by selecting the nearest neighbor in one cloud from the other (where non-matching points are ignored).

We model the LiDAR sensor residual as follows:

$$\mathbf{r}_{\mathbf{\Delta L}} = \log\left(\left(\mathbf{\Delta L}_i^{i+1} \right)^{-1} \mathbf{T}_i^{i+1} \right)^{\vee} \tag{4}$$

where $\mathbf{\Delta L}_i^{i+1}$ is obtained using the ICP algorithm (as described in the previous section) from point-cloud readings acquired on keyframes i and $i+1$. As an initial seed for the ICP minimization we use the relative transform obtained from odometric readings integrated between i and $i+1$ (see following section).

To model the uncertainty of $\mathbf{\Delta L}$ (where we drop indices for clarity), we adopt the approach outlined in [3]. Specifically, we take into account the initial uncertainty in LiDAR readings by propagating their covariance through a first-order approximation:

$$\mathbf{\Sigma}_{\mathbf{\Delta L}} = \mathbf{J}_{\mathbf{Z}}^{\mathbf{\Delta L}} \mathbf{\Sigma}_{\mathbf{Z}} \left(\mathbf{J}_{\mathbf{Z}}^{\mathbf{\Delta L}} \right)^{\top} \ , \tag{5}$$

where $\mathbf{\Sigma}_{\mathbf{Z}}$ is the covariance of the point clouds, that is given by the LiDAR sensor specifications sheet.

While $\mathbf{\Delta L}$ does not present a closed form, as it is obtained as a result of a minimization algorithm (2), $\mathbf{\Delta L}$ and $\mathbf{Z}$ are related by an implicit function. Furthermore, we know the gradient of the cost function C is zero at L:

$$\frac{\delta C}{\mathbf{L}} = \mathbf{0}^T \tag{6}$$

As a result, as demonstrated in [3] we can apply the implicit function theorem and finally obtain $\mathbf{J}_{\mathbf{Z}}^{\mathbf{\Delta L}}$:

$$\mathbf{J}_{\mathbf{Z}}^{\mathbf{\Delta L}} = -\left(\frac{\delta^2 C\left(\mathbf{L}, \mathbf{Z}\right)}{\delta \mathbf{L}^2} \right)^{-1} \frac{\delta^2 C\left(\mathbf{L}, \mathbf{Z}\right)}{\delta \mathbf{Z} \delta \mathbf{L}} \bigg|_{\mathbf{L}=\mathbf{\Delta L}} \tag{7}$$

C. Odometry Preintegration

To include odometric information to the estimation we follow the preintegration approach originally proposed for inertial measurements [10]. First, we define an encoder $\mathbf{q}$ reading as

$$\mathbf{q} = \begin{bmatrix} \varphi_{fr} & \varphi_{fl} & \varphi_{rl} & \varphi_{rr} \end{bmatrix}^T \tag{8}$$

where φ_{fr}, φ_{fl}, φ_{rl} and φ_{rr} represent the rotational increments for front-right, front-left, rear-left and rear-right wheel respectively, as measured by the encoders.

We then define the kinematic model of the omnidirectional-drive platform $\mathbf{K}(\mathbf{c})$ as:

$$\mathbf{K}(\mathbf{c}) = \frac{1}{4} \begin{bmatrix} r_{fr} & r_{fl} & r_{rl} & r_{rr} \\ r_{fr} & -r_{fl} & r_{rl} & -r_{rr} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{r_{fr}}{d} & \frac{-r_{fl}}{d} & \frac{-r_{rl}}{d} & \frac{r_{rr}}{d} \end{bmatrix} \tag{9}$$

with

$$\mathbf{c} = \begin{bmatrix} \mathbf{r} & d \end{bmatrix}^{\top}$$
$$d = \frac{d_1 + d_2}{2}$$

where d_1 and d_2 represent the horizontal and lateral separation of the wheels, respectively.

The incremental motion of the robot, denoted as $\mathbf{O}(\mathbf{q}, \mathbf{c})$, is then computed as follows:

$$\mathbf{O}(\mathbf{q}, \mathbf{c}) = \exp\left(\mathbf{b}(\mathbf{q}, \mathbf{c})^{\wedge} \right) \in \mathrm{SE}(3) \tag{10}$$

$$\mathbf{b}(\mathbf{q}, \mathbf{c}) = \mathbf{K}(\mathbf{c})\mathbf{q} \in se(3) \tag{11}$$

For successive encoder readings $\mathbf{q}_1, \ldots, \mathbf{q}_n$ acquired between keyframes i and $i+1$ we can then construct a preintegrated measurement $\mathbf{\Delta T}_i^{i+1}$ as:

$$\mathbf{\Delta T}_i^{i+1} = \mathbf{O}(\mathbf{q}_1, \mathbf{c}) \ldots \mathbf{O}(\mathbf{q}_n, \mathbf{c}) \tag{12}$$

$$= \mathbf{O}_1^{n-1} \mathbf{O}(\mathbf{q}_n, \mathbf{c}) \tag{13}$$

Finally, the odometry preintegration residual can be defined as:

$$\mathbf{r}_{\mathbf{\Delta T}} = \log\big((\mathbf{\Delta T}_i^{i+1})^{-1}\mathbf{T}_i^{i+1}\big)^{\vee} \tag{14}$$

D. Wheel Radius Estimation

To perform online estimation of the wheel radii $\mathbf{r}$, we can adopt a similar approach as described in [10] for estimating IMU bias terms. Specifically, we can pre-integrate the odometry readings using an initial value of $\bar{\mathbf{r}}$, and incorporate a correction term to account for updates to $\mathbf{r}$ during the estimation process. Thus, we can define:

$$\bar{\mathbf{c}} = \begin{bmatrix} \bar{\mathbf{r}} & d \end{bmatrix}^T \tag{15}$$

where d is obtained prior to estimation and kept fixed, as it is not expected to change over time. Similarly to [10], and dropping the indices on $\mathbf{\Delta T}_i^{i+1}$ for clarity, we use the following approximation for (13):

$$\mathbf{\Delta T}(\mathbf{q}, \mathbf{c}) \simeq \mathbf{\Delta T}(\mathbf{q}, \bar{c})\mathbf{J}_{\mathbf{c}}^{\mathbf{\Delta T}}(\mathbf{q}, \mathbf{c} - \bar{\mathbf{c}}) \tag{16}$$

where $\bar{\mathbf{c}}$ is the value of $\mathbf{c}$ at the moment the preintegration measurement is initialized and

$$\mathbf{J}_{\mathbf{c}}^{\mathbf{\Delta T}} = \mathbf{J}_{\mathbf{O}_1^{n-1}}^{\mathbf{\Delta T}}\mathbf{J}_{\mathbf{c}}^{\mathbf{O}_1^{n-1}} + \mathbf{J}_{\mathbf{c}}^{\mathbf{O}} \tag{17}$$

where

$$\mathbf{J}_{\mathbf{O}_1^{n-1}}^{\mathbf{\Delta T}} = \mathbf{J}_{\mathbf{O}_1^{n-1}}^{\boxplus} \tag{18}$$

$$\mathbf{J}_{\mathbf{c}}^{\mathbf{O}} = \mathbf{J}_r(\mathbf{b})\,\mathbf{J}_{\mathbf{c}}^{\mathbf{b}} \tag{19}$$

$$\mathbf{J}_{\mathbf{c}}^{\mathbf{b}} = \frac{1}{4}\begin{bmatrix} \varphi_{fr} & \varphi_{fl} & \varphi_{rl} & \varphi_{rr} \\ \varphi_{fr} & -\varphi_{fl} & \varphi_{rl} & -\varphi_{rr} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{\varphi_{fr}}{d} & \frac{-\varphi_{fl}}{d} & \frac{-\varphi_{rl}}{d} & \frac{\varphi_{rr}}{d} \end{bmatrix} \tag{20}$$

with $\mathbf{J}_r(\mathbf{b})$ the right jacobian of SE(3) evaluated at $\mathbf{b}$.

E. Odometry Covariance

We model the noise in incremental encoders as

$$\mathbf{q} = \tilde{\mathbf{q}} + \boldsymbol{\epsilon}_{\mathbf{q}} \tag{21}$$

where $\tilde{\mathbf{q}}$ is the actual encoder measurement and $\boldsymbol{\epsilon}_{\mathbf{q}} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{q}})$. Furthermore, since we assume a previously estimated value for d, we also consider its contribution to the propagated covariance as $d = \tilde{d} + \epsilon_d$, $\epsilon_d \sim \mathcal{N}(0, \sigma_d)$.

We can then propagate the covariance of a single integration step (16) as:

$$\mathbf{\Sigma}_{\mathbf{O}} = \mathbf{J}_{\mathbf{q}}^{\mathbf{O}}\mathbf{\Sigma}_{\mathbf{q}}(\mathbf{J}_{\mathbf{q}}^{\mathbf{O}})^{\top} + \mathbf{J}_{\mathbf{c}}^{\mathbf{O}}\mathbf{\Sigma}_{\mathbf{c}}(\mathbf{J}_{\mathbf{c}}^{\mathbf{O}})^{\top} \tag{22}$$

where $\mathbf{\Sigma}_{\mathbf{c}} = \mathbf{I}_5\,[\mathbf{0}\ \sigma_d]^{\top}$ and $\mathbf{J}_{\mathbf{q}}^{\mathbf{O}} = \mathbf{J}_{\mathbf{b}}^{\mathbf{O}}\mathbf{J}_{\mathbf{q}}^{\mathbf{b}} = \mathbf{J}_r(\mathbf{b})\mathbf{K}(\mathbf{c})$. Note that the wheel radii covariance are not considered in this step since these variables are part of the estimation, in contrast to the wheel's separations which is prior information. The wheel radii covariance are obtained as a result of the estimation process.

The covariance for a single increment can then be accumulated to the complete preintegrated measurement (13) in a similar fashion as done in [10].

F. Repeat

For the repeat phase, we adopt a similar methodology to that outlined in [21], albeit with adaptations to accommodate the different sensor modality and robotic platform involved. We provide first a brief overview of the standard map localization approach.

The repeat phase comprises two distinct localization procedures: 1) the incremental localization method, which has already been described in the previous section, and 2) the global localization process in the map estimated at the teach phase. To perform global localization, we select the keyframe K_m from the teach map that is closest to the most recent keyframe K_l in the repeat phase, which is then designated as the *map reference* keyframe. This keyframe is subject to continuous change as the robot moves and new keyframes are generated during the incremental localization process in the repeat phase.

By executing a scan-matching procedure that utilizes the scans from both keyframes K_m and K_l, we can determine the relative transform $\mathbf{T}_m^l$ between them. Subsequently, to establish the current pose of the robot with respect to the earlier map, we combine $\mathbf{T}_m^l$ with the most recent transform $\mathbf{T}_l^n$ in incremental localization at the repeat phase (last pose in (1)) as:

$$\mathbf{T}_m^n = \mathbf{T}_k^l\mathbf{T}_l^n \tag{23}$$

This separation into two distinct localization threads allows us to keep an updated pose $\mathbf{T}_k^n$ at all times, even during temporary failure of the global localization. In such case, the previously found pose $\mathbf{T}_k^*$ is extended (predicted) with newer relative transforms $\mathbf{T}_n^{n+1}$, $\mathbf{T}_{n+1}^{n+2}$, etc. This adds robustness to the whole T&R framework and even allows for deviations from the teach map if they were necessary (for example, to avoid an obstacle not previously present).

From the set of keyframes close to the map-reference keyframe a smooth trajectory is built which takes into account the linear and rotational velocities recorded during the teach phase. This allows to evaluate the trajectory at any given time t and obtain a control setpoint in order to smoothly follow the taught path (for more details see [21]).

As this work's contributions primarily concerns the incremental localization algorithm, we opted for a straightforward control strategy based on a position/orientation PI controller for our experiments. Such controller operates independently on each of the three controllable axes of the robot.

V. Results

A. Experimental Setup

To evaluate the performance of the proposed T&R framework, we conducted a series of experiments in the Coppelia V-Rep simulator. The robotic platform used was the KUKA youBot, which features an omnidirectional drive platform consisting of four mechanum-style wheels. Simulated wheel encoder readings were obtained by converting wheel rotation into 2000 discrete ticks. For the LiDAR we chose the Hokuyo URG-04LX sensor which gives readings at 50 Hz, up to a distance of 4 m, over a 240 ° angle. We introduce noise to the

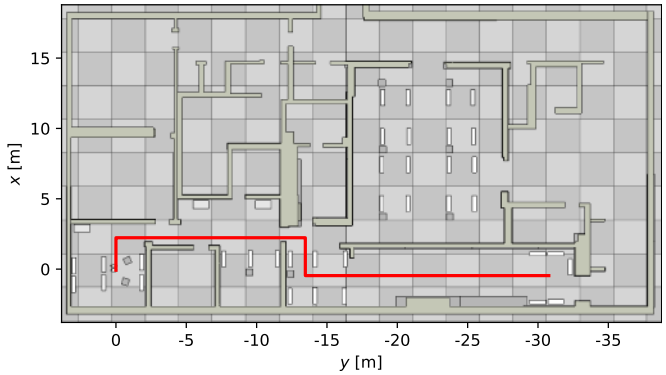


Fig. 1. Top view of the simulation environment and teach trajectory.

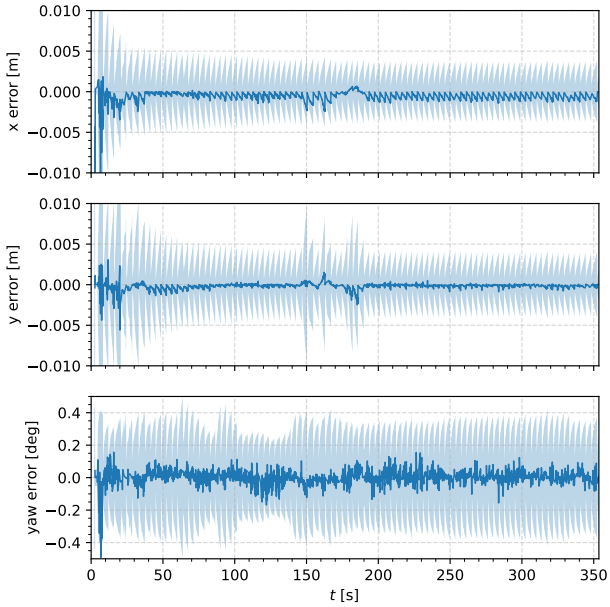


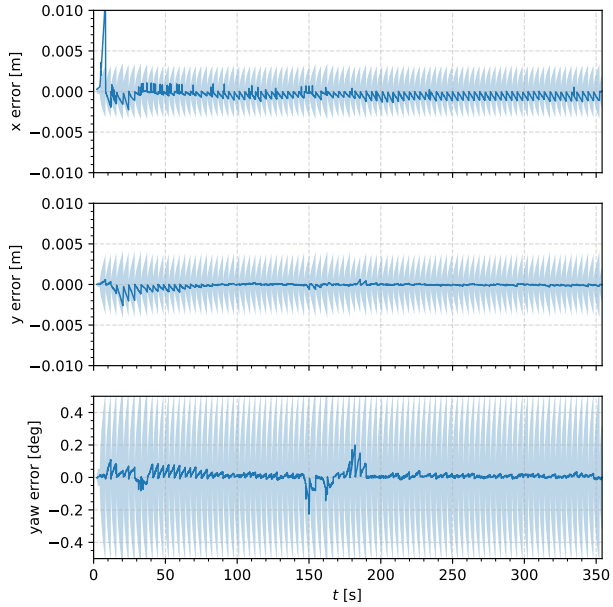
Fig. 2. Relative pose errors (in dark blue) and estimated uncertaintes (in light blue).

LiDAR scans by perturbing the distance of each reading with Gaussian noise, with an uncertainty of 2 % over distance, as proposed in [22].

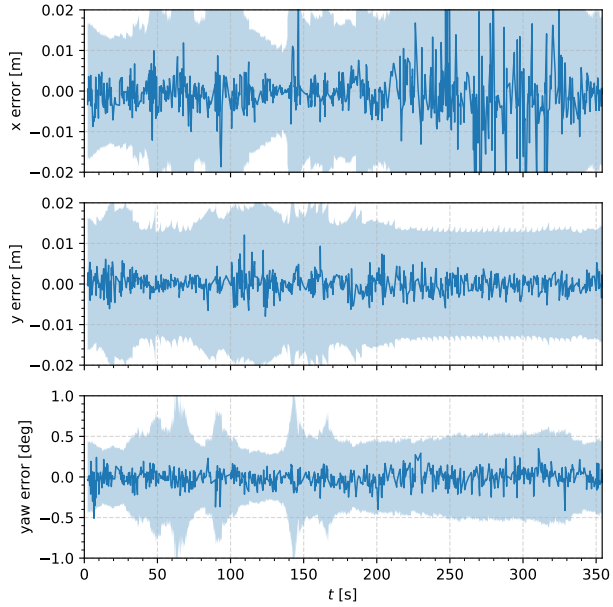
The simulated environment corresponds to a model of a region of the Marcum Conference Center of the University of Miami (see Figure 1). The experiments consist of an initial teach run over the environment and subsequent closed-loop runs of the repeat phase.

B. Incremental Localization and Online Calibration

In this section, we assess the quality of the incremental localization by evaluating the error of the latest estimated relative transform against the ground truth as well as the values of the estimated wheel radii (starting with an initial error of 2 mm for each wheel). Figure 2 displays the pose error (in dark blue), along with the corresponding pose covariance (illustrated with a 99% confidence interval in light blue). Additionally, we present the relative pose $\mathbf{\Delta T}$ and $\mathbf{L}$, along with their covariance (fig. 3(a), 3(b)), which provides insight into the performance of each sensor modality over



(a) Relative pose errors (in dark blue) and estimated uncertainties (in light blue) for odometry preintegration.



(b) Relative pose errors (in dark blue) and estimated uncertainties (in light blue) for ICP.

Fig. 3. Relative poses from odometry preintegration and ICP.

time and its relationship with the final estimated pose. Lastly, Figure 4 illustrates the error and estimated uncertainties of the wheel radii during online calibration.

The aforementioned results show that, in general, the localization errors are small and consistent with the estimated uncertainties. Upon analyzing the convergence of the wheel radii, it can be observed that the initial error is quickly reduced. Between the time intervals of $t = 255\,\mathrm{s}$ and $t = 307\,\mathrm{s}$, when the robot navigates through a long hallway, the error in the x direction as well as the uncertainty of the pose obtained from ICP significantly increases. Nevertheless, since the pose obtained from odometry preintegration is

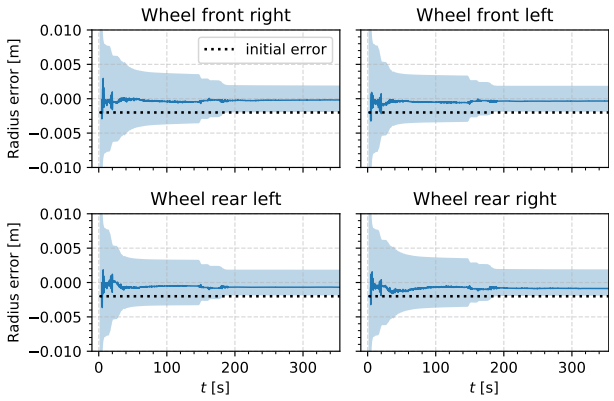


Fig. 4. Wheel radii estimation results (errors in dark blue, uncertainties in light blue).

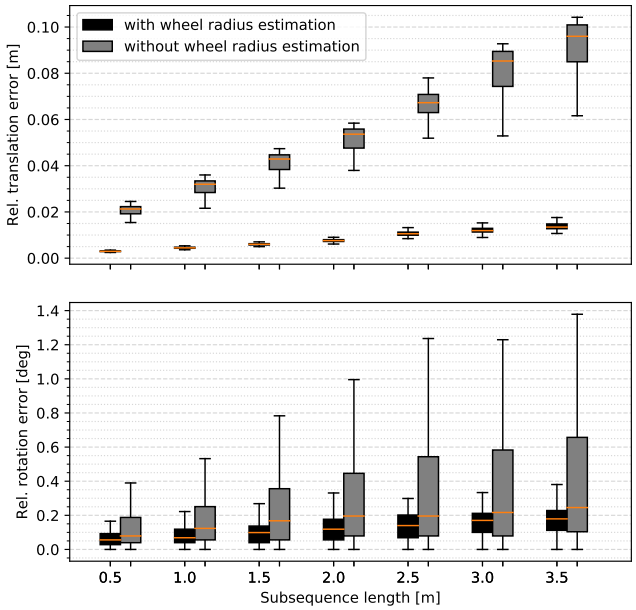


Fig. 5. Box-and-whisker diagrams showing the Relative Pose Error (RPE) on subsequences of increasing length, with and without wheel radii estimation. Observe the substantial improvement when wheel radii is estimated.

unaffected by this situation, the pose estimate after fusion has a low error that is consistent with the low estimated variance.

We conducted further analysis of the localization error by measuring the Relative Pose Error (RPE), as defined in [21], for subsequences of increasing lengths (Figure 5), for both cases of having online estimation of wheel radii enabled and disabled. Since our approach utilizes a relative method and odometric reading integration accumulates error between a pair of keyframes, this analysis enables us to observe the rate at which error accumulates with respect to distance. By doing so, we can make a trade-off between localization and path accuracy when deciding at what distance to place keyframes. From Figure 5 we can observe the impact on odometry motion estimation arising from a relatively small error in wheel radii as well as how the overall localization performance improves when performing online estimation

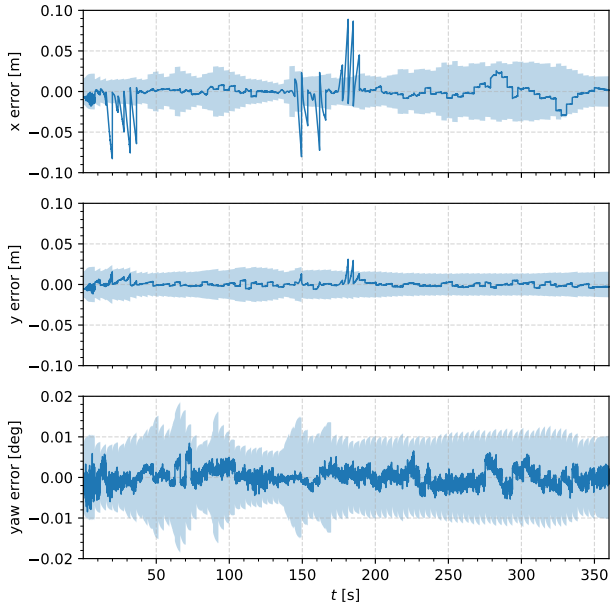


Fig. 6. Error (dark blue) and estimated uncertainties (light blue) between relative poses in the repeat phase.

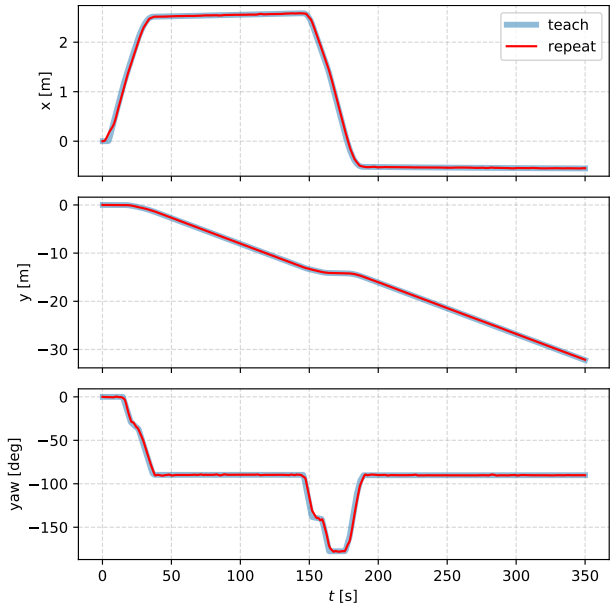


Fig. 7. Teach and repeat trajectories. Observe that, using our approach, the repeat trajectory closely follows the teach one.

of these parameters. This is true even for relatively short distances and particularly for the translational component.

C. Repeat Autonomous Navigation

For the repeat phase we let the system control the robot motion in order to analyze the localization against the prior map built during the teach phase analyzed in the previous section. For this we present the error between the robot's relative pose with respect to the prior map (as explained in section IV-F, for more details see [20]) compared to ground truth information, as well as its uncertainty (see figure 6). We also compare the path followed by the robot compared

to that of the teach phase in Figure 7.

Similarly to the results obtained for the teach phase we can see that the error in the robot pose with respect to the prior map is generally small and the estimated uncertainties are consistent with it. Furthermore we can see that the robot successfully follows the previously taught path.

VI. Conclusions

In this work we presented a novel T&R framework for omnidirectional robot platforms, fusing information from LiDAR and wheel encoder sensors under a relative formulation. Our results demonstrate that the integration of both sensor modalities not only enables robust localization in environments where LiDAR sensors often struggle, but also facilitates the real-time estimation of the platform's wheel radii. This, in turn, significantly enhances the accuracy of motion estimation based on the odometric model.

As part of our future work, we plan to conduct live experiments to validate our approach under real-world conditions. Additionally, we aim to enhance our online calibration approach to accommodate changes in wheel radii that may arise due to variations in cargo loads during transport operations. We also intend to refine our robot control strategy to further improve the method's performance.

ACKNOWLEDGMENT

This work was funded by the Spanish projects PID2021-127685NB-I00 and TED2021-131150B-I00 and the Aragón Government project DGA_FSE-T45_20R.

References

[1] Paul Besl and Neil McKay. Method for Registration of 3-D Shapes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 239–256, 1992.

[2] Keenan Burnett, Yuchen Wu, David J Yoon, Angela P Schoellig, and Timothy D Barfoot. Are we ready for radar to replace lidar in all-weather mapping and localization? *IEEE Robotics and Automation Letters*, 7(4):10328–10335, 2022.

[3] Andrea Censi. An accurate closed-form estimate of icp's covariance. In *IEEE International Conference on Robotics and Automation*, page 3167–3172, 2007.

[4] Yang Chen and Gerard Medioni. Object Modeling by Registration of Multiple Range Images. In *IEEE Conf. on Robotics and Au- tomation*, 1991.

[5] Lee Clement, Jonathan Kelly, and Timothy D Barfoot. Robust monocular visual teach and repeat aided by local ground planarity and color-constant imagery. *Journal of Field Robotics*, 34(1):74–97, 2017.

[6] Dominic Dall'Osto, Tobias Fischer, and Michael Milford. Fast and robust bio-inspired teach and repeat navigation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 500–507. IEEE, 2021.

[7] Julie Dequaire, Chi Hay Tong, Winston Churchill, and Ingmar Posner. Off the beaten track: Predicting localisation performance in visual teach and repeat. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 795–800. IEEE, 2016.

[8] Jeremie Deray, Joan Solà, and Juan Andrade-Cetto. Joint on-manifold self-calibration of odometry model and sensor extrinsics using pre-integration. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2019.

[9] Marius Fehr, Thomas Schneider, Marcin Dymczyk, Jürgen Sturm, and Roland Siegwart. Visual-Inertial Teach and Repeat for Aerial Inspection. page arXiv:1803.09650, Mar 2018.

[10] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2017.

[11] Paul Furgale and Timothy D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, (2006):1–27, 2010.

[12] Fei Gao, Luqi Wang, Kaixuan Wang, William Wu, Boyu Zhou, Luxin Han, and Shaojie Shen. Optimal trajectory generation for quadrotor teach-and-repeat. *IEEE Robotics and Automation Letters*, 4(2):1493–1500, 2019.

[13] Fei Gao, Luqi Wang, Boyu Zhou, Xin Zhou, Jie Pan, and Shaojie Shen. Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments. *IEEE Transactions on Robotics*, 36(5):1526–1545, 2020.

[14] Peter King, Andrew Vardy, and Alexander L Forrest. Teach-and-repeat path following for an autonomous underwater vehicle. *Journal of Field Robotics*, 35(5):748–763, 2018.

[15] Tomáš Krajník, Filip Majer, Lucie Halodová, and Tomáš Vintr. Navigation without localisation: reliable teach and repeat based on the convergence theorem. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1657–1664. IEEE, 2018.

[16] Philipp Krüsi et al. Lighting-invariant Adaptive Route Following Using Iterative Closest Point Matching. *JFR*, 32(4):534–564, 2015.

[17] Mohammad Mahdavian, KangKang Yin, and Mo Chen. Robust visual teach and repeat for ugvs using 3d semantic maps. *IEEE Robotics and Automation Letters*, 7(4):8590–8597, 2022.

[18] Matias Mattamala, Milad Ramezani, Marco Camurri, and Maurice Fallon. Learning camera performance models for active multi-camera visual teach and repeat. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14346–14352. IEEE, 2021.

[19] Colin McManus, Paul Furgale, Braden Stenning, and Timothy D Barfoot. Lighting-invariant visual teach and repeat using appearance-based lidar. *Journal of Field Robotics*, 30(2):254–287, 2013.

[20] Matias Nitsche, Facundo Pessacg, and Javier Civera. Visual-inertial teach & repeat for aerial robot navigation. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2019.

[21] Matias Nitsche, Facundo Pessacg, and Javier Civera. Visual-inertial teach and repeat. *Robotics and Autonomous Systems*, 131:103577, 2020.

[22] Yoichi Okubo, Cang Ye, and Johann Borenstein. Characterization of the Hokuyo URG-04LX laser rangefinder for mobile robot obstacle negotiation. In *SPIE Unmanned Systems Technology XI*, 2009.

[23] Chris J Ostafew, Angela P Schoellig, and Timothy D Barfoot. Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 176–181. IEEE, 2013.

[24] Michael Paton, Kirk MacTavish, Michael Warren, and Timothy D. Barfoot. Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat. In *IROS*, 2016.

[25] Andreas Pfrunder, Angela P.= Schoellig, and Timothy D Barfoot. A Proof-of-Concept Demonstration of Visual Teach and Repeat on a Quadrocopter Using an Altitude Sensor and a Monocular Camera. In *CRV*, 2014.

[26] Zdeněk Rozsypálek, Tomáš Rouček, Tomáš Vintr, and Tomáš Krajník. Multidimensional particle filter for long-term visual teach and repeat in changing environments. *IEEE Robotics and Automation Letters*, 8(4):1951–1958, 2023.

[27] Ştefan Săftescu, Matthew Gadd, Daniele De Martini, Dan Barnes, and Paul Newman. Kidnapped radar: Topological radar localisation using rotationally-invariant metric learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4358–4364. IEEE, 2020.

[28] Christoph Sprunk, Gian Diego Tipaldi, Andrea Cherubini, and Wolfram Burgard. Lidar-based teach-and-repeat of mobile robot trajectories. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3144–3149. IEEE, 2013.

[29] Michael Warren, Melissa Greeff, Bhavit Patel, Jack Collier, Angela P Schoellig, and Timothy D Barfoot. There's No Place Like Home: Visual Teach and Repeat for Emergency Return of Multirotor UAVs During GPS Failure. *IEEE Robotics and Automation Letters*, 4(1):161–168, jan 2019.

[30] Warren Warren, Michael Paton, Kirk MacTavish, Angela P Schoellig, and Timothy D Barfoot. Towards Visual Teach and Repeat for GPS-Denied Flight of a Fixed-Wing UAV. In Marco Hutter and Roland Siegwart, editors, *Field and Service Robotics*, pages 481–498, 2018.

[31] Cheng Zhao, Li Sun, Tomáš Krajník, Tom Duckett, and Zhi Yan. Monocular teach-and-repeat navigation using a deep steering network with scale estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2613–2619. IEEE, 2021.

Adaptive Compliant Robot Control with Failure Recovery for Object Press-Fitting

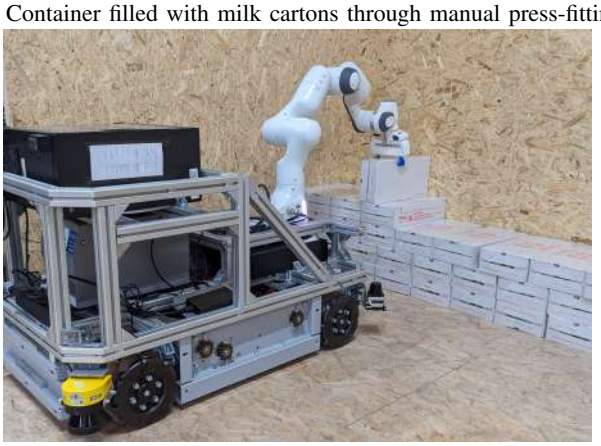
Ekansh Sharma[†], Christoph Henke[‡], Alex Mitrevski[†], and Paul G. Plöger[†]

***Abstract*— Loading of shipping containers for dairy products often includes a press-fit task, which involves manually stacking milk cartons in a container without using pallets or packaging. Automating this task with a mobile manipulator can reduce worker strain, and also enhance the efficiency and safety of the container loading process. This paper proposes an approach called Adaptive Compliant Control with Integrated Failure Recovery (ACCIFR), which enables a mobile manipulator to reliably perform the press-fit task. We base the approach on a demonstration learning-based compliant control framework, such that we integrate a monitoring and failure recovery mechanism for successful task execution. Concretely, we monitor the execution through distance and force feedback, detect collisions while the robot is performing the press-fit task, and use wrench measurements to classify the direction of collision; this information informs the subsequent recovery process. We evaluate the method on a miniature container setup, considering variations in the (i) starting position of the end effector, (ii) goal configuration, and (iii) object grasping position. The results demonstrate that the proposed approach outperforms the baseline demonstration-based learning framework regarding adaptability to environmental variations and the ability to recover from collision failures, making it a promising solution for practical press-fit applications.**

I. INTRODUCTION

The process of loading shipping containers with general cargo typically involves a press-fitting task, wherein the products are tightly packed with minimal clearance, as depicted in Fig. 1a. Press-fitting eliminates the need for additional packaging, such as pallets, which can be costly and take up valuable space within shipping containers. For dairy products, the task is usually performed manually in artificially cooled environments to ensure product quality. This manual process can, however, be physically demanding for workers, who must repetitively stack and push milk cartons. Furthermore, a shortage of skilled labor for this type of work is a persistent challenge in the industry [1]. Automating the task using a mobile manipulator, as depicted in Fig. 1b, can offer significant benefits in terms of efficiency and productivity; in particular, robots can work continuously without fatigue in temperature-controlled environments, resulting in a more reliable loading process.

(a) Container filled with milk cartons through manual press-fitting



(b) Custom mobile manipulator developed at the Institute for Business Cybernetics for automating the process

Fig. 1: Shipping container loading process

Performing the task with a mobile manipulator for loading shipping containers can be challenging due to the need for contact-rich object manipulation and susceptibility to environmental variations and collisions. Existing research on the peg-in-hole assembly task can, however, provide valuable insights for developing a strategy to overcome these challenges and perform press-fit tasks. For instance, learning-

*This work was conducted at the Institute for Business Cybernetics e.V as part of the MASON project (https://mason-projekt.de/), which is supported by the Federal Ministry for Economic Affairs and Climate Action under the funding IGF-Project no. 22403. The work was also supported by the b-it foundation.

[†]Autonomous Systems Group, Department of Computer Science, Hochschule Bonn-Rhein-Sieg, Sankt Augustin, Germany ekansh.sharma@smail.inf.h-brs.de, <aleksandar.mitrevski, paul.ploeger>@h-brs.de

[‡]Institute for Business Cybernetics e.V. at the RWTH Aachen University, Aachen, Germany christoph.henke@ifu.rwth-aachen.de

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

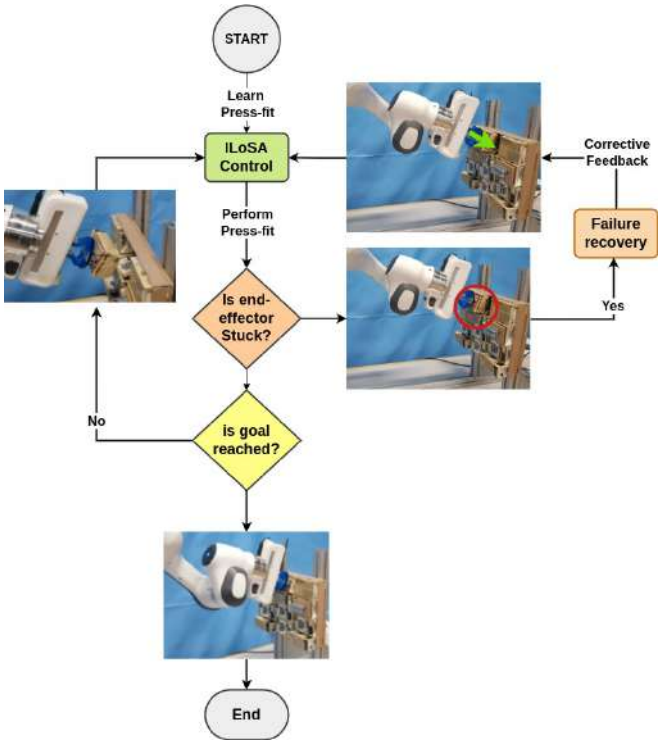


Fig. 2: Press-fitting of milk cartons by a mobile manipulator using the proposed Adaptive Compliant Control with Integrated Failure Recovery (ACCIFR). The system comprises three key components: (i) the ILoSA framework for learning and execution (green), (ii) press-fit monitoring via force and distance feedback (yellow), and (iii) a failure recovery mechanism to detect and recover from collisions (orange).

based approaches, such as reinforcement learning [2], [3], [4], [5], learning from demonstration [6], [7], [8] and contact-state recognition [9], [10] have been successfully applied to learn and perform a peg-in-hole task. Such approaches can be adapted to tackle the press-fit task, as press-fitting can be seen as an instance of the peg-in-hole task. Most of these methods require a large amount of data, however, which can be difficult to collect in real industrial environments and, furthermore, typically do not incorporate mechanisms that enable failure detection and subsequent recovery. This, in turn, limits the applicability of existing approaches to real-world press-fitting scenarios.

This paper presents an approach that we refer to as Adaptive Compliant Control with Integrated Failure Recovery (ACCIFR) to learn and perform a press-fit task using a mobile manipulator. Our approach, illustrated in Fig. 2, is built upon the ILoSA framework [11] to learn variable impedance policies from a single demonstration and subsequent user corrections, such that we incorporate a press-fit monitoring system into our approach, which utilizes force and position-based feedback to ensure the reliable press-fitting of milk cartons. Furthermore, our approach extends the ILoSA framework by integrating a failure recovery mechanism that can automatically detect and recover from collisions, inspired by [12]. Concretely, when a collision event occurs, a classifier predicts the contact side using time-series wrench[1] data, while the recovery mechanism provides corrective feedback based on the predicted collision side, facilitating effective collision recovery. We demonstrate the generalizability of the proposed approach by performing press-fit tasks using a Franka Emika manipulator and a small container setup, considering variations in the starting position of the end effector, the goal configuration, and the object grasping position. We also evaluate the accuracy of our contact-state recognition classifier in predicting the contact side over varying lengths of time-series history. The results show that ACCIFR improves the performance of the baseline ILoSA, thus suggesting that our approach enables a robot manipulator to learn and perform practical press-fit tasks with the ability to reliably generalize over different environmental variations and recover from collisions using its failure recovery mechanism.

II. RELATED WORK

The press-fit task is a challenging problem that has received relatively little attention in the context of robotic solutions; however, existing research on the extensively studied peg-in-hole assembly task [13], [4], [14] can provide valuable insights to address this problem. In theory, the press-fit task can be considered as a variation of the peg-in-hole task, where a carton is inserted into a confined space inside a container. In [15], existing robotics peg-in-hole strategies are classified into two main categories: contact model-based and contact model-free approaches, where the former employ contact-state recognition with compliant control, while the latter use learning through environment interaction or learning from demonstrations.

Contact-state recognition allows robots to perform manipulation tasks by adjusting their behavior based on the current contact situation. This can be achieved through analytical modeling [16] or statistical modeling [17]. Conceptually, analytical modeling for a press-fit task is challenging due to the complexity of modeling numerous constraints, especially the soft body characteristics of the carton for in-contact manipulation, and is generally susceptible to uncertainty. In contrast, statistical modeling directly learns the relationship from gathered samples to predict the contact state without requiring task-specific information. Yan et al. [9] introduce a supervised learning-based contact-state recognition model using support vector machines. This system receives force and torque input from sensors and outputs the corresponding impedance controller and skill parameters to adjust the pose and orientation of the robot's end effector. Jasim and Plapper [10] use a Gaussian mixture model and the expectation-maximization algorithm to model input observations (wrench and pose) and estimate the contact state using Bayesian classification. Our proposed approach also uses supervised

[1]Here, wrench data refers to the measurements of forces and torques exerted on a robot's end effector. Such a measurement consists of six components — three force components and three torque components — over the (x, y, z)-axes.

learning-based contact-state recognition for failure recovery; however, in contrast to [9] and [10], we use a time-series-based classifier to estimate the contact side from a short history of raw wrench data and use this information to generate corrective feedback for failure recovery, making it simpler and more efficient to implement for a press-fit task.

Reinforcement learning (RL)-based methods are also widely used to enable robots to learn new tasks and adapt to new situations [18]. Traditional control approaches are typically combined with RL techniques to perform peg-in-hole tasks. Johannink et al. [2] combine a learnable parametrized policy with a fixed hand-engineered controller, such that the twin delayed deep deterministic policy gradient (TD3) is used as the underlying RL algorithm. Beltran-Hernandez et al. [19] use the soft actor-critic (SAC) algorithm to find a policy that generates trajectory commands and the parameters of a force controller based on an estimated goal pose. Learning manipulation skills in combination with RL to perform peg-in-hole tasks is also a common approach in the literature. While some of these frameworks use a fixed set of manipulation primitives for a specific task [4], others use a dynamic sequence of manipulation primitives automatically discovered through deep RL [5]. The sample efficiency, stability, and generalization ability for real-world press-fitting is, however, challenging; most of the above approaches thus suggest using Sim2Real with random exploration and domain randomization [20], [3]. However, developing an accurate simulation for press-fit tasks that can represent the contact-rich manipulation and soft-body characteristics of drink cartons is not trivial, while random exploration in the real world can be dangerous.

Learning from demonstration (LfD) is an alternative promising approach to learning and performing a manipulation task. LfD is often combined with RL to improve data efficiency [21], [22] due to its ability to bootstrap the agent at the start of the training. Some approaches use only a few demonstrations to learn any manipulation task. For instance, Hermann et al. [6] propose an adaptive curriculum generation framework that only requires a few expert demonstrations to learn a task in simulation and transfer it to a real robot. Similarly, Zhan et al. [7] propose a framework that utilizes only ten expert demonstrations and achieves optimal policies across several manipulation tasks. Recent work in learning from the demonstration is inclined towards one-shot demonstration learning, namely learning any manipulation task through a single demonstration. Johns [8] presents an approach that learns manipulation tasks using a single expert demonstration combined with vision-based self-supervised training from a bottleneck pose (from which the object interaction begins); the approach uses a coarse-to-fine trajectory where the robot approaches the object's bottleneck pose in a coarse manner and then interacts with the object in a fine manner by replaying the end effector velocities recorded during the demonstration. A promising alternative to visual-based imitation learning is presented by Franzese et al. [11], which introduces the Interactive Learning of Stiffness and Attractors (ILoSA) framework. ILoSA also utilizes a single demonstration and uses active user corrections to learn diverse manipulation tasks. Here, Gaussian processes are used to learn variable impedance policies, identify uncertainty regions, as well as enable interactive corrections, modulation of stiffness, and active disturbance rejection. Our proposed approach modifies and extends the ILoSA framework to learn and perform the press-fit manipulation task from a single demonstration and user correction, but integrates a monitoring and failure recovery mechanism to perform the press-fit task reliably.

III. BACKGROUND: COMPLIANT CONTROL USING ILOSA

The proposed approach is built upon ILoSA [11], an interactive imitation learning framework. In this section, we provide a short overview of ILoSA so that our approach, which is described in the next section, can be understood without ambiguities.

ILoSA uses two main teaching methods: kinesthetic demonstration and teleoperated feedback. We collect a single kinesthetic demonstration D to initialize the policy for the end-effector's intended behavior:

$$D = \{(\xi, s)\} \tag{1}$$

Here, $\xi = \{\phi_0, ..., \phi_T\}$ is a sequence of features ϕ defining the state of the robotic system, while $s = \{\Delta x^d, K_s^d\}$ represents task-specific information, such that Δx^d is the attractor distance recorded in the demonstration and K_s^d is the recorded stiffness. The acquired policy can then be executed, such that online corrections U are provided using teleoperated feedback to improve the policy:

$$U = \{(x_{_{offset}}, y_{_{offset}}, z_{_{offset}})\} \tag{2}$$

Here, $x_{_{offset}}$ is the directional feedback along the x-axis; $y_{_{offset}}$ and $z_{_{offset}}$ are defined equivalently over the y- and z-axis, respectively.

In ILoSA, a policy is learned using Gaussian processes and can thus represent uncertain regions, which allows interactive corrections and stiffness modulation. Specifically, corrections impact the parameters of the impedance controller by altering the attractor distance, represented by Δx, and the stiffness of the end effector, represented by K_s, as

$$\Lambda(q)\ddot{x} = K_s \Delta x - D\dot{x} + f_{ext} \tag{3}$$

Here, $\Lambda(q)$ is the Cartesian inertia matrix of the physical system, D is the critical damping matrix, and f_{ext} is external force. The hyperparameters of the GPs are determined using a combination of expectation-maximization and the L-BFGS method. Concretely, during the demonstration phase, the hyperparameters are optimized accordingly, but are kept constant during the correction phase, as the correlation is assumed to be invariant.

ILoSA uses an uncertainty measure to determine if the robot is in an unvisited area, in which case a corrective sample is added to a database. A correction is then distributed among all existing samples that are correlated with the current end effector position using the following update rule,

which is used to correct the attractor distance or stiffness (or both) based on interpreting the received feedback:

$$\mu(\boldsymbol{x}) = \boldsymbol{k}_*(\boldsymbol{\xi}, \boldsymbol{x})^\top \left(\boldsymbol{K}(\boldsymbol{\xi}, \boldsymbol{\xi}) + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} = \boldsymbol{A}(\boldsymbol{\xi}, \boldsymbol{x})\boldsymbol{y} \quad (4)$$

$$\Sigma = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_*(\boldsymbol{\xi}, \boldsymbol{x})^\top \left(\boldsymbol{K}(\boldsymbol{\xi}, \boldsymbol{\xi}) + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_*(\boldsymbol{\xi}, \boldsymbol{x}) \quad (5)$$

$$\mu + \epsilon_\mu = \boldsymbol{A}(\boldsymbol{\xi}, \boldsymbol{x})\left(\boldsymbol{y} + \epsilon_y\right) \Rightarrow y_{\text{new}} = \boldsymbol{y} + \boldsymbol{A}(\boldsymbol{\xi}, \boldsymbol{x})^+ \epsilon_\mu \quad (6)$$

Here, k is the variance of $\boldsymbol{x}$, K is the covariance matrix of the training points $\boldsymbol{\xi}$, $\boldsymbol{k}_*$ is the covariance between $\boldsymbol{x}$ and the training points, σ_n^2 denotes the variance of the Gaussian noise of the training inputs, and $\boldsymbol{y}$ are the training outputs. K, $\boldsymbol{k}*$, and k are all functions of a kernel, and $\boldsymbol{A}(\boldsymbol{\xi}, \boldsymbol{x})^+$ is the pseudoinverse of $\boldsymbol{A}$. The correction provided at $\boldsymbol{x}$ is represented by ϵ_μ. $\boldsymbol{A}^+$ acts as a selector that automatically adjusts the modification needed for correlated elements in the database to align with the user's desired corrections.

ILoSA uses the directional feedback provided by the user to infer changes in the attractor distance or stiffness, which allows for incremental correction of the end effector's dynamics. It concretely uses the teleoperated input feedback to directly determine the attractor distance increment Δx_{inc} , while the change in stiffness K_{s_inc} is obtained as

$$(K_s + K_{s\_inc})\Delta_{\text{lim}} = K_s \left|\Delta x + \Delta x_{\text{inc}}\right| \quad (7)$$

On its own, ILoSA can only be used to learn and execute a policy, but the method does not involve monitoring and recovery mechanisms, which limits its usefulness for practical press-fit applications. In this paper, we embed ILoSA within a monitoring and recovery framework that allows reliable execution in the press-fit context.

IV. COMPLIANT CONTROL WITH FAILURE RECOVERY

In this work, we propose an approach called Adaptive Compliant Control with Integrated Failure Recovery (ACCIFR) for learning and performing a press-fit task using a mobile manipulator. ACCIFR uses ILoSA as a baseline framework to learn a compliant policy for the press-fit task, but incorporates a collision monitoring and recovery mechanism in order to prevent the robot from getting stuck due to collisions with the environment. Our approach aims to efficiently execute the press-fit task while maintaining the ability to generalize to task variations. To achieve this, we integrate a failure recovery mechanism that uses contact-state recognition to detect and recover from collisions, namely ACCIFR uses corrective feedback to adjust its policy and recover from a collision. These features enable the ACCIFR algorithm to handle various environmental variations, recover from failures, and lead to successful press-fitting.

A. Formulation of a Press-Fit Task

In a container loading process, the press-fit task involves pushing a carton to fit at a designated spot inside the container. Formally, the objective of the press-fit task is to move the robot's end effector E from a starting pose S, following a trajectory T in space, to fit an object O at a goal pose G, as shown in Fig. 3. We assume that S and

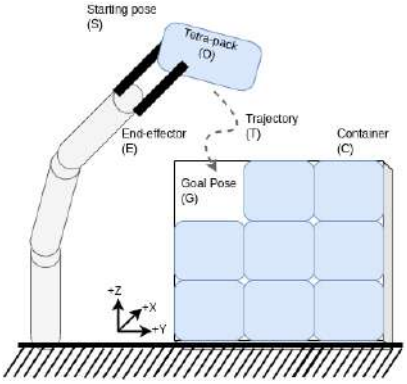


Fig. 3: Illustration of a press-fit task

the estimated goal pose G will be known a priori. Also, we assume that O is grasped in such a way that it can be pushed to fit inside the goal pose. The observable measurements are the end effector pose and the wrench data at each time step. The success of the press-fit task relies on the robot's ability to accurately navigate T and fit O into G in a compliant manner, namely while avoiding collisions with other objects and the walls of the container.

B. Press-Fit Monitoring and Recovery

To ensure that the press-fit is achieved correctly, we monitor both the distance between the current end effector pose Δx_{t-1} and the goal pose G, as well as the force applied along the robot's x-direction. The execution is deemed successful when the goal pose is reached within a predefined distance threshold, denoted as D_{th}, and a desired force threshold, denoted as F_{th}, is achieved. This helps to ensure that the object is press-fitted securely, which is essential for the successful execution of the task.

During execution of the press-fit task, the robot can experience collisions with the environment, particularly with other cartons as well as with the container edges. Our failure recovery mechanism, illustrated in Fig. 4, follows a three-step approach to detect and recover from collisions. The first step involves collision detection. Once a collision is detected, a classifier predicts the collision side based on time-series wrench data. Finally, corrective feedback recovers the end effector by driving it towards the goal. We describe each of these steps below.

1) Collision detection: ILoSA predicts and sets the attractor pose and stiffness at each timestep during active control. We utilize the generated prediction for collision detection, namely we continuously monitor the difference between Δx_{t-1} and Δx_t. If there is no difference between the current attractor pose Δx_{t-1} and the next predicted attractor pose Δx_t, and the goal pose is not reached, we consider that the end effector has either collided or is stuck.

2) Contact side prediction: To predict the side of a collision after it has been detected, we use the *InceptionTime* [23] deep learning-based time-series classification

Fig. 4: Failure recovery mechanism: (i) a collision is detected (in red), (ii) a classifier predicts the contact side, and (iii) corrective feedback steers the end effector towards the goal (in green) based on the contact-side prediction.

model.[2] We employ a deep learning classifier due to its inherent capability to automatically learn complex patterns and representations directly from the raw time-series wrench data, eliminating the need for explicit feature engineering as typically required by classical machine learning approaches. The input to the classifier, denoted as I_{wrench}, is multivariate time-series wrench data of the form

$$I_{wrench} = \begin{bmatrix} x_{f_1} & \dots & x_{f_n} \\ y_{f_1} & \dots & y_{f_n} \\ z_{f_1} & \dots & z_{f_n} \\ x_{\tau_1} & \dots & x_{\tau_n} \\ y_{\tau_1} & \dots & y_{\tau_n} \\ z_{\tau_1} & \dots & z_{\tau_n} \end{bmatrix} \tag{8}$$

where $(x_f,\ y_f,\ z_f)$ and $(x_\tau,\ y_\tau,\ z_\tau)$ are force and torque readings, respectively, sensed at the end effector, and n represents the length of the wrench data history. The output of the classifier is the predicted contact side label, denoted as $P_{contact}$. For the evaluation in this paper, we collected labeled wrench data for collisions on the left and right sides for simplicity, but the classifier can be extended if other collision categories need to be classified. For training, we collected a wrench data stream for several seconds after the collision occurred. The data was obtained from 80 trials following a collision and then randomly divided into training and test sets using an $80\%/20\%$ split ratio. The raw data was preprocessed to ensure it is in a suitable format as in Eq. 8; the classifier was trained on the preprocessed data using the cross-entropy loss function and Adam optimizer.

3) Corrective feedback: In this paper, we use a failure recovery mechanism that is inspired by [12], where, for parameters that have led to a failure of a parameterized skill, a hypothesis about the failure in terms of violated symbolic relations is found; this hypothesis is then used to guide the subsequent recovery process. Concretely, given parameters $\boldsymbol{x}_f$ that have resulted in a failure as well as parameters $\boldsymbol{x}'_f$ that violate relations of a nominal execution model, a corrective set of parameters $\boldsymbol{x}^*$ is identified by moving $\boldsymbol{x}_f$ in a direction away from $\boldsymbol{x}'_f$, as this makes it more likely that the violation of the relations will be remedied.

[2]We use the implementation of the model provided in the PyTorch Time library for this purpose `https://github.com/VincentSch4rf/torchtime`.

Algorithm 1 Adaptive Compliant Control with Integrated Failure Recovery (ACCIFR). The elements of the ILoSA algorithm are shown in black font; our added components are shown in blue.

procedure KINESTHETICDEMONSTRATION
  while trajectory recording **do**
    $receive(x_t \rightarrow \xi)$
    $\Delta x^d(x_{t-1}) \leftarrow x_t - x_{t-1}$
end procedure
$train(GPs)$
procedure INTERACTIVECORRECTIONS($\xi, \Delta x^d, K_s^d$)
  while not $success$ **do**
    $receive(x)$
    $[\Delta x, \Sigma] \leftarrow \text{GP}_{\Delta x}(x)$
    $K_s \leftarrow \text{GP}_{K_s}(x)$
    if $feedback$ **then**
      $[\Delta x_{inc}, K_{s\_inc}] \leftarrow interpret(feedback, \Delta x, K_s)$
      if $\Sigma \geq \Sigma_{Threshold}$ **then**
        $append(x \rightarrow \xi,\ \Delta x + \Delta x_{inc} \rightarrow \Delta x^d,\ K_s + K_{s\_inc} \rightarrow K_s^d)$
      else
        $correct(\Delta x_{inc} \rightarrow \Delta x^d, K_{s\_inc} \rightarrow K_s^d)$
      $fit(GPs)$
    $\Delta x \leftarrow \text{GP}_{\Delta x}(x)$
    $K_s \leftarrow \text{GP}_{K_s}(x)$
    if $collision(\Delta x_{t-1}, \Delta x_t)$ **then**
      $P_{contact} \leftarrow predictContactSide(I_{wrench})$
      $feedback \leftarrow correction(P_{contact})$
    else
      $f_{stable} \leftarrow -\alpha \nabla \Sigma$
      $[\Delta x, K_s] \leftarrow modulation(\Delta x, K_s, f_{stable}, \Sigma)$
      $send(\Delta x, K_s)$
    $success \leftarrow monitor(\Delta x^g,\ \Delta x_{t-1},\ D_{th},\ F_{th})$
end procedure

For the recovery mechanism we propose in this paper, we use the wrench data classifier described above to ground relations describing collision directions and then use the idea of recovery by moving away from the direction in which a collision is identified. Concretely, after a collision is detected, corrective feedback recovers the end effector from the collision by steering it away from the side of the collision and towards the goal pose. We use manually defined corrective feedback instead of varying the feedback magnitude as in [12] since, unlike in [12], we do not have an underlying model that can evaluate the expected quality of the correction; the feedback magnitude was empirically found for both collision sides. This feedback is then interpreted just as the teleoperated feedback described in Eq. 2.

Overall, ACCIFR, summarized in Algorithm 1, is a modified and extended version of ILoSA [11] that accounts for failure monitoring and recovery, thereby contributing to more reliable press-fit execution.

Fig. 5: Experimental setup for press-fitting

V. EVALUATION

We conducted experiments to evaluate the performance of our proposed ACCIFR approach compared to the baseline ILoSA approach in terms of its ability to generalize to different variations of the press-fit task. We used a Franka Emika robot manipulator with 7 degrees of freedom, a miniature container setup with milk cartons, and a 3D mouse for user correction, as shown in Fig. 5. We trained the system using a single demonstration and user correction; the same trained models were used throughout the evaluation. We conducted experiments in three different scenarios of the press-fit manipulation task, considering variations in (i) the robot's starting position, (ii) the goal position, and (iii) the object grasping position. We performed a total of 20 runs for each scenario with each variation, such that we evaluated the performance using the number of successful runs and the number of collisions encountered during the task.[3] In the trials with the robot, we used a contact-state classifier that takes an input history of 290 wrench measurements, which is about $10s$; however, we also evaluated the prediction performance of the contact-state recognition classifier with varying lengths of wrench data history.

Our evaluation is based on the following assumptions: (i) before press-fit execution, the robot manipulator starts at a predefined pre-place pose, (ii) an estimated goal pose for placing the carton is given to the robot, (iii) only the arm of the mobile manipulator is moved to perform the press-fit task (fixed base), and (iv) milk cartons have soft-body characteristics.[4]

A. Generalizability Test I: Starting Position Variation

In this scenario, we varied the initial starting position of the end effector in five different ways to fit an object at a fixed goal pose with position $(0.80, -0.05, 0.43)$ and quaternion orientation $(0.58, -0.50, 0.48, -0.40)$ with respect to the robot's base frame. The results in Tab. I demonstrate that the ILoSA framework performed exceptionally well in all five cases, namely the end effector successfully reached the goal without collision for each run in each variation. This demonstrates that the baseline ILoSA is capable of dealing with small variations in the starting position, which may suggest that the added recovery by ACCIFR is not needed; however, the benefit of monitoring and recovery becomes clear below.

[3]The success of each trial was evaluated manually by the experimenter.

[4]A video that illustrates the evaluation process can be found at `https://youtu.be/cFChdalPccc`.

TABLE I: Performance of ILoSA for variations in the starting position

S.No.	Variation	Goal reached (out of 20)
1	position: $(0.74, -0.05, 0.43)$ orientation: $(0.58, -0.50, 0.48, -0.40)$	20
2	position: $(0.74, -0.05, \mathbf{0.52})$ orientation: $(0.58, -0.50, 0.48, -0.40)$	20
3	position: $(0.74, \mathbf{0.01}, 0.43)$ orientation: $(0.58, -0.50, 0.48, -0.40)$	20
4	position: $(0.74, \mathbf{-0.14}, 0.43)$ orientation: $(0.58, -0.50, 0.48, -0.40)$	20
5	position: $(\mathbf{0.59}, \mathbf{-0.14}, \mathbf{0.52})$ orientation: $(0.58, -0.50, 0.48, -0.40)$	20

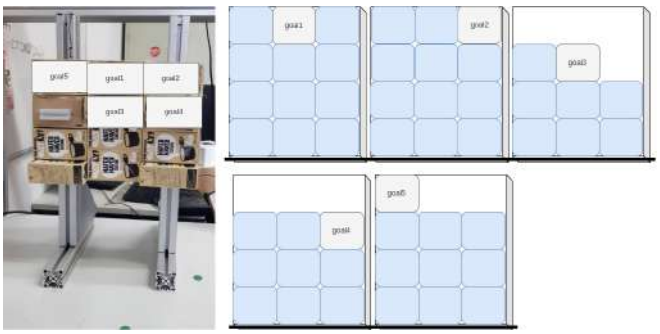


Fig. 6: Validation scenario evaluating the generalizability of ACCIFR across different goal configurations, representing contact situations from various directions.

B. Generalizability Test II: Goal Configuration Variation

In this scenario, we compare the performance of ACCIFR with the baseline ILoSA to perform press-fitting with five different goal configurations, as shown in Fig. 6. For this, we modified the algorithms to predict the next Cartesian pose of the end effector in the *local* frame to reach the goal pose, which means that the end effector has to only move by a certain distance (in the x-direction) to reach the goal.

As shown in Tab. II, the proposed ACCIFR approach outperformed ILoSA regarding successful runs. ILoSA performed well for the default goal pose ($goal1$) but struggled to maintain stability while attempting to reach the other goal poses ($goal2$ and $goal3$), namely it did not reach some goal poses due to collisions. On the other hand, using ACCIFR, the end effector could recover from collisions and reach the goal pose in most cases. Nevertheless, the end effector could not reach $goal4$ in 6 out of 20 runs. We hypothesize that this was caused by the impedance control parameters learned from the user corrections, which was supported by the observation that adding new user corrections improved the performance. The results of this experiment demonstrate that the recovery mechanism effectively enabled the end effector to reach different goal poses, but also that appropriate recording of user corrections is essential for improving the system's performance.

TABLE II: Comparision of ILoSA and ACCIFR for variations in the goal configuration

S.No.	Variation	Goal reached (out of 20)	
		ILoSA	ACCIFR
1	$goal1$	20	20
2	$goal2$	20	20
3	$goal3$	20	20
4	$goal4$	0	14
5	$goal5$	0	20

Fig. 7: Validation scenario evaluating the generalizability of ACCIFR across different object grasping positions, representing pose uncertainty during grasping.

C. Generalizability Test III: Variation in the Object Grasping Position

While performing the press-fit task, the carton might be grasped at a different grasping position than the demonstration with which it is trained, which can also lead to unforeseen collisions with the environment. In this experiment, we compare the performance of ACCIFR with the baseline ILoSA to perform a press-fit task with five different grasping position variations, as shown in Fig. 7.

The experimental results in Tab. III demonstrate that the proposed ACCIFR approach significantly outperforms the baseline ILoSA regarding generalization to different grasping positions. In particular, ILoSA reached the goal pose without collision only when the object was held at the same grasping position as in the demonstration, but it failed in all other variations. In contrast, the ACCIFR approach was successful in all five grasping position variations and recovered from collisions in all cases due to the failure recovery mechanism. The only failure for ACCIFR occurred in the case of $grasp1$, where the grasped object slipped from the gripper in one trial, thus preventing the end effector from reaching the goal pose. These results demonstrate the enhanced adaptability, stability, and collision recovery capabilities of the proposed ACCIFR approach for the press-fit task with variations in the grasping position.

D. Contact-State Classifier Analysis

To examine the effect of the length of the wrench data history on the contact-state classifier and determine the optimal length of wrench data, we also trained multiple versions of the classifier with different lengths of wrench

TABLE III: Comparision of ILoSA and ACCIFR for variations in the object grasping position

S.No.	Variation	Goal reached (out of 20)	
		ILoSA	ACCIFR
1	$grasp1$	0	19
2	$grasp2$	0	20
3	$grasp3$	0	20
4	$grasp4$	20	20
5	$grasp5$	0	20

TABLE IV: Influence of the length of wrench data history on the contact-state classification accuracy

S. No.	Time (in sec)	Average length of wrench data history	Classification accuracy (in %)
1	10	290	100
2	5	147	100
3	2	59	100
4	1	29	100
5	0.5	15	100
6	0.2	6	100
7	0.1	3	100
8	0.05	1	100

data history. Based on the results shown in Tab. IV, it can be seen that the contact-state classifier used in ACCIFR can accurately predict the side of a collision regardless of the length of the wrench data history used. In particular, the classification accuracy remains at 100% for all time windows tested, indicating that the classifier can correctly predict the contact side even when using wrench data with a short history length. This could be attributed to the classifier's ability to extract highly discriminative features from the wrench data, facilitated by the *InceptionTime* model; this allows for the prediction of the contact side to be generated quickly, which enables the robot to swiftly recover from failures, thereby minimizing the impact of collisions. The high accuracy can also be explained by the structure of the experimental setup, where collision profiles remain static between trials; however, as this is a fairly accurate model of a real container loading scenario, the results suggest that the contact-state classifier can also be useful for the real-world press-fitting task.

VI. DISCUSSION AND CONCLUSIONS

The proposed approach, which we refer to as Adaptive Compliant Control with Integrated Failure Recovery (ACCIFR), enables a mobile manipulator to perform a press-fit task. The approach learns the task using a single demonstration and user corrections through the ILoSA framework, while the failure recovery mechanism enables the end effector to avoid getting stuck after colliding with objects, ultimately steering it towards the goal location. Regardless of the initial expert demonstration, ACCIFR's ability to generalize to different variations can be attributed to both ILoSA and the integrated failure recovery mechanism. Moreover, the experimental evaluation indicated that ACCIFR outperformed ILoSA in terms of generalization performance. Concretely, ACCIFR achieved a success rate of 90% for

variations in the goal configuration, whereas ILoSA achieved only 60%. This is also visible in the case of variations in the object grasping position, where ACCIFR achieved an almost perfect success rate, while ILoSA only achieved a success rate of 20%. Our supervised learning-based contact-state classifier exhibited good performance across varying lengths of time-series data, ranging from an average length of 290 to 1, indicating its ability to capture the necessary information for predicting the contact side.

The work presented in this paper focused solely on automating the press-fit task and thus assumes that the mobile manipulator will always start from a pre-place pose, that it knows the estimated goal pose, and that it only uses the arm during the press-fit execution. These assumptions were made due to the limited scope of our study, as the press-fit task is just one aspect of a larger research project.[5] Additional components for automating the shipping container loading process with a custom mobile manipulator are under current development.

Our evaluation of the proposed approach revealed some limitations as well. The controller showed erratic behavior during some trials, which may affect the robot's reliability in performing a press-fit task; future work should thus focus on improving and optimizing the controller for enhanced stability and accuracy. We also observed a hardware limitation in the system, where the object slipped from the end effector during the press-fit task. Using a gripper that can reliably hold the object in place (potentially custom-designed) is essential, especially when performing in-contact manipulation with drink cartons. Another limitation is that our failure recovery mechanism can only recover from collisions on the left or right side; future work should generalize failure recovery behaviors beyond left and right collisions, for instance by incorporating self-exploration and active learning, which could enable the robot to learn from its experiences and actively seek new knowledge to improve its performance. Finally, future work could compare the performance of our time-series classifier with classical learning approaches to assess the practical need for a deep learning-based approach.

ACKNOWLEDGMENT

We would like to thank Vincent Scharf and Konstantin Zähl for their support in the experiments.

REFERENCES

[1] Y.-G. Ahn, T. Kim, B.-R. Kim, and M.-K. Lee, "A Study on the Development Priority of Smart Shipping Items - Focusing on the Expert Survey," *Sustainability*, vol. 14, no. 11, 2022.

[2] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual Reinforcement Learning for Robot Control," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 6023–6029.

[3] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, "Variable Compliance Control for Robotic Peg-in-Hole Assembly: A Deep-Reinforcement-Learning Approach," *Applied Sciences*, vol. 10, no. 19, 2020.

[5]The project MASON aims to automate the shipping container loading process using a custom mobile manipulator.

[4] L. Johannsmeier, M. Gerchow, and S. Haddadin, "A Framework for Robot Manipulation: Skill Formalism, Meta Learning and Adaptive Control," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 5844–5850.

[5] N. Vuong, H. Pham, and Q.-C. Pham, "Learning Sequences of Manipulation Primitives for Robotic Assembly," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2021, pp. 4086–4092.

[6] L. Hermann, M. Argus, A. Eitel, A. Amiranashvili, W. Burgard, and T. Brox, "Adaptive Curriculum Generation from Demonstrations for Sim-to-Real Visuomotor Control," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020, pp. 6498–6505.

[7] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin, "A Framework for Efficient Robotic Manipulation," *CoRR*, vol. abs/2012.07975, 2020.

[8] E. Johns, "Coarse-to-Fine Imitation Learning: Robot Manipulation from a Single Demonstration," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2021, pp. 4613–4619.

[9] C. Yan, J. Wu, and Q. Zhu, "Learning-based Contact Status Recognition for Peg-in-Hole Assembly," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2021, pp. 6003–6009.

[10] I. F. Jasim and P. W. Plapper, "Contact-state monitoring of force-guided robotic assembly tasks using expectation maximization-based Gaussian mixtures models," *Int. Journal of Advanced Manufacturing Technology*, vol. 73, no. 5, pp. 623–633, Jul 2014.

[11] G. Franzese, A. Mészà'ros, L. Peternel, and J. Kober, "ILoSA: Interactive Learning of Stiffness and Attractors," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2021, pp. 7778–7785.

[12] A. Mitrevski, P. G. Plöger, and G. Lakemeyer, "A Hybrid Skill Parameterisation Model Combining Symbolic and Subsymbolic Elements for Introspective Robots," *Robotics and Autonomous Systems*, vol. 161, pp. 104 350:1–22, Mar. 2023.

[13] M. Suomalainen, Y. Karayiannidis, and V. Kyrki, "A Survey of Robot Manipulation in Contact," *Robot. Auton. Syst.*, vol. 156, no. C, oct 2022.

[14] L. B. Chernyakhovskaya and D. A. Simakov, "Peg-on-hole: mathematical investigation of motion of a peg and of forces of its interaction with a vertically fixed hole during their alignment with a three-point contact," *Int. Journal of Advanced Manufacturing Technology*, vol. 107, no. 1, pp. 689–704, Mar 2020.

[15] J. Xu, Z. Hou, Z. Liu, and H. Qiao, "Compare Contact Model-based Control and Contact Model-free Learning: A Survey of Robotic Peg-in-hole Assembly Strategies," *CoRR*, vol. abs/1904.05240, 2019.

[16] D. E. Whitney, "Quasi-Static Assembly of Compliantly Supported Rigid Parts," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, no. 1, pp. 65–77, 03 1982.

[17] Z. Jakovljevic, P. B. Petrovic, and J. Hodolic, "Contact states recognition in robotic part mating based on support vector machines," *Int. Journal of Advanced Manufacturing Technology*, vol. 59, no. 1, pp. 377–395, Mar 2012.

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[19] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, T. Nishi, S. Kikuchi, T. Matsubara, and K. Harada, "Learning Force Control for Contact-Rich Manipulation Tasks With Rigid Position-Controlled Robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5709–5716, 2020.

[20] Y. Deng, Z. Hou, W. Yang, and J. Xu, "Sample-efficiency, stability and generalization analysis for deep reinforcement learning on robotic peg-in-hole assembly," in *Intelligent Robotics and Applications*, X.-J. Liu, Z. Nie, J. Yu, F. Xie, and R. Song, Eds. Cham: Springer International Publishing, 2021, pp. 393–403.

[21] B. Abbatematteo, E. Rosen, S. Tellex, and G. Konidaris, "Bootstrapping Motor Skill Learning with Motion Planning," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2021, pp. 4926–4933.

[22] G. V. de la Cruz, Y. Du, and M. E. Taylor, "Pre-training with non-expert human demonstration for deep reinforcement learning," *The Knowledge Engineering Review*, vol. 34, p. e10, 2019.

[23] I. Fawaz *et al.*, "InceptionTime: Finding AlexNet for Time Series Classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.

Graph-based LiDAR-Inertial SLAM Enhanced by Loosely-Coupled Visual Odometry

Vsevolod Hulchuk Jan Bayer Jan Faigl

***Abstract*— In this paper, we address robot localization using Simultaneous Localization and Mapping (SLAM) with Light Detection and Ranging (LiDAR) perception enhanced by visual odometry in scenarios where laser scan matching can be ambiguous because of a lack of sufficient features in the scan. We propose a Graph-based SLAM approach that benefits from fusing data from multiple types of sensors to overcome the disadvantages of using only LiDAR data for localization. The proposed method uses a failure detection model based on the quality of the LiDAR scan matching and inertial measurement unit data. The failure model improves LiDAR-based localization by an additional localization source, including low-cost black-box visual odometers like the Intel RealSense T265. The proposed method is compared to the state-of-the-art localization system LIO-SAM in cluttered and open urban areas. Based on the performed experimental deployments, the proposed failure detection model with black-box visual odometry sensor yields improved localization performance measured by the absolute trajectory and relative pose error indicators.**

I. Introduction

The localization is important for many mobile robotics applications, including underground exploration, indoor inspection, and outdoor navigation. In these scenarios, the robot's sensors-based localization is needed if external localization systems, such as satellite navigation, are unavailable or do not work reliably because of signal reflections from tall structures. The widely adopted method for localizing a robot using its sensors is *Simultaneous Localization and Mapping* (SLAM) [1], which becomes the de-facto standard in applications where a prior map of the environments cannot be utilized. SLAM can be based on data from various sensors, including *Light Detection and Ranging* (LiDAR) laser scanners [2], visual cameras [3], *Inertial Measurement Units* (IMU), or wheeled odometry, to name just a few.

Using exteroceptive sensors to build a map of the operational environment within which the robot is localized allows for decreasing the localization drift compared to purely proprioceptive incremental methods such as odometry and dead reckoning. Even matching only consecutive frames using *Visual Odometry* (VO) [4] helps to overcome drifts of IMU measurements or slippage of wheeled odometry. Nevertheless, the map's quality is important and related to the data quality, specifically the depth estimates of the range measurements. Current LiDAR sensors provide relatively precise range measurements and can have resolution over one hundred lines [5]. These properties make them suitable for localization, especially in cluttered environments, where LiDAR scans can be precisely matched with respect to (w.r.t.) each other [6]. However, the scan matching may be ambiguous in long corridors or flat fields, leading to localization failure or high drift, as depicted in Fig. 1.

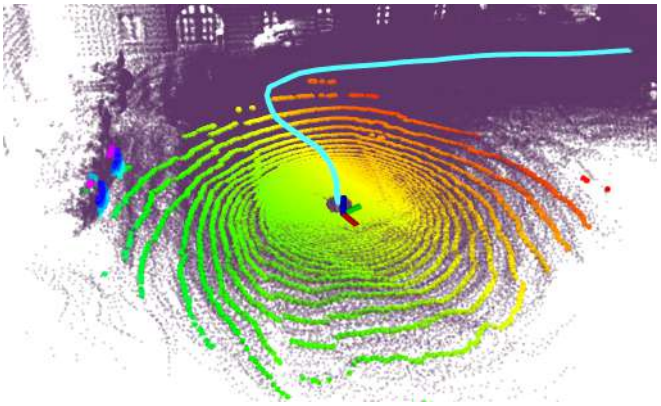


Fig. 1. A situation where LiDAR scan can be aligned with the previous scans (map) in multiple ways since the area covered by the scan is mostly flat, which prompts scan-matching ambiguity. A dense map of the environment is in purple. Distance data of the current scan are denoted in blue to red.

Incremental localization methods, such as IMU and odometry-based methods, including VO, might help to overcome areas where LiDAR scan matching is ambiguous locally, albeit it can lead to higher drift than the LiDAR-based SLAM in the long run. Thus, combining data sources can be advantageous in SLAM, and two main sensor fusion approaches can be found in the literature. The first is tightly-coupled methods that account for sensor raw data, such as in *LiDAR Inertial Odometry via Smoothing and Mapping* [7] (LIO-SAM), where an IMU displacement measurement serves as an initial guess for the scan-matching.

The second class of methods uses a loosely-coupled approach to fuse multiple localization sources, meaning that two displacement outputs from localization systems are fused at the top. Consequently, the resulting estimation tends to be more robust as a failure of one source does not provoke the failure of another one. Also, loose coupling allows the integration of several independent localization systems, making the whole system modular and easily replaceable compared to tightly-coupled systems.

In this paper, we propose an extension of the Pose-

Authors are with the Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Czechia.{hulchvse|bayerja1|faiglj}@fel.cvut.cz

The presented work has been supported by the Czech Science Foundation (GAČR) under the research project No. 22-05762S. The support of the Grant Agency of the CTU in Prague under grant No. SGS22/168/OHK3/3T/13 is also gratefully acknowledged.

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

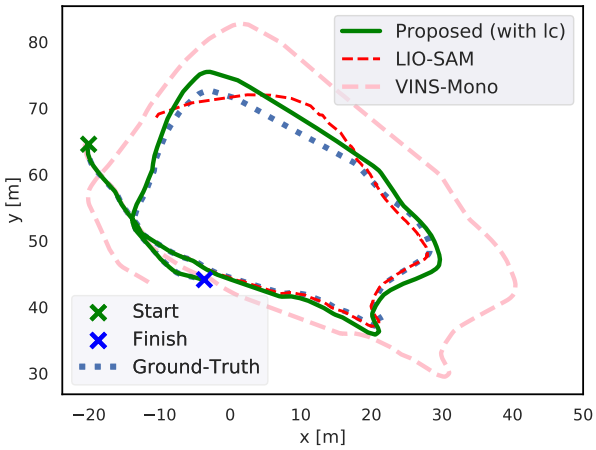


Fig. 2. Result of the proposed Graph-based LiDAR-Inertial SLAM with loosely coupled visual odometry on the rural dataset. Notice that even though the matching of the LiDAR scans was unsuccessful in some areas, the proposed method can use scale and pose drifting visual localization VINS-Mono to overcome such areas and close the loop.

Graph SLAM, combining tightly and loosely coupled ideas. We propose to use tightly coupled sensory fusion between LiDAR and IMU, similar to LIO-SAM. Besides, the developed solution allows utilizing additional sources of pose estimates in a loosely-coupled manner, improving the SLAM performance when LiDAR data matching fails. Various methods of incremental localization can be loose-coupled in the proposed method, such as visual localization, wheel odometry, RADAR-based localization [8], or thermal-inertial odometry [9]. Nevertheless, the properties of the proposed method are demonstrated while using a black box embedded stereo visual localization system, the Intel RealSense T265 (T265) [10], and visual-inertial localization VINS-Mono [11].

We propose a relatively straightforward failure detection model that triggers the incorporation of the additional low-quality pose estimate into the developed Pose-Graph SLAM. The model assesses LiDAR scan matching quality to indicate possible matching failure and IMU-based pose change prediction to confirm the failure for switching the pose estimate. Incorporating the additional localization source is enhanced by an auto-scaling mechanism and improved graph structure.

The triggering threshold has been experimentally established using a real robotic system; the proposed Graph-based SLAM has been deployed in several deployments and compared with the selected state-of-the-art LiDAR-based SLAM. Based on the experimental results, the proposed method demonstrates improvement of the localization performance by the additional source of the incremental localization while not sacrificing LiDAR-based performance in scenarios where LiDAR scan matching performs well, see Fig. 2. We consider the main contributions of the proposed approach as follows.

- Modular enhancement of existing Pose-Graph SLAM by a loosely coupled additional localization system.
- Two-step failure detection model, allowing detection of scan matching failure.

The rest of the paper is organized as follows. Section II overviews the related literature, including a brief description of the selected reference LIO-SAM framework. The proposed method is described in Section III. Experimental results are reported and discussed in Section IV. Finally, the paper is concluded in Section V.

II. Related Work

Many SLAM systems have been proposed [2], [3] and evaluated in the Kitti benchmark [12]. Based on the results reported in [12], most of the top ten performing methods use LiDAR measurements for robot pose estimation. One of the top-performing LiDAR-based methods is LOAM [13], albeit it lacks an explicit loop closure and is limited to only one type of sensor. On the other hand, multiple possible sensors are used in the RTAB-Map [14], which is a general tightly-coupled LiDAR-Visual SLAM framework using multiple graph frameworks. However, failure handling is not resolved in the framework yet, and the authors mention it as a future research direction.

Contrary to the RTAB-Map, the authors of [15] loosely coupled several localization sources. The first step of the coupling is the sanity check, where localization failures are identified for each localization source using the dynamic model of the vehicle. Then, Chamfer distance-based [16] score is used to select the best pose estimate. The advantage of [15] is high robustness, but since the localization sources are completely independent, the visual odometry cannot help the LiDAR-based SLAM to close the loop in the case of temporal LiDAR-based SLAM failure. Furthermore, the Chamfer distance-based score measures the alignment of the LiDAR scans. It does not directly detect when the perfect alignment of LiDAR scans may correspond to a wrong displacement in monotonous corridors or fields.

In [17], the authors review available sensory fusion approaches for LiDAR-Visual SLAM. They mention that the graph-based SLAM [1] is often used for sensor fusion because it abstracts from raw measurements. The approach represents measurements, poses, and observations in a graph structure. Pose-graph SLAM [18] is a specific kind of graph-based SLAM that is the most used nowadays. It restricts the graph's nodes to be poses and positions of robots and landmarks and edges to be measurements-based constraints between them. The authors of [19] demonstrate the computational advantages of the pose-Graph SLAM for large-scale maps, comparing the solution with conventional filtering approaches. The approach is further explored in [20], where the authors review iSAM2 [21], which iteratively re-optimizes only nodes influenced by new observations. Multiple graph optimization frameworks have been proposed, but ORB-SLAM3 [22] uses the g2o library [23] in Loop Closure for Bundle Adjustment [24] to improve the Visual-Inertial Odometry. In VINS-Mono [11], the authors present a Visual-Inertial SLAM solution that fuses a monocular camera and IMU in a tightly-coupled manner for obtaining odometry and optimizing the global trajectory with pose-graph SLAM.

LiDAR-Inertial odometry is the core of LIO-SAM [7] that uses scan matching based on LOAM [13], where the initial guess of the LiDAR pose is based on integrated

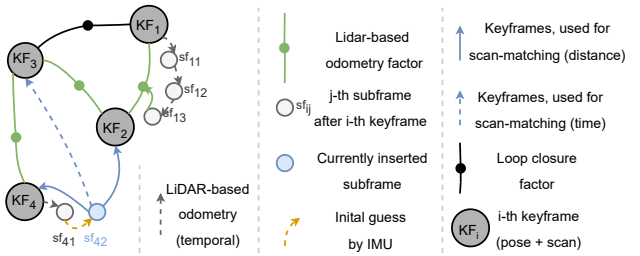


Fig. 3. Map optimization graph in LIO-SAM [7].

IMU measurements. The scans aligned by LiDAR odometry are marked as keyframes if the distance from a pose corresponding to the previous keyframe is above a certain threshold. Otherwise, the pose is treated as a temporal subframe. The relations between the keyframes are represented by constraints that are used to construct a sparse graph within the GTSAM [25] optimization framework. Loop closure is then performed as a parallel process using the *Iterative Closest Point* (ICP) [26], and the loop constraints are added if the ICP converges. For the loop closure detection, the latest keyframe is attempted to be matched against the nearby keyframes, including recent keyframes and keyframes that are close to the current robot pose. If the matching of the keyframes is successful, the transformation between them is inserted into the graph as a constraining factor. The graph structure is illustrated in Fig. 3.

LIO-SAM is further extended by tightly-coupled VO in *LiDAR-Visual-Inertial Odometry via Smoothing and Mapping* (LVI-SAM) [27]. LVI-SAM tightly couples LIO-SAM with Visual SLAM VINS-mono [11] to improve performance in challenging scenarios using sensor-specific failure detectors for LiDAR and VO. However, such an approach does not support flexibility in changing the source of additional localization systems and restricts end-users to specific additional sensors (camera) and algorithms (VINS-mono).

Based on the literature review, we opt for LIO-SAM as a suitable base system for integrating the additional sensor for localization. It provides the advantage of a great performance of LiDAR-based methods [12] while avoiding the disadvantage of the tightly-coupled visual odometry of LVI-SAM, which supports only the specific method of visual odometry. LIO-SAM framework accounts for ambiguities of the scan-matching by checking scan-matching convergence. The convergency is then reflected in uncertainties while optimizing IMU measurements. On the other hand, the system is developed for structure-rich environments. Besides, it does not explicitly handle situations where the scan-matching results are completely unusable. Both the drawbacks are addressed by the proposed loosely-coupled combination of LiDAR-Inertial SLAM and VO.

III. Proposed Method

The proposed loosely coupled VO with the graph-based LiDAR-Inertial SLAM leverages LIO-SAM [7]. It uses the same way of calculating LiDAR-Inertial odometry (referred to as LiDAR-based odometry). However, we modify the factor graph construction to incorporate measurements from an additional localization system, such as VO. The inputs to the proposed method are LiDAR scans, IMU measurements, and pose estimates of the additional localization system(s). Although the proposed approach is general, we consider *Visual-Inertial Odometry* (VIO) as the additional localization system that produces a 6 DoF robot pose estimate to present the proposed concept. The following assumptions are made in the design of the proposed method.

- For simplicity of the description, only a single additional localization system VIO is used, albeit multiple localization sources can be straightforwardly utilized.
- The additional localization system provides pose estimates w.r.t. to the same coordinate frame as the LiDAR-based odometry.
- All sensors' data is synchronized in time.

The proposed method consists of two parts: (i) *failure detection*, which indicates that LiDAR-based odometry failed, and (ii) *visual localization integration*, which integrates the additional localization into the factor graph in the case of the detected failure.

A. Failure Detection

Failure detection starts with the failure indication defined by the *Failure indicator* I_{fail}. If the indication is positive, *Failure resolution* determines if VIO provides a more suitable pose estimate than the LiDAR-based odometry. The overview of the failure detection process is depicted in Fig. 4, and it works as follows.

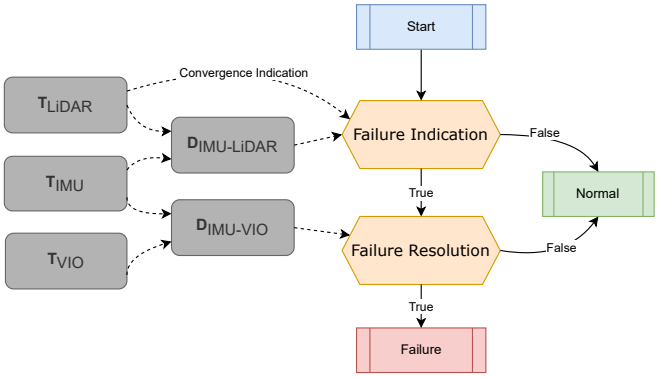


Fig. 4. Failure detection algorithm.

The failure indicator I_{Fail} is combined from two components: *convergence indicator* I_{Conv} and *IMU-based indicator* I_{IMU} as

$$I_{\text{Fail}} = I_{\text{IMU}} \texttt{ or } I_{\text{Conv}}. \tag{1}$$

I_{Conv} is triggered when the LiDAR scan matching does not converge, but it might not cover all cases when it is suitable to switch to VIO. Therefore, we also use I_{IMU} to increase the failure detection rate, which is supported by the experimental results reported in Section IV-A. The advantage of I_{IMU} is that it is not directly influenced by a lack of spatial and visual features in the environment. The indicator uses a rough estimation of the robot motion by IMU-based odometry increment $T_{\text{IMU}} \in SE(3)$ to estimate the adequacy

of the LiDAR-based odometry increment $T_{\text{LiDAR}} \in SE(3)$. Using LiDAR scans at 10 Hz to improve the estimate of the robot motion ensures that the IMU-based motion estimates do not suffer from localization drift by integrating IMU measurements for an extended period. The difference of the increments $D_{\text{IMU-LiDAR}}$ is computed as

$$D_{\text{IMU-LiDAR}} = T_{\text{IMU}} \cdot T_{\text{LiDAR}}^{-1}. \tag{2}$$

We analyze the norm of the rotational component and a translational component of the difference defined by

$$\begin{aligned} r_{\text{IMU-LiDAR}} &= ||\operatorname{rot}(D_{\text{IMU-LiDAR}})||^{\text{ANG}} \\ t_{\text{IMU-LiDAR}} &= ||\operatorname{trans}(D_{\text{IMU-LiDAR}})|| \end{aligned} \tag{3}$$

where $\operatorname{rot}(D_{\text{IMU-LiDAR}}) \in SO(3)$ is the rotational component and $\operatorname{trans}(D_{\text{IMU-LiDAR}}) \in R^3$ is the translational component of $D_{\text{IMU-LiDAR}}$. The term $||\cdot||^{\text{ANG}}$ denotes the angular metric of the rotation that is determined as a rotation angle of the angle-axis representation of the rotation.

The IMU indicator I_{IMU} works as an outlier detector [28], and it is defined as logical `or` of two threshold values

$$I_{\text{IMU}} = (r_{\text{IMU-LiDAR}} > c_r) \texttt{ or } (t_{\text{IMU-LiDAR}} > c_t) \tag{4}$$

that triggers when either the rotational or translational component of the difference $D_{\text{IMU-LiDAR}}$ is larger than the corresponding thresholds c_r and c_t, respectively. The thresholds are determined experimentally using outlier detection methodology; see the following section.

The failure resolution begins if the failure indicator I_{Fail} (1) is true. The VIO pose estimate is used if it is significantly closer to the IMU-based odometry than the LiDAR-based odometry. Thus, the resolution is defined by the following condition

$$\begin{gathered} (r_{\text{IMU-VIO}} < \alpha \cdot r_{\text{IMU-LiDAR}}) \\ \texttt{and} \\ (t_{\text{IMU-VIO}} < \alpha \cdot t_{\text{IMU-LiDAR}}) \end{gathered} \tag{5}$$

where $r_{\text{IMU-VIO}}$ and $t_{\text{IMU-VIO}}$ are defined similarly to the IMU-LiDAR difference $D_{\text{IMU-LiDAR}}$ defined in (3).

Note that the LiDAR-based odometry failure might be indicated based on I_{Fail}, but failure resolution (5) would not activate the usage of VIO pose estimate if the latter does not improve the LiDAR-based one. We incorporate an empirically obtained $\alpha = 0.8$ factor in the IMU-LiDAR difference when considering additional odometry over LiDAR. It is done to ensure the significance of any potential improvement by additional odometry and account for IMU noise.

B. Visual Odometry Integration – Scale Self-Adjustment

Let us suppose the LiDAR-based odometry failure is indicated, and VIO provides more precise localization according to the rule (5). In that case, the VIO is incorporated into the factor graph in place of the LiDAR-based odometry, introducing a constraint between the keyframes if the keyframe is inserted. Since the additional odometry (such as visual or wheeled) might suffer from a wrong scale or slow scale drift, the proposed method performs dynamic scale self-adjustment, estimating the scale of the odometry when the LiDAR-based localization is considered sufficiently precise.

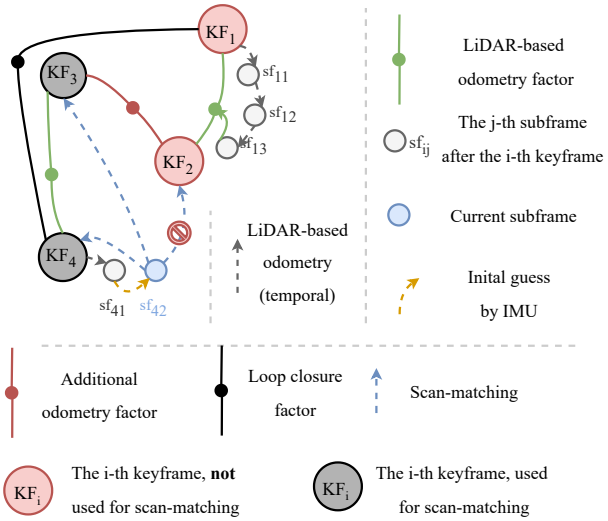


Fig. 5. The proposed method for combining the LiDAR-based odometry with VIO-based pose estimate increments.

We propose to utilize the median value of the moving window to compute the scale. In particular, 500 keyframes-long window includes the past ratios of the absolute values of the translations $t_{\text{VIO}}/t_{\text{LiDAR}}$, where $t_{\langle\text{source}\rangle}$ is the norm of the translational part of the odometry increment. Then, the factor graph structure is created according to the scheme depicted in Fig. 5 as follows.

- The LiDAR-based odometry creates constraints between the previous and the new keyframes based on scan-matching when the LiDAR-based odometry works successfully. When the new LiDAR scan (frame) is available, it is scan-matched against a reference map combined with the nearby keyframes to create such a constraint. Similarly to LIO-SAM, only if the estimated pose increment exceeds a configurable threshold the frame is inserted into the map as a keyframe. Otherwise, it is treated as a temporal sub-frame to improve the initial guess of the next frame pose and output localization information.
- On the other hand, the VIO constraint is inserted instead of the LiDAR-based one if the failure is detected. However, in contrast to LiDAR-based constraints, the VIO-based ones are not guaranteed to be optimized for the keyframes alignment as they optimize visual features alignment and may suffer from the incorrect and drifting scale. Thus, combining keyframes connected with VIO-based constraints can result in a poorly aligned reference map, and a new LiDAR scan would not be successfully matched against such a reference map. Therefore, only keyframes inserted after the last VIO usage are combined in the reference map when the new LiDAR scan is processed.

Finally, it is necessary to properly handle Loop closure constraints of the graph-based SLAM that aim to match keyframes that are far from each other. These constraints may fix the drift introduced by the VIO constraints. However, false loop closures may appear for the structure-

less keyframes, consequently breaking the graph. Therefore, keyframes corresponding to the LiDAR-based odometry failure are deemed unsuitable for loop closures. Further, the inserted VIO constraints are set to have ten times larger uncertainty than the LiDAR-based ones to ensure that loop closure constraints will fix only VIO constraints without affecting LiDAR-based constraints significantly. The effect of the proposed loop closing system has been experimentally examined, and results are reported in the following section; in particular, the effect is demonstrated in Fig. 10.

IV. Experimental Results

The proposed method has been experimentally validated using a four-wheeled skid-steered robot Husky. The robot was equipped with the Ouster OS0 LiDAR with 128 lines, and the maximum range is approximately $50\,\mathrm{m}$, a 9-axis IMU Xsens MTi-30, and a fisheye stereo tracking camera, the *Intel RealSense T265* (T265). T265 provides out-of-the-box VIO odometry, but its internal loop closures have been disabled to make it compliant with made assumptions on the additional localization systems. The careful extrinsic calibration by measuring the relative pose of T265 w.r.t. the LiDAR was done to comply with the proposed method's assumptions. Thus T265 pose estimations were transformed into the LiDAR frame before using them by the proposed method. The 3 DoF ground truth localization of the robot has been recorded using the Leica TS16 total station, shown in Fig. 6a.

(a) Total station setup (b) Bird's-eye view on urban campus area

Fig. 6. The urban experimental scenario at the Czech Technical University in Prague campus at Charles Square.

The LIO-SAM itself was already evaluated using publicly available datasets in [7]. In this work, we primarily focus on areas that induce the scan matching ambiguity. Thus, two environments have been considered for system performance evaluation. The first environment is the backyard area of the Czech Technical University (CTU) in Prague campus at Charles Square, depicted in Fig. 6b. The second environment is a parking lot at Prague's outskirt visualized in Fig. 7. While the first environment can be considered structure-rich, the parking lot in the rural area contains wide-open locations where LiDAR scans do not provide sufficient features for successful scan matching. The testing environments are denoted as *campus* and *rural* scenarios.

The length of the traveled trajectory is $285\,\mathrm{m}$ and $300\,\mathrm{m}$ for the campus and rural scenarios, respectively. The proposed method is examined with different failure indicators to

(a) Used wheeled robot (b) Bird's-eye view on a parking lot

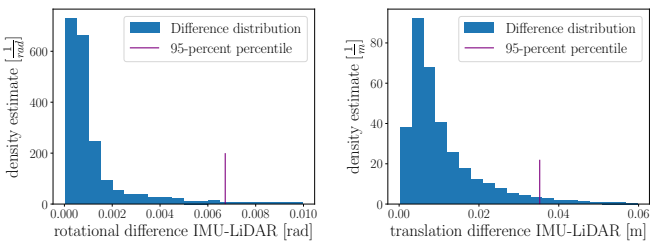
Fig. 7. Experimental parking lot scenario in Prague's outskirts.

justify the combined indicator denoted IMU + Convergence. Besides, the performance is compared with the LIO-SAM [7] as the former localization method to show the benefits of the proposed loosely-coupled VIO.

The evaluation is based on the methodology [29] using medians of the *absolute trajectory error* ATE_t and *relative pose error* RPE_t indicators considering the translational parts of the localization error. In particular, ATE_t evaluates the global accuracy of the trajectory, while the median RPE_t estimates the local consistency of the localization (drift). For RPE_t, the step Δ is set to $1\,\mathrm{m}$, which corresponds to the minimum distance between consecutive poses. Besides, the standard deviation STD_t of the RPE_t is reported to account for outliers. The indication `Fail` is used in cases when the system received corrupted odometry, which led to wrong IMU bias estimation. Such situations prevented the localization system from recovering.

A. Parameterization of the Failure Detection

The failure detection model's parameters have to be estimated, and the following intent describes the estimation. Note that these results are only used to calibrate the proposed method but do not serve to estimate the performance of the proposed method. The proposed IMU-based failure detection model is based on outlier detection [28] for differences between IMU-based and LiDAR-based pose increment $D_{\text{IMU-LiDAR}}$ as of (4) with two established threshold values c_r and c_t, which single out the outliers (failures). The threshold values are set as follows.



(a) Rotational part (b) Translational part

Fig. 8. Histograms of $D_{\text{IMU-LiDAR}}$ differences in the campus scenario. The threshold values c_r and c_t are established as 95 percent quantiles depicted by the vertical line segment.

We model the baseline distributions of differences $r_{\text{IMU-LIDAR}}$ and $t_{\text{IMU-LIDAR}}$ in the non-failure scenario and set the outliers thresholds as 95 percent quantiles of the

distributions as shown in Fig. 8. LiDAR-based odometry provides satisfactory results that can be treated as "Non-Failure" in the full-range campus dataset; thus, the data is used to model the distribution. Note that the data used does not intersect with data from the campus dataset in reported evaluation tables, ensuring that the model tuning and evaluation are performed using different data.

B. Performance in the Campus Scenario

The robot has been operated in the campus scenario where the total station provides the ground truth data for evaluation. We examine the localization performance of the proposed method based on the scan-matching failure indicator I_{Conv} only and with both indicators I_{Conv} and I_{IMU}. First, we examine the method using only the scan-matching failure indicator and using both indicators. Limiting the LiDAR range to 10 m has induced the scan-matching ambiguity as illustrated in Fig. 1.

TABLE I

LOCALIZATION PERFORMANCE IN THE CAMPUS SCENARIO WITH AND W/O FAILURE DETECTION AND LIDAR RANGE CROPPED TO 10 m

Method / Failure Indicator	ATE_t [m]	RPE_t [m]	STD_t [m]
LIO-SAM [7] (No indicator)	`Fail`	`Fail`	`Fail`
Proposed IMU	`Fail`	`Fail`	`Fail`
Proposed Convergence	5.35	0.08	0.22
Proposed IMU + Convergence	**4.70**	**0.06**	**0.26**

`Fail` indicates the method has not been able to produce reasonable results.

The results in Table I indicate that a solo IMU-based indicator cannot detect failure by itself but significantly improves the performance when combined with the convergence-based indicator, reflected in more precise localization results.

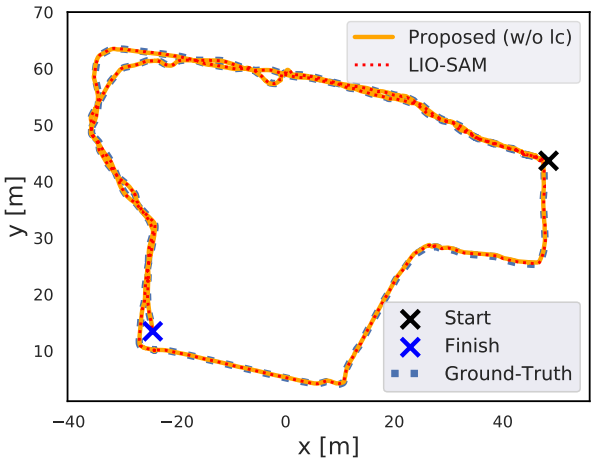
TABLE II

LOCALIZATION PERFORMANCE IN THE CAMPUS SCENARIO WITH FULL LIDAR RANGE AND LIMITED RANGE TO 10 m

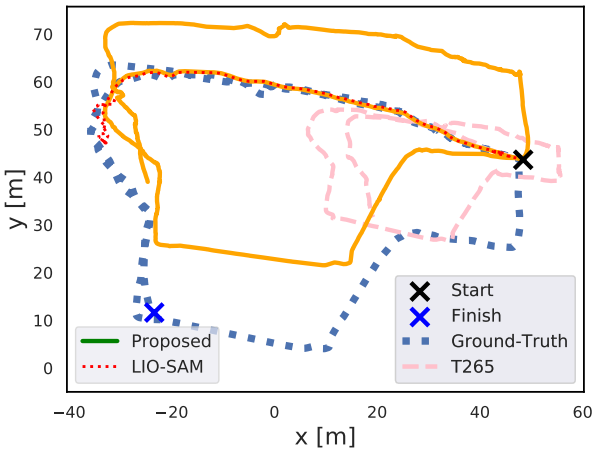
Method	Full range			Limited range		
	ATE_t [m]	RPE_t [m]	STD_t [m]	ATE_t [m]	RPE_t [m]	STD_t [m]
LIO-SAM	0.08	0.04	0.03	`Fail`	`Fail`	`Fail`
T265	16.40	0.83	0.50	16.40	0.83	0.50
T265 scaled*	7.06	0.20	0.30	7.06	0.20	0.30
Proposed method (w/o lc)	0.13	0.04	0.03	4.70	**0.06**	**0.26**
Proposed method (with lc)	0.13	0.04	0.03	**2.8**	0.08	0.3

`Fail` indicates the method has not been able to produce reasonable results.
*Odometry scaled to optimize ATE_t with the constant scale factor after the experiment.

Next, we examine the proposed method and LIO-SAM in two setups: *full range* and *limited range*. Besides, we consider the method in two setups: without and with the *loop closure* (lc). The methods are fed with data directly captured by LiDAR without any range restrictions for the full range. However, for the limited range, LiDAR's range is cropped to 10 m to examine the localization system performance under conditions where LiDAR scan matching might be ambiguous. In addition to LIO-SAM and the proposed method, we evaluate the localization provided by the T265 with its and with the optimal scale. The optimal scale is the scale minimizing the ATE_t for the T265 trajectory, estimated after the experiment and applied to the entire T265 trajectory. It is considered to estimate the best possible reachable result using T265 with the constant scale. Nevertheless, the proposed method is inputted with the raw T265 data, estimating the scale online using the method introduced in Section III-B. The performance indicators are depicted in Table II and trajectories in Fig. 9.



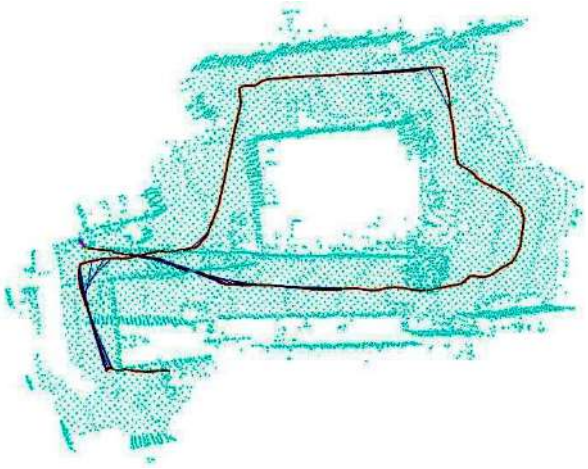
(a) Full range



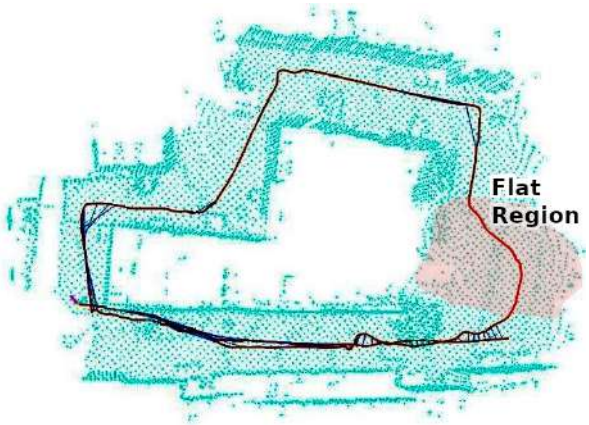
(b) Limited range to 10 m

Fig. 9. Aligned trajectories in the campus dataset.

The presented results support the hypothesis that the environment is structure-rich for the full range and that LIO-SAM and the proposed method provide competitive results. On the other hand, T265 suffers from localization drift and provides worse results, but as rarely used, it only slightly worsens the performance of the proposed method compared to LIO-SAM. However, when the LiDAR range is cropped to 10 m, LIO-SAM fails to output any feasible result once the robot enters the area where it is too far from the buildings. The limited LiDAR scans are ambiguous for the scans-matching algorithm, and the whole localization fails. The proposed method handles these ambiguous LiDAR scans by switching to VIO, as shown in Fig. 9b. Although it introduces



(a) Before loop closure



(b) After loop closure

Fig. 10. Loop closure example conducted by the proposed method in the campus scenario.

a drift caused by the additional odometry, it performs best.

Finally, we examined the loop closure of the proposed method. The obtained maps and trajectories before and after the loop closure are depicted in Fig. 10. It can be observed that the loop closure compensates for the drift introduced by the relatively low-quality VIO. The resulting map is aligned because the loop closure constraint optimized the trajectory where the LiDAR-based odometry was ambiguous, which is the flat region at the right part of the map. At the same time, LiDAR-based constraints that align keyframes with no ambiguity are almost not changed because those have much lower uncertainty in the graph structure. In Table II, it can be seen that for the limited range setup, the loop closure highly improved global consistency reflected by the ATE_t metric while slightly worsening local consistency reflected by the RPE_t metric.

C. Performance in the Rural Scenario

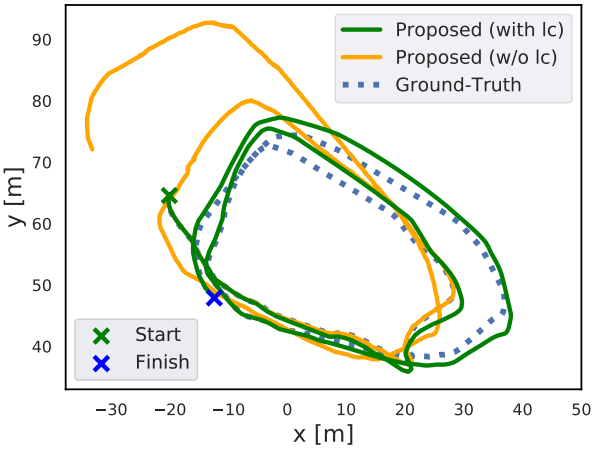
The next deployment took place in the rural scenario with wide open areas. In this case, we use fisheye images from the T265 processed by the VINS-mono [11] odometry to show the flexibility of the proposed method to incorporate measurements from various types of additional localization systems. Thus, we examine the performance of LIO-SAM, VINS-Mono, and two variants of the proposed method, without and with loop closure (lc). The results are summarized in Table III.

TABLE III
LOCALIZATION PERFORMANCE IN THE RURAL SCENARIO

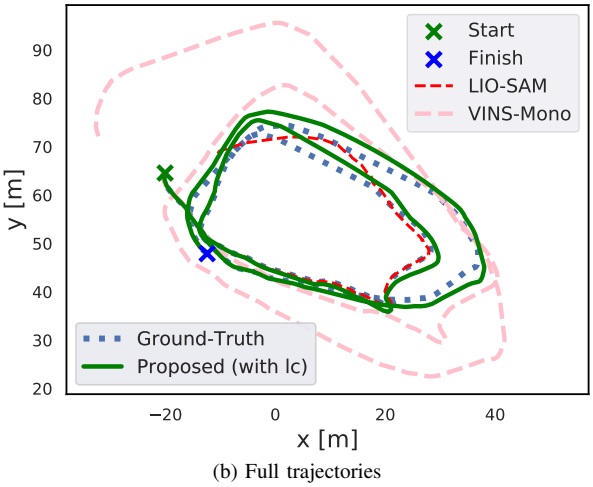
Method	ATE_t [m]	RPE_t [m]	STD_t [m]
LIO-SAM	`Fail`	`Fail`	`Fail`
VINS-Mono	10.9	0.39	0.25
VINS-Mono scaled*	4.97	0.42	0.18
Proposed method (w/o lc)	7.7	0.19	**0.13**
Proposed method (with lc)	**2.4**	**0.15**	1.0

`Fail` indicates the method has not been able to produce reasonable results.
*The odometry scaled to optimize ATE_t with the constant scale factor after the experiment.

From the results, it can be seen that the proposed method performed better than LIO-SAM since it did not fail. VINS-mono provided the robot with smooth but scale and pose drifted odometry. It can be seen in Fig. 11a that due to the loop closure, the proposed method is able to re-estimate the whole trajectory, mainly altering the part where the additional odometry was used. The trajectories the evaluated methods provide are depicted in Fig. 11b.



(a) Loop closure effect



(b) Full trajectories

Fig. 11. Proposed method results in parking dataset.

V. Conclusion

We propose an augmentation of the graph-based SLAM based on LiDAR-Inertial odometry in a modular way for incorporating an additional localization source. Although the additional localization is combined with the LiDAR-Inertial odometry in a loosely-coupled manner, the resulting factor graph can be optimized by identifying loop closures based on LiDAR data even in cases when LiDAR scans matching failed at some part of the trajectory. The proposed improvement is based on failure detection by an IMU model, setting the graph constraints uncertainties according to the nature of localization sources and setting the selection rules for keyframes usage. The proposed method has been tested in urban and rural scenarios demonstrating competitive results compared to LIO-SAM when LiDAR scan matching is not ambiguous. The proposed method outperforms LIO-SAM when the ambiguity of the scan matching induced high localization drift and even a failure of LIO-SAM. The results also indicate that the proposed method can utilize additional localization systems. Moreover, the automatic auto-scale of the data from additional localization supports drifting black-box localization systems like the utilized T265.

For future work, we plan extensive evaluation and comparison of the proposed method with other SLAM methods, including vision-based ones.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intelligent Robotics and Autonomous Agents series. MIT Press, 2005.

[2] L. Huang, "Review on lidar-based slam techniques," in *International Conference on Signal Processing and Machine Learning (CONF-SPML)*, 2021, pp. 163–168.

[3] I. Abaspur Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, "A survey of state-of-the-art on visual slam," *Expert Systems with Applications*, vol. 205, no. 117734, 2022.

[4] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2004.

[5] R. Roriz, J. Cabral, and T. Gomes, "Automotive lidar technology: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6282–6297, 2021.

[6] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP Variants on Real-World Data Sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[7] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5135–5142.

[8] E. Ward and J. Folkesson, "Vehicle localization with low cost radar sensors," in *IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 864–870.

[9] S. Khattak, C. Papachristos, and K. Alexis, "Keyframe-based direct thermal–inertial odometry," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3563–3569.

[10] "Intel RealSense Tracking Camera T265," https://www.intelrealsense.com/tracking-camera-t265/, accessed Apr 9, 2023.

[11] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[12] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[13] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[14] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[15] A. Reinke, X. Chen, and C. Stachniss, "Simple but effective redundant odometry for autonomous vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9631–9637.

[16] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *Proceedings: Image Understanding Workshop*, 1977, pp. 21–27.

[17] C. Debeunne and D. Vivet, "A review of visual-lidar fusion based simultaneous localization and mapping," *Sensors*, vol. 20, no. 7, p. 2068, 2020.

[18] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[19] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2d mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 22–29.

[20] Y. J. Xu X, Zhang L, "A review of multi-sensor fusion slam systems based on 3d lidar," *Remote Sensing*, vol. 14, no. 12, p. 2835, 2022.

[21] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3281–3288.

[22] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[23] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3607–3613.

[24] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *1999 Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms*, 2000, pp. 298–372.

[25] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

[26] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[27] T. Shan, B. Englot, C. Ratti, and D. Rus, "LVI-SAM: tightly-coupled lidar-visual-inertial odometry via smoothing and mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5692–5698.

[28] P. Filzmoser, R. G. Garrett, and C. Reimann, "Multivariate outlier detection in exploration geochemistry," *Computers & geosciences*, vol. 31, no. 5, pp. 579–587, 2005.

[29] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.

Symmetric Object Pose Estimation via Flexible Modular CNN

Simone Mentasti[1], Claudia Speranza, and Matteo Matteucci[1]

***Abstract*— Object pose estimation is a crucial task in various applications, including human-robot interaction, mobile robotics, and augmented reality. It involves determining the position and orientation of an object relative to a reference frame. This is a challenging task due to the need for accurate object detection and recognition, as well as understanding its geometry and the surrounding environment. Depending on the application and available resources, this task can be performed using Lidars, as in autonomous driving, or smaller RGBD cameras, as in mobile robotics. This work proposes an innovative convolutional neural network (CNN) for object pose estimation from RGBD data. The model is designed to have two separate branches, one for estimating the object's position and one for estimating the orientation, to facilitate the training process without loss in performance. Moreover, our approach emphasizes the problem of symmetric object pose estimation, for which we designed a new loss function to better represent the rotation error. The proposed model, with the newly introduced loss function, outperforms state of the art models on public datasets for object pose estimate, both for standard asymmetric objects and symmetric ones.**

I. Introduction

Object pose estimation involves detecting the 6 Degrees of Freedom (6 DoF) pose of an object in three-dimensional space with respect to a reference point of view. This task is crucial in various fields, such as robotics [1], augmented reality [2] [3], and autonomous driving [4]. In robotics, for instance, it is essential in grasping, which requires the coordination of perception, planning, and control of movements.

6 DoF pose estimation problem is commonly tackled by positioning one or more cameras on the robot and by designing a system able to derive the object's pose only from the data captured by these devices. Several methods have been proposed to estimate an object's pose from camera frames in recent years. Classical approaches rely on handcrafted features and template matching [5], but they struggle when occlusions and significant light variation are present. For these reasons, approaches that rely on deep learning algorithms have achieved higher performance than traditional computer-vision-based solutions [6]. Nevertheless, they require specific postprocessing techniques, like customized Iterative Close Point procedure for PoseCNN [7] or the 3D model of the object to compute the loss at training time for DenseFusion [6].

Our work proposes a flexible deep learning-based model that uses depth cameras to detect the object's pose from a single color image enriched by depth information. Our solution estimates the object's translation and rotation independently to facilitate the training process. Moreover, we introduce a novel approach to deal with the multiplicity of equivalent orientations that characterize symmetric objects. The contribution of this paper is thus twofold:

- We propose a new architecture for object pose estimate, designed to have two separate branches, one for the object's pose and one for the orientation. This allows disjoint training of the two tasks, making the process more efficient and straightforward.
- We introduce a new loss for symmetric objects that better measures the error in rotation compared to the state-of-the-art one presented in [6]. At the same time, we define a new metric to evaluate any 6DoF pose estimate model based on the idea behind our custom loss.

This work is structured as follows. In Section II, we present an overview of the current state of the art on object pose estimation. Then in Section III, we detail our newly designed model, and we introduce the custom loss function. In Section IV, we compare the performance of our model against a state of the art architecture on a common dataset and validate our loss function. To perform the analysis, we use both our proposed metrics and the state of the art one to guarantee a fair comparison. Section V briefly concludes the paper.

II. Related works

Object detection and pose estimation can be performed using different sensor modalities. Among these, the most prevalent ones are images and PointClouds. Images may consist of RGB data captured using conventional cameras or RGBD data, which augment visual information with depth. The additional depth channel provides discretized distance information for each pixel in the image. Conversely, PointClouds represents a set of data points in a 3D coordinate system. Each point in the PointCloud is defined by its x, y, and z coordinates, which correspond to its spatial position in the 3D space, and may also include supplementary information such as color, intensity, or reflectivity. PointClouds offer a 3D depiction of the targeted scene.

Both images and PointClouds can be leveraged to perform object detection and pose estimation. When using solely PointCloud data, the most widely applied deep-learning-based approach is PointNet [8].

PointNet [8] is a pioneer in processing point sets directly. It uses neural networks to extract high-level features from PointClouds of different objects to perform classification and

[1] Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, Milan, Italy {name.surname}@polimi.it

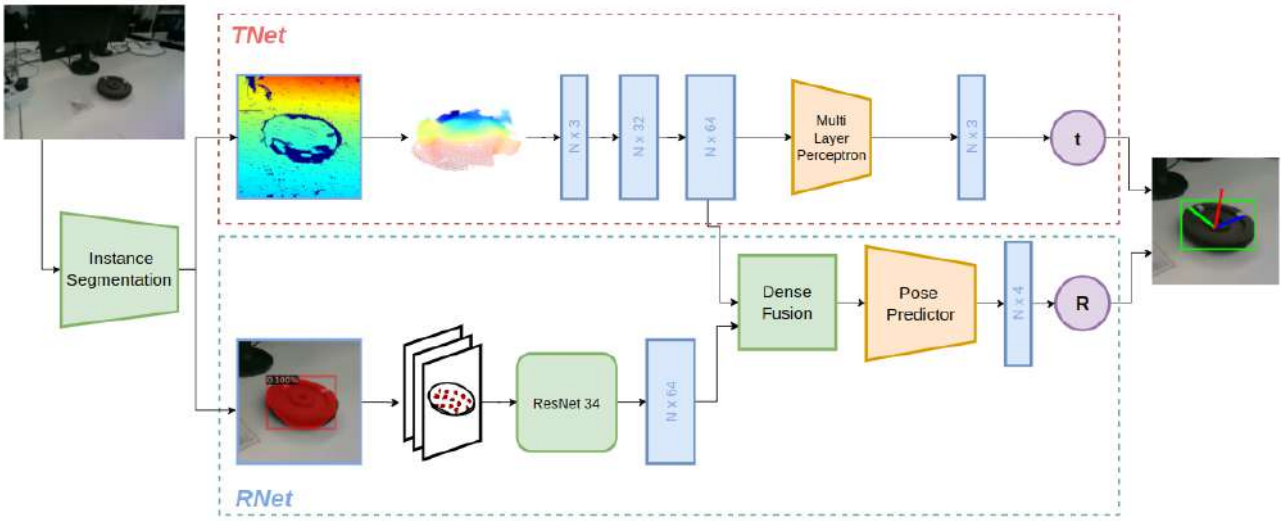


Fig. 1: Schema of the proposed Object 6DoF pose estimation model. The network takes as input RGBD data, detect the object and predicts its position and orientation. The model is structured as two independent branches that can be trained separately.

object part segmentation. The basic idea of PointNet is to learn a spatial encoding of each point and then aggregate all individual point features to a global PointCloud signature. PointNet++ [8] extends the original model introducing a hierarchical neural network that applies PointNet recursively on a nested partitioning of the input points' set. The idea of PointNet++ is first to partition the set of points into overlapping local regions, then, to extract local features capturing geometric structures from small neighborhoods.

Despite their ability to achieve high accuracy, PointNet and its variants demand high-density and accurate PointClouds, which can solely be acquired using Lidar sensors. Due to their high cost and size, these sensors are suitable for implementation in autonomous vehicles, but cannot be effortlessly integrated into smaller and less expensive systems. Hence, an alternative solution is to leverage RGB images or RGBD data, where depth is computed from stereo-vision techniques or infrared projection. While these sensors produce less accurate depth information than Lidar sensors, they greatly reduce sensor size, weight, and cost, making feasible their integration into smaller devices and mobile robots.

The classic RGB-based object pose estimation approach relies on detecting 2D image keypoints, followed by using a PnP algorithm [9] to estimate the 6DoF pose. Recently new voting-based approaches have also been introduced to estimate the object pose, as shown in [10]. With the advancement of deep learning techniques, some neural network-based 2D keypoint detection methods have been introduced. For instance, some methods, such as [11], directly regress the 2D coordinates of the keypoints, while [12] uses heatmaps to locate the 2D keypoints and [13] predicts the 3D coordinates of each object's pixel. One widely used model that exploits this is PoseCNN [7], which achieves remarkable performance in estimating 6DoF pose using only RGB images. The fundamental concept behind PoseCNN is to divide the pose estimation task into different components, enabling the network to model their dependencies explicitly. Another promising model is the Geometry-guided Direct Regression Network (GDR-Net) [14], which estimates the 6D pose end-to-end from dense correspondence based on intermediate geometric representations.

The advent of accessible depth cameras has enabled methods that infer poses of low-textured objects even in poorly lighted environments more accurately than RGB-only methods. To exploit the new depth channel information, researchers had to face the problem of combining heterogeneous data from color images and 3D PointClouds. Xu et al. [15] proposed the PointFusion network that extracts point could features using a variant of PointNet and derives the image appearance features from a CNN. The two vectors of features are then combined in a fusion network to extract 3D bounding boxes. Later, Wang et al. introduced a novel local feature fusion scheme and a fast iterative refinement to improve the pose estimation further with a network called DenseFusion [6].

Our proposed architecture extends the state of the art on RGBD object pose estimation, providing a modular network that can be trained independently for the translation and rotation task. This makes the whole process more efficient and straightforward removing the need for a 3D model to compute the training loss [6]. We also introduce a custom loss that better models the rotation error, compared to the one preciously used in [7].

III. Modular CNN for 6 DoF pose estimation

The goal of an object pose estimate network is to retrieve the 6 degrees of freedom pose of an object in the three-dimensional space with respect to the observer. The 6DoF pose is commonly represented with a homogeneous transfor-

mation matrix, composed by a rotation matrix $R \in SO(3)$, being $SO(3)$ the group of all rotations around the origin of a three-dimensional Euclidian space, and a translation $t \in \mathbb{R}^3_x$. The system's inputs are the frames from an RGBD depth camera (i.e., RGB and depth images).

Figure 1 presents an overview of our network architecture, composed of three distinct and nearly independent blocks. The first block focuses on identifying the object of interest in the color frame for which we aim to estimate the pose. The second block, referred as the center regression branch or TNet, estimates the three-dimensional coordinates of the object center. Finally, the orientation estimation block, or RNet, predicts the object rotation. Since the blocks are independent, it is possible to separate the computation of the object translation from the rotation estimation, thus allowing us to study, train and optimize each sub-task independently. Furthermore, this modular approach enables us to evaluate the performance of the model with respect to each target separately and modify the architecture accordingly. Additionally, the separate networks for translation and rotation prediction offer flexibility to the user to apply only a specific part of the full model if necessary (e.g., when the rotation is known in advance).

Having two separate branches allows us to train the model with a simple loss function (e.g., the norm of the distance between ground truth and prediction). Contrarily, training a complete model often requires ad-hoc solutions and, as shown in the literature [6], a 3D model of the target object. In the next sections, we detail the three core components of our model and the designed loss functions.

A. Instance segmentation

The preprocessing segmentation serves as an initial tool for the network. Its primary objective is to identify the object of interest in the image to be positioned in the 3D space. By performing this first step as part of the model, we can provide a cropped image to the rest of the network, which only contains the object's information and eliminates the environment's interference. For this task, we used an instance segmentation network, which predicts the object's pixels in the image, allowing us to generate the required mask. In particular, we adopted a pretrained version of Mask R-CNN [16], a widely used state of the art model for image segmentation fine-tuned on our object of interest.

B. TNet

The goal of the TNet branch is to predict the object translation t (i.e., the vector representing the location of the center of the object in the three-dimensional space with respect to the camera reference frame). For this task we only exploit the depth image that has been cut out using the mask from the instance segmentation model. We extract a fixed number N of pixels randomly selected from the depth frame, and we use the camera intrinsic parameters to reconstruct the 3D coordinates corresponding to each selected pixel. In this way we obtain a sparse PointCloud restricted to the visible object surface. Therefore, the input of the center regression

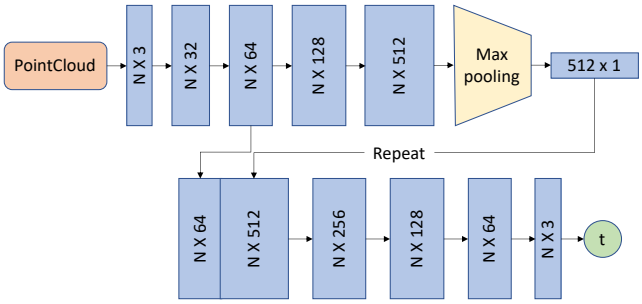


Fig. 2: Architecture of the TNet branch. The 3D position is only computed from PointCloud data

model is a $3 \times N$ matrix. From this, the network performs a feature extraction of local and global geometric structures for each pixel, and then it extracts an estimate of the object center from each selected point.

Exploiting only the reconstructed PointCloud data while ignoring the color information for this task produces a network that deals with smaller input data and less parameters. In this way we obtain a model that is fast to train and requires small computational power to infer the object translation. Having a simpler task to complete with respect to the full pose estimation, the depth information provides sufficient data for the model to learn to predict the object center. Moreover, PointCloud data are not affected by variable light conditions or any other noise that characterizes RGB images.

The detailed structure of TNet is shown in Figure 2. The model receives as input a $3 \times N$ matrix corresponding to the three-dimensional coordinates of N points randomly extracted from the visible object surface. The objective of this model is to estimate the center of the object by predicting for each point in the input its translation with respect to the center. Therefore, we predict a three-dimensional vector Δx_i for each point, obtaining in this way again a $3 \times N$ matrix as output. From this, we can compute the center position predicted by the i^{th} point, by adding the Δx_i to its three-dimensional position. The final translation output can be obtained as the average of all the per-pixel predicted center positions. The network is built using a series of fully connected layers with ReLu activation functions that extract a vector of 1024 features for each point, then a max pooling layer is used as a symmetric function to aggregate information from all points and to generate a vector of global features. Finally, global features are combined with the output of the second layer to derive the final output.

To train the network we define a loss function that penalizes the incorrect predictions. This is done by taking the average of the error on the prediction of each point. Let x_i be the position of the i^{th} point, we define the distance between the center x_C as a vector Δx_i computed as follows:

$$\Delta x_i = x_C - x_i \quad for \quad i = 1, ..., N. \tag{1}$$

$\Delta \hat{x_i}$ represents the prediction of our model for the i^{th} point, while Δx_i is the target vector. The loss function of the

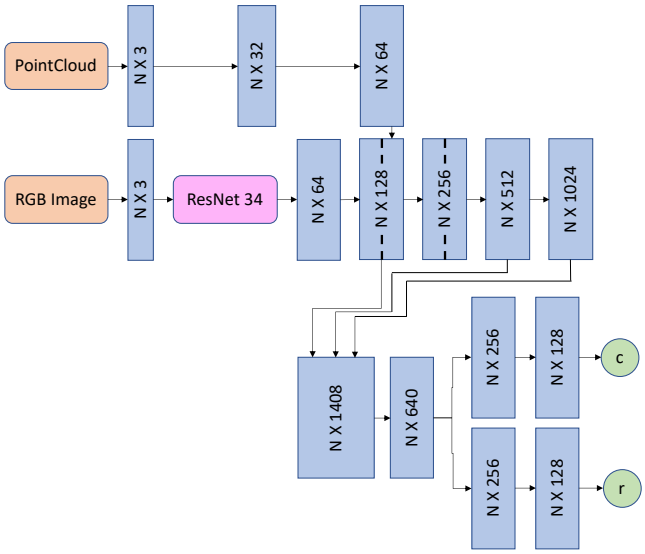


Fig. 3: Architecture of the RNet branch. RGB data and PointCloud are processed by the network to predict the rotation vector and the confidence score.

translation can be defined as:

$$L^T = \frac{1}{N}\sum_{i=1}^{N} \|\Delta x_i - \Delta \hat{x_i}\|_1 . \tag{2}$$

C. RNet

For the rotation task, we designed a more complex network that relies on color and depth images to predict the object's rotation. As for the center regression model, we randomly pick N pixels between the ones selected by the instance segmentation preprocessing step. Then, we extract the three channels data from the RGB frame and the corresponding distance information from the depth frame from each pixel. These are fed to two different sub-networks that compute the corresponding embeddings, which are then fused and processed by a set of fully connected layers that derive from each of the originally picked pixels an estimate of the rotation in the form of quaternions. The model architecture is presented in Figure 3.

We use the same structure as the initial layers of the TNet model to extract features from the depth information. This allows us to reuse the weights learned for the center regression and reduce the number of layers of the model. Instead, the feature extraction for the colored pixels is made with Resnet34 model [17], a widely employed model for image processing.

The extracted color information and PointCloud embeddings are then combined and fed to a fusion module, similar to the DenseFusion network [6]. This part of the network concatenates each pair of features and generates a fixed-size global feature vector using an average pooling layer. In the end, this global feature vector is appended to the local feature vector for each pixel. In this way, we enrich each dense pixel feature with the global fused features to provide a global context. Finally, we feed each resulting per-pixel feature into a final block that predicts the object's pose. The pose predictor comprises four fully connected layers that progressively reduce the initial dimension of the per-pixel feature to a four-dimensional vector representing the quaternions of the estimated object rotation.

The N rotation prediction cannot be combined as shown for the translation prediction by computing the average of the estimated center translation. For this reason, we train our network to evaluate the returned poses' accuracy. To do so, we add to the last module a parallel predictor that computes a confidence score c_i for each pixel starting from the same fused local and global embeddings.

To train this network, we need to define a distance between the predicted and the target rotations. To this end, we define a per-pixel orientation error as the angle of the rotation that aligns the estimated and ground truth orientations. We can compute the per-pixel loss as follows using the distance between the two orientations, represented as quaternions:

$$L_i^R = 2 * \arccos |\langle\, q_{target},\, \hat{q_i}\,\rangle|. \tag{3}$$

To simplify this loss function in order to make it easier for the optimization algorithm to update the weights, we can eliminate the inverse cosine function by rewriting the loss as follows:

$$L_i^R = 1 - |\langle\, q_{target},\, \hat{q_i}\,\rangle|. \tag{4}$$

According to [18], this is still a valid metric in $SO(3)$ that takes values in the range [0, 1]. Finally, we must combine the per-pixel losses to obtain a single value to be optimized. The overall loss can be defined as a weighted average of the per-pixel losses weighted over the per-pixel confidence score c_i, and we add a regularization term that penalizes predictions with small per-pixel confidences. We express the loss as:

$$L^R = \frac{1}{N}\sum_{i=1}^{N}(L_i c_i - \omega log(c_i)) \tag{5}$$

where N is the number of randomly sampled pixels from the RGB-D image and w is a balancing hyperparameter.

D. Loss function

Thanks to its modular design, the proposed model can be trained in two steps, first the translation block and then the rotation one. But to achieve higher accuracy, it is preferred to fine-tune the whole model jointly afterward. To perform joint finetuning, we need to define a loss function that considers the error in translation and rotation simultaneously. In literature, a common approach is to compute this metric by sampling random points on a 3D model of the object and computing the squared distance between these points in the ground truth and in the predicted 6DoF pose [7]. This assumes the existence of a 3D model of the object of interest that can be used to generate these points. Since this is not always possible, we designed our model with the modular architecture previously shown, which can be trained with only the 6DoF pose of the object.

In literature, to perform the jointed training, the loss to minimize for the prediction per-point is defined as:

$$L_i = \frac{1}{M}\sum_{j=1}^{M}\left\|(Rx_j + t) - (\hat{R}_i x_j + \hat{t}_i)\right\|_2 \quad (6)$$

where x_j denotes the j^{th} point of the M randomly selected 3D points from the object's 3D model, R and t are the rotational matrix and the translation vector that define the ground truth pose, and $\hat{R}_i$ and $\hat{t}_i$ define the predicted pose generated from the i^{th} point.

This loss function is only well-defined for asymmetric objects, where the object's shape or texture determines a unique canonical frame. Indeed, symmetric objects have more than one and possibly an infinite number of canonical frames, which leads to ambiguous minima. Therefore, we need to define an alternative loss function for symmetric objects. To this end, [7], [6], [19], and most of the works on pose estimation define the loss for symmetric objects as the distance between each point on the ground truth model and the closest point on the estimated model orientation. While the overall loss definition remains unchanged, the per-pixel loss function becomes:

$$L_i^S = \frac{1}{M}\sum_{j=1}^{M}\min_{0<k<M}\left\|(Rx_j + t) - (\hat{R}_i x_k + \hat{t}_i)\right\|_2. \quad (7)$$

Rotations that are equivalent to the 3D shape symmetry of the object are not penalized. But this implementation ignores the type of symmetry that characterizes the object underestimating incorrectly predicted rotations. This loss can be seen as a lower bound of the actual prediction error. To improve the previous formulation and provide a loss that better respects the roto translation error, we propose a new metric that minimizes the loss function over all the acceptable object symmetries. Instead of computing the distance with respect to the closest point, we calculate the classical loss function as defined in Equation 6 for every possible symmetric rotation of the predicted model, and we take the minimum of these values as the new per-pixel loss.

To define the new loss function in a rigorous way, we need to introduce some notations, as presented in [20]. In particular, we define:

1) *Order of symmetric rotation*: we say that an object has an n order of rotational symmetry around the axis θ, i.e. $\mathcal{O}(\theta) = n$, when its 3D shape is equivalent to its shape rotated by $\mathbf{R}_\theta\left(\frac{2\pi i}{n}\right) \quad \forall i \in \{0,\dots,n-1\}$, being $\mathbf{R}_\theta(\alpha)$ the rotational matrix corresponding to a rotation of an angle α around an axis θ.
   The min value of $\mathcal{O}(\theta)$ is 1, when the object has no symmetry around the θ axis. At the opposite, the order of symmetry of an object with a circular symmetry is infinite. A sphere has infinite order of symmetry around all the axes.
2) *Equivalent ViewPoint set*: we define the set of all equivalent ViewPoints with respect to a three-dimensional vector v around an axis θ as

$$E_o(\theta) = \left\{\mathbf{R}_\theta\left(\frac{2\pi i}{n}\right)v \quad \forall i \in \{0,\dots,n-1\}\right\}, \quad (8)$$

   with symmetry order $o \in 2, 3, \dots, \infty$.

Moreover, the order of symmetries across multiple axes is not independent. Indeed, the following properties hold for circular symmetries:

Proposition 1: If an object is not a sphere, then the following conditions must hold:

1) The object can have up to one axis with infinite order rotational symmetry.
2) If an axis θ has infinite order rotational symmetry, then the order of symmetry of any axis not orthogonal to θ can only be one.
3) If an axis θ has infinite order rotational symmetry, then the order of symmetry of any axis orthogonal to θ can be a maximum of two.

We can now give a formal definition of our loss. Given an object with rotational symmetry on the three orthogonal axes x, y and z equal to n_x, n_y and n_z. And given $\underline{k} = (k_1, k_2, k_3)$ we define an *equivalent ViewPoint* as

$$E_{\underline{k}}(\hat{R}_i x_k + \hat{t}_i) = \hat{R}_i\,\mathbf{R}_z\left(\frac{2\pi k_3}{n_z}\right)\mathbf{R}_y\left(\frac{2\pi k_2}{n_y}\right)\mathbf{R}_x\left(\frac{2\pi k_1}{n_x}\right)x_j + \hat{t}_i \quad (9)$$

with $k_1 \in \{0,\dots,n_x-1\}$, $k_2 \in \{0,\dots,n_y-1\}$ and $k_3 \in \{0,\dots,n_z-1\}$. Then, the per-pixel ViewPoint loss (VP-Loss) is defined as follows:

$$L_i^{VP} = \min_{\underline{k}\in K}\frac{1}{M}\sum_{j=1}^{M}\left\|(Rx_j + t) - E_{\underline{k}}(\hat{R}_i x_k + \hat{t}_i)\right\|_2 \quad (10)$$

with $K = \big\{(k_1,k_2,k_3) \mid k_1 \in \{0,\dots,n_x-1\}, k_2 \in \{0,\dots,n_y-1\}, k_3 \in \{0,\dots,n_z-1\}\big\}$. In summary, the ViewPoint loss computes the average distance between the target and predicted model as for non-symmetric objects but does the computation for all possible equivalent ViewPoints(i.e., all symmetric rotations of the model) and takes the minimum over them.

IV. Experimental results

Since the newly introduced loss is a core component of the object pose architecture, the first step of the experimental validation concern the validation of our loss function. Moreover, before comparing our model against other state of the art approaches, we also introduce a new evaluation metric inspired by the ViewPoint loss and designed to model the rotation error of symmetric objects better.

To validate the proposed loss, we compare it with the state of the art presented in Equation 6. For this task we consider a cube of size $5\,cm$ with square holes on the sides as shown in Figure 4, the order of symmetries around the main axes are $n_x = n_y = 2$ and $n_z = 4$. Indeed, we can rotate the object around the z axis of an angle $\alpha \in \left\{\frac{\pi}{2}, \pi, \frac{3\pi}{2}\right\}$ and the

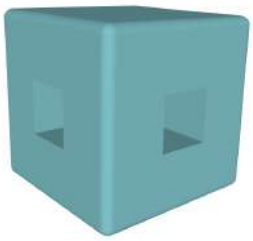


Fig. 4: 3D model of a cube with holes, used for the loss validation

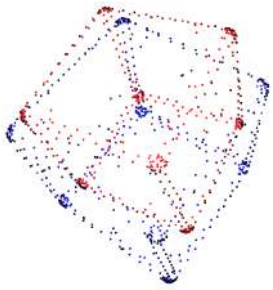


Fig. 5: 3D PointClouds of the target and predicted models

	Base Loss (Eq. 6)	**CP Loss (Eq. 7)**	**VP Loss (Eq. 10)**
Noise	*6.193 mm*	*4.773 mm*	*6.184 mm*
Rotation of π around z	*61.965 mm*	*4.675 mm*	*6.234 mm*
Rotation of $\frac{\pi}{4}$ around z	*25.145 mm*	*5.375 mm*	*22.567 mm*

TABLE I: Losses comparison on different configurations (i.e., rotation around the z axis) of the cube model from Fig. 4. The Base Loss is not defined for symmetric objects and does not consider equivalents ViewPoints. While both the Closest Point (CP Loss) and the ViewPoint loss (VP loss) are designed especially for symmetric objects.

cube would look the same. We consider two models of the same cube in the 3D space to compare the different losses. One represents the target and one the predicted cube. Then we uniformly sample 200 points from both and compute the distance between the points to obtain the loss, as shown in Figure 5. The results from the computation of the losses in some explicative scenarios are reported in Table I. As expected, the loss value from the closest point formulation is the smallest when a small noise is applied to the position and orientation of the object. If we add to the noise a rotation of π around z, then we have that the basic loss value for a non-symmetric object is very high since every point in the predicted model lies at the opposite side of the target, while the ViewPoint loss values in this setting are close to the corresponding value. This is because a rotation of π around z is one of the equivalent ViewPoints of the object, therefore, it is considered correct apart from the noise. On the other hand, a rotation of $\frac{\pi}{4}$ around z is not an equivalent configuration because of the holes. This results in a high ViewPoint loss that is close to the base loss since all the equivalent configurations are equally distant from the target model. Contrarly the state of the art closest point loss (CP loss), reported in the second column of Table I, does not distinguish so clearly between the different situations. In particular, in this last test, where we generated a rotation error of $\frac{\pi}{4}$, the computed loss is not significantly higher than a correct rotation, prooving the advantage of our ViewPoint loss against the state of the art approach.

A similar problem to the design of the loss emerges when the model has to be evaluated against other state of the art approaches. In particular, the most used metric to evaluate predictions are the Avarage Distance (ADD) and its variant for symmetric objects (ADD-S).

$$ADD = \frac{1}{M} \sum_{x \in \Theta} \left\| (Rx + t) - (\hat{R}x + \hat{t}\,) \right\|_2 \tag{11}$$

$$ADD_S = \frac{1}{M} \sum_{x_1 \in \Theta} \min_{x_2 \in \Theta} \left\| (Rx_1 + t) - (\hat{R}x_2 + \hat{t}\,) \right\|_2 . \tag{12}$$

where $[\hat{R}, \hat{t}\,]$ is the predicted 6DoF pose, $[R, t\,]$ the ground true pose, and x is a vertex out of M vertexes on the object mesh Θ. Since this metric is highly connected with the closest point loss, it has the same limitations previously explained. For this reason, we designed a custom metric to evaluate the model. In particular, we define the new metric taking inspiration from the ViewPoint loss defined in Equation 10, called *ADD-VP*. The ADD-VP evaluates the mean pair-wise distance between object vertexes transformed by the ground truth 6DoF pose $[R, t\,]$ and the closest predicted pose between all the equivalent ones, based on the object's symmetries. It is defined as follows:

$$ADD_{VP} = \min_{\underline{k} \in K} \frac{1}{M} \sum_{x \in \Theta} \left\| (Rx + t) - E_{\underline{k}}(\hat{R}x + \hat{t}\,) \right\|_2 \tag{13}$$

where $E_{\underline{k}}(\hat{R}x + \hat{t})$ represents the equivalent ViewPoint with parameters $\underline{k} = (k_1, k_2, k_3)$ as described by Equation 9. With this metric we can provide a more accurate evaluation of the model.

Finally, we compare our proposed architecture, trained with the ViewPoint loss, against the state of the art model DenseFusion [6] on the Linemod dataset [21], training both models for 500 epochs. To validate the results for asymmetric objects we adopt the ADD metric, like most works in literature. Regarding the two symmetric objects found in the dataset, we report both the ADD-S and the ADD-VP. The first one is required to make a fair comparison with the other models. On the other hand, the second one gives a more realistic evaluation of the error, and it is the one optimized in our implementation. In Table II, we summarize the percentage of correctly predicted poses on the test set of each object in the Linemod dataset. We fix the threshold for accepting a prediction as correct to 10% of the object's diameter, to be coherent with metrics used by state of the art models. For our implementation, we present both possible training modes. First, the percentage of correctly predicted poses obtained by combining the translation predictor, TNet, and the rotation predictor, RNet, which guarantees a simpler and easier implementation. Then, we include the outcome from the refinement obtained by training the full model using the novel ViewPoint loss.

The results show that our full model with ViewPoint loss significantly outperforms the state-of-the-art model on all the objects from the Linemod dataset trained on the same number of epochs. Similarly, the predictions from the model

TABLE II: Percentage of correctly predicted poses on Linemod test set of DenseFusion and our model's components (symmetric object in italic).

ID	Name	DenseFusion		TNet + RNet		Full Model	
	non-symmetric object *symmetric object*	ADD ↑ ADD-S ↑	ADD-PV ↑	ADD ↑ ADD-S ↑	ADD-PV ↑	ADD ↑ ADD-S ↑	ADD-PV ↑
1	Ape	0.6663		0.7560		**0.8651**	
2	Benchvise	0.7982		0.8256		**0.8852**	
4	Camera	0.6607		0.7367		**0.7993**	
5	Can	0.8277		0.8343		**0.8632**	
6	Cat	0.8832		0.8476		**0.9104**	
8	Driller	0.7889		0.8299		**0.8636**	
9	Duck	0.6629		0.7402		**0.7950**	
10	*Eggobox*	**0.9952**	0.0056	0.9733	0.9432	0.9821	**0.9512**
11	*Glue*	**0.9903**	0.0348	0.9523	0.9621	0.9635	**0.9678**
12	Holepuncher	0.6079		0.7228		**0.7644**	
13	Iron	0.9019		0.8726		**0.9351**	
14	Lamp	0.8829		0.8315		**0.9056**	
15	Phone	0.8347		0.8957		**0.9576**	
	TOTAL	0.8071		0.8261		**0.8769**	

that combines RNet and TNet without using the objects' 3D models are comparable to state-of-the-art models. It is also interesting to note how the proposed model, trained whith our custom loss, achieves comparable results to the state of the art on the ADD-S metric. Contrarly the DenseFusion model, trained with the closest point loss, performs poorly using the ADD-PV metric.

V. Conclusions

This paper presents a novel architecture for 6DoF object pose estimation. The proposed convolutional neural network is designed to have two independent branches, one for position and one for orientation estimate. In such a way, the two blocks can be trained separately, making the whole process simpler and less resource-demanding. Moreover, we introduce a new loss function specifically designed for symmetric objects (i.e., the ViewPoint Loss), which provides a better representation of the roto-translation error in these specific scenarios. The comparison with state of the art shows how the proposed method outperforms other approaches, and our loss help to produce more accurate predictions from the model.

References

[1] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, 2021.

[2] H. Uchiyama and E. Marchand, "Object detection and pose tracking for augmented reality: Recent approaches," in *18th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, 2012.

[3] C. K. Sahu, C. Young, and R. Rai, "Artificial intelligence (ai) in augmented reality (ar)-assisted manufacturing applications: a review," *International Journal of Production Research*, vol. 59, no. 16, pp. 4903–4959, 2021.

[4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[5] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *2011 international conference on computer vision*. IEEE, 2011, pp. 858–865.

[6] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3343–3352.

[7] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[8] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[9] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[10] Y. Wu, M. Zand, A. Etemad, and M. Greenspan, "Vote from the center: 6 dof pose estimation in rgb-d images by radial keypoint voting," in *European Conference on Computer Vision*. Springer, 2022.

[11] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6d object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3385–3394.

[12] M. Oberweger, M. Rad, and V. Lepetit, "Making deep heatmaps robust to partial occlusions for 3d object pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 119–134.

[13] K. Park, T. Patten, and M. Vincze, "Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[14] G. Wang, F. Manhardt, F. Tombari, and X. Ji, "Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 611–16 621.

[15] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 244–253.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2017.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[18] D. Huynh, "Metrics for 3d rotations: Comparison and analysis," 2019.

[19] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation," 2019.

[20] E. Corona, K. Kundu, and S. Fidler, "Pose estimation for objects with rotational symmetry," 2018.

[21] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Computer Vision–ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I 11*. Springer, 2013, pp. 548–562.

Learning State-Space Models for Mapping Spatial Motion Patterns

Junyi Shi[1] and Tomasz Piotr Kucner[1,2]

***Abstract*— Mapping the surrounding environment is essential for the successful operation of autonomous robots. While extensive research has focused on mapping geometric structures and static objects, the environment is also influenced by the movement of dynamic objects. Incorporating information about spatial motion patterns can allow mobile robots to navigate and operate successfully in populated areas. In this paper, we propose a deep state-space model that learns the map representations of spatial motion patterns and how they change over time at a certain place. To evaluate our methods, we use two different datasets: one generated dataset with specific motion patterns and another with real-world pedestrian data. We test the performance of our model by evaluating its learning ability, mapping quality, and application to downstream tasks. The results demonstrate that our model can effectively learn the corresponding motion pattern, and has the potential to be applied to robotic application tasks.**

I. Introduction

In recent years, the utilization of mobile robots has witnessed significant growth across various applications such as logistics, healthcare and exploration. Mapping, serving as a fundamental approach for modeling environmental information, plays a vital role in enabling robots to plan their movements, avoid obstacles, and locate targets. However, mobile robots still encounter limitations in dealing with changing environments. To allow mobile robots successfully navigate and operate in populated areas, it is necessary to develop methods for mapping dynamic information.

In daily life, it can be observed that individuals often adhere to implicit traffic rules while navigating their surroundings. Pedestrians exhibit distinct movement depending on their location, such as when traversing a corridor or approaching building entrances. Moreover, people from different regions tend to follow specific directional norms. For instance, individuals in the UK and Japan tend to favor the left side, while those in the US and Canada exhibit different behaviors. This observation naturally gives rise to a hypothesis: there exists a spatial motion pattern that guides the movement of pedestrians. With the map representation of these motion patterns, mobiles robot can benefit in a variety of applications such as motion planning [1], human motion prediction [2], task planning [3], and human-robot interaction [4].

Modelling these spatial motion patterns can be challenging. Previous studies are either based on the assumption that motion patterns remain constant within a given location [5] or undergo significant changes over extended periods [6]. However, such assumptions are somewhat divorced from reality. In reality, motion patterns tend to evolve gradually, as seen in an example of an underground station where the number of people does not remain constant, nor does it increase instantaneously. Instead, it changes gradually as the station approaches a certain rush hour.

In this paper, we adopt the assumption that dynamics within a changeable environment are driven by a certain kind of motion pattern that undergoes gradual changes over time. Our approach focuses on learning a map representation that describes the implicit motion pattern and its temporal variations. By leveraging data collected over successive time periods, our method can effectively learn the corresponding motion patterns and predict their subsequent movements.

Our contributions can be summarised as follows:

- We implement a generative model to describe the spatial motion pattern, which aggregates and encodes the spatial information of dynamics into a map representation.
- We employ a state-space model (SSM) to represent how the spatial motion pattern changes over time at a certain place.
- We demonstrate the predictive performance using our learned model, by evaluating the learning ability, the mapping quality and the model's applicability to downstream tasks.

II. Related Work

Our work is based on the concept of *Maps of Dynamics* (MoD), which refers to spatial or spatio-temporal representations of patterns of dynamics [7].

MoDs can be classified into different groups based on the type of dynamics being mapped. When considering discrete objects, they can be classified into three main groups: static objects, semi-static objects, and dynamic objects. [8]. Static objects, such as trees and buildings, rarely change position over long periods of time. In mapping systems, these are often represented using geometric maps, such as occupancy grid map [9] or OctoMap [10], which are not considered as MoDs. Semi-static objects, such as chairs and boxes, might change position within a relatively low frequency or as a consequence of specific events. Krajník et al. [6] introduce occupancy grids for mapping semi-static objects, combined with the temporal model Frequency Map Enhancement (FreMEn), in order to model the state changes of the semi-static cells. Dynamic objects, such as pedestrians and animals, are some objects that move purposefully and can be observed during the change of their states. Kucner et al. [11] and Wang et al. [12] treat dynamics as a change of occupancy

[1]Junyi Shi and Tomasz Piotr Kucner are with the Department of Electrical Engineering and Automation, Aalto University, Finland. junyi.shi, tomasz.kucner@aalto.fi

[2]Tomasz Piotr Kuncer is also with the Finnish Center of Artificial Intelligence, Finland.

979–8-3503-0704-7/23/$31.00 ©2023 IEEE

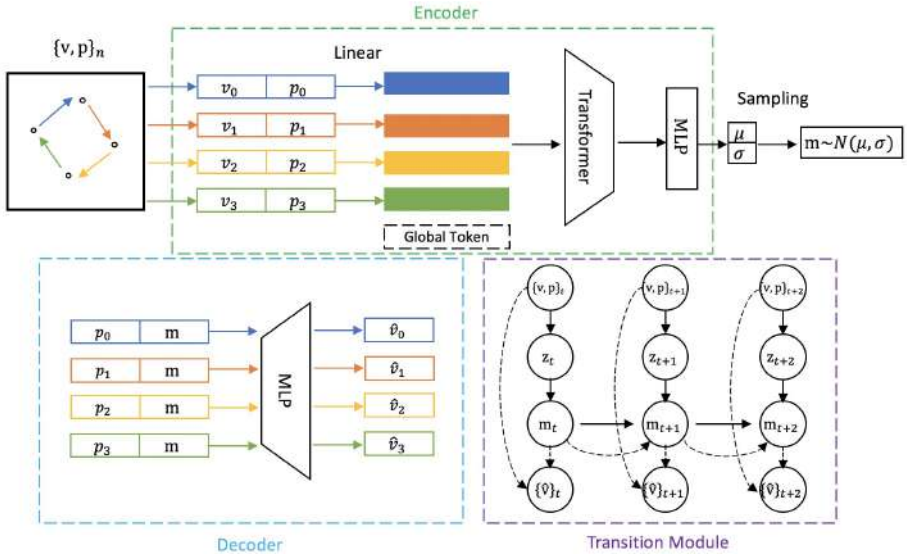


Fig. 1. Overview of our system. Our model consists of three major components: Encoder, Decoder and Transition Module. The encoder takes the pairs of velocity $\mathbf{v}$ and position $\mathbf{p}$ as input, where the Transformer converts a set of points to a single feature vector $\mathbf{z}$ and output a normal distribution. After sampling, the system obtains the motion pattern $\mathbf{m}$. The decoder takes position $\mathbf{p}$ and motion pattern $\mathbf{m}$ as inputs, and generate the reconstruction $\hat{\mathbf{v}}$. Observations from the past state are used to generate predictions into the future state in the transition module.

in grid map cells and construct models capable of grasping the spatial relation between the states of neighboring cells. Dynamic objects can also be modelled by their trajectory, as explored by Bennewitz et al. [13] and Ellis et al. [14], or represented by velocity fields, as proposed by Verdoja et al. [15] and CLiFF-Map [5].

Traditionally, state-space models have been used to produce estimates of currently unknown state variables based on their previous observations [16]. As a common approach, it is widely used in applications such as state estimation [17], target tracking [18] and navigation [19]. In recent years, deep sequential generative models are appealing as temporal models, which have shown impressive performance in various types of inference tasks, such as system identification [20], geometric mapping [21]. By learning from past experience, it can be applied to model environmental dynamics and uncertainty due to the probabilistic nature of the model.

In this paper, we focus primarily on mapping spatial motion patterns of dynamic entities. In contrast to previous studies, we adopt the assumption that the motion patterns of these entities undergo gradual changes within short time frames, and can be implicitly represented. We propose a deep sequential generative model specifically designed for the MoD problem, by learning a state-space model that represents the underlying motion patterns based on past experiences.

III. Methodology

A. Problem Setup

In our work, we employ a motion probability distribution to represent the dynamics. The motion distribution, denoted as $\mathcal{M}$, is defined as a conditional distribution of velocity $\mathbf{v}$ given position $\mathbf{p}$:

$$\mathcal{M} = p(\mathbf{v} \mid \mathbf{p}), \tag{1}$$

where $\mathbf{p}$ is a 2D Euclidean vector denoting the position. The velocity $\mathbf{v}$ is using a polar coordinate frame, which combines the orientation ψ and speed ρ:

$$\mathbf{v} = (\psi, \rho)^\top, \psi \in [-\pi, \pi) \tag{2}$$

By using a polar representation rather than a 2D Euclidean vector representation, each component of the velocity vector has an explicit physical meaning and can be analyzed independently.

At each time step t we are interested in, we make the assumption that there are no changes in the underlying motion pattern. We observe n points at the time step t in the form of $\{\mathbf{v}, \mathbf{p}\}$, which can be viewed as samples from the joint distribution:

$$p(\mathbf{v}_t, \mathbf{p}_t) = p(\mathbf{p_t})p(\mathbf{v}_t \mid \mathbf{p}_t) \tag{3}$$

We further assume that the dynamics in the given location are driven by a spatial motion pattern $\mathbf{m}$, which serves as a parameter of the motion distribution. Different values of $\mathbf{m}$ give rise to distinct motion distributions. The joint distribution with $\mathbf{m}$, conditioned upon Equation (3), can be expressed as follows:

$$p(\mathbf{v}_t, \mathbf{p}_t \mid \mathbf{m}) = p(\mathbf{p}_t)p(\mathbf{v}_t \mid \mathbf{p}_t, \mathbf{m}), \tag{4}$$

where we assume the position $\mathbf{p}$ is independent of $\mathbf{m}$.

The purpose of this formulation is to estimate the motion distribution based on the given set of observations. In practical implementation, we utilize Gaussian distributions for all the distributions in our formulation. Specifically, we employ a neural network that outputs both the mean and variance of the Gaussian distribution, allowing us to learn and estimate the parameters of the motion distribution.

B. Network Structure

In our work, we employ variational inference and amortized inference [22] techniques to address the problem. The neural network utilized in our approach consists of three distinct components: the approximate posterior distribution defined as $q_\phi(\mathbf{m} \mid \{\mathbf{v}, \mathbf{p}\}_n)$, the prior distribution $p_\theta(\mathbf{m})$ and the emission model $p_\theta(\mathbf{v} \mid \mathbf{p}, \mathbf{m})$.

To handle the varying number of observations at each time step, we introduce the concept of a set feature extractor. The set feature extractor converts a set of points to a single feature vector: $\mathbf{z} = f(\{\mathbf{v}, \mathbf{p}\}_n)$. The set feature extractor allows us to align the encoder with other components and simplify the dependencies on the motion set $\{\mathbf{v}, \mathbf{p}\}_n$.

Based on this, the posterior is split into two parts. First, a set feature extractor is applied to convert the set into a vector representation. Then, Multilayer Perceptrons (MLPs) are applied to compute the mean and variance of the posterior distribution. There are multiple options available for the set feature extractor, we choose Transformer [23] as the extractor for its reliable performance.

In most cases, the variational autoencoder (VAE) does not require the learning of the prior distribution, a standard Gaussian $\mathcal{N}$ can be simply utilized. Furthermore, a specialized decoder can only be implemented with a known state structure, such as the motion pattern $\mathbf{m}$ in our case. Therefore, we utilize a common flatten decoder, which is a combination of several MLPs.

We employ *evidence lower bound* (ELBO) [24] as the objective function, which is given as:

$$\begin{aligned}\mathcal{L}_{elbo} =& \mathbb{E}_{\{\mathbf{v},\mathbf{p}\}_n \sim D}\Big[\mathbb{E}_{\mathbf{m}\sim q_\phi(\mathbf{m}|\{\mathbf{v},\mathbf{p}\}_n)} \\ & [\frac{1}{n}\sum_{i=1}^{n} -\log p_\theta(\mathbf{v}_i|\mathbf{p}_i,\mathbf{m})] + \\ & D_{KL}[q_\phi(\mathbf{m}|\{\mathbf{v},\mathbf{p}\}_n)||p_\theta(\mathbf{m})]\Big],\end{aligned} \tag{5}$$

The ELBO provides a lower bound on the marginal likelihood, which is intractable to compute directly. Maximizing the ELBO is equivalent to minimizing the Kullback-Leibler (KL) divergence between the approximated posterior and the true posterior.

C. Sequential Modelling

Since we assume that there is a underling law that guiding the changes of motion pattern, we can extend our model to handle sequential data using the state-space model formulation.

To accomplish this, we extend the posterior and the prior distributions to a sequential form, which can be expressed as follows:

$$\begin{array}{lll} \text{Posterior:} & \mathbf{m}_{t+1} & \sim q_\phi\left(\mathbf{m}_{t+1} \mid \mathbf{m}_t, \{\mathbf{v}, \mathbf{p}\}_{n_{t+1}}\right) \\ \text{Prior:} & \mathbf{m}_{t+1} & \sim p_\theta\left(\mathbf{m}_{t+1} \mid \mathbf{m}_t\right) \end{array} \tag{6}$$

In the sequential modelling, the decoder remains the same as the VAE model. Specifically, we employ a recurrent state-space model (RSSM) proposed by Hafner et. al [25], which is one of the state-of-the-art SSMs. Typically, transitions in a recurrent neural network are purely deterministic, while transitions in a state-space model are purely stochastic. RSSM uses a mix of deterministic and stochastic latent state, which allow the model itself to robustly learn to predict multiple future states. For the SSM, we also utilize MLPs to compute the mean and variance. The whole structure of the model is shown in Figure 1.

IV. Experiments

We evaluated three aspects of our MoDs: the learning ability, the mapping quality and the model's potential applicability to downstream tasks.

The model described in Section III is implemented in PyTorch [26]. We employed GRU [27] as the recurrent neural network (RNN) in our model for the deterministic transition. The dimension of the set feature extractor is 256, the dimension of the hidden state for encoder is 1024 and decoder is 256, the dimension for the deterministic transition is 512, the dimension for the stochastic transition is 256 and the dimension of the latent variable is 256.

A. Evaluation of Learning Ability

We started our evaluation with a generated toy dataset, which has clear, explicit motion patterns. A vortex-like pattern is defined as:

$$\dot{\rho} = 0.5\rho, \tag{7}$$

$$\dot{\psi} = \psi + \frac{\pi}{2}. \tag{8}$$

The velocity fields of the vortex pattern are generated using `scipy.integrate.odeint` [28], as shown in Fig. 2.

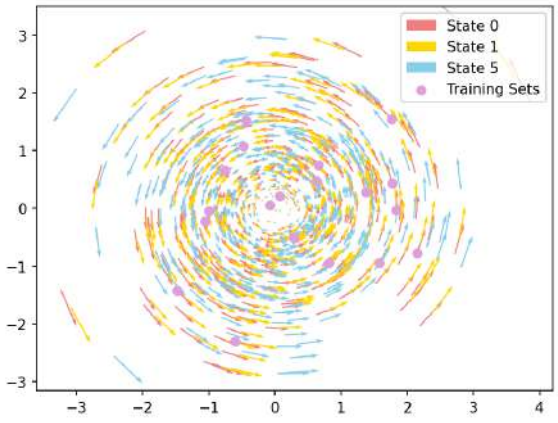


Fig. 2. The toy vortex dataset.

We trained the model with a batch size of 16 in 500 epochs using AdamW [29] with an learning rate of 0.001. In order to simulate a realistic scenario, only 20 randomly selected velocities in every time step were used as the training set. 20 time steps were used for training, the model observed 5 time steps and predicted 5 time steps or 20 time steps in the experiment.

As shown in Fig. 3, tests were done on another generated vortex dataset, which demonstrated the performance of

our model in a similar scenario. We analysed the velocity predicted by our model, considering the magnitude of the velocity error at the ground-truth location. Two metrics were used: Average Velocity Error (AVE) and Final Velocity Error (FVE). The former metric is calculated by the mean square error (MSE) over all estimated points of the states and the true points, while the latter one is calculated at the predicted final state.

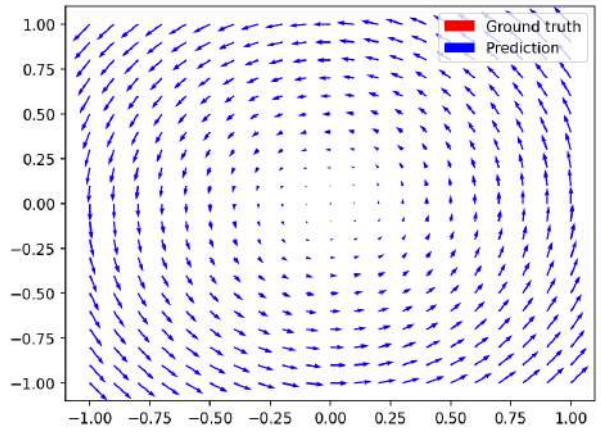


Fig. 3. Results of a vortex-like spatial motion pattern. The output of the proposed method is shown in blue, which overlaid with the ground truth in red. The arrows indicate the magnitude and orientation of the velocity field at this future time.

The results are shown in Table I, where we tested the errors from the SSM compared to the baseline VAE. SSM demonstrated a strong learning capability, performing well in both AVE and FVE metrics. As the other parameters of the two networks are identical, SSM only adds the transition module, so it can be assumed that this increases the ability of our model to learn changes in motion patterns over time.

TABLE I
EXPERIMENTAL RESULTS ON VORTEX DATASET

Model	Horizon	AVE	FVE
VAE	5 time step	0.00591	0.00595
	20 time step	0.00601	0.00584
SSM	5 time step	0.00004	**0.00006**
	20 time step	**0.00003**	0.00007

B. Quantitative Evaluation

Experimenting in a simulated environment was not enough, so we further introduced real-world datasets for evaluation. However, in real scenarios, we are unable to obtain true values of the motion patterns. Therefore, we implemented the quantitative evaluation to assess the mapping quality of the representation.

The ATC dataset [30] was used in the experiments, which comprised real pedestrian data from the Asia and Pacific Trade Center in Osaka, Japan. The dataset was obtained using a tracking system comprising numerous 3D range sensors, covering an area about 900 m^2. The data collection took place over 92 days between 24 October 2012 and 29 November 2013, specifically on Wednesdays and Sundays, between the hours of 9:40 and 20:20. The spatial geometric map for the environment is shown in Figure 4, which contains a long corridor and several entrances.

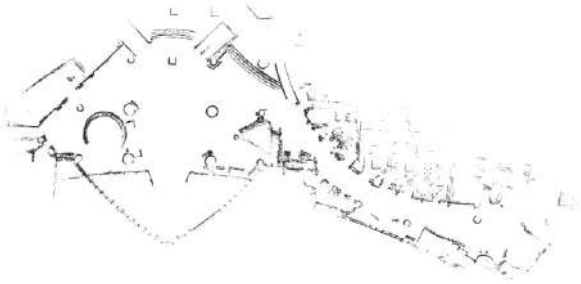


Fig. 4. The occupancy grid map of the ATC shopping mall.

A divergence estimator proposed by Wang et al. [31] was used to provide the quality of the map in absolute values. Wang's divergence estimator computes the differences between the output of the model and the original data. For each query location in the map, we obtain a motion distribution from the output of the encoder. Simultaneously, we also have a set of observations $\{\mathbf{v}_1^o, \ldots, \mathbf{v}_n^o\}$ from the given dataset. Wang's divergence estimator was then employed to estimate the divergence between the above two distributions, which employed only the samples coming from them. The estimator is given as follows:

$$\hat{D}_{n,m}\left(\mathcal{M}^{'} \| \mathcal{M}\right) = \frac{d}{n}\sum_{i=1}^{n}\log_2\frac{v_k(i)}{\rho_k(i)} + \log_2\frac{m}{n-1} \quad (9)$$

In the divergence estimation, the distance $\rho_k(i)$ between $\mathbf{v}_i^o$ and its k-NN in $\left\{\mathbf{v}_j^o\right\}_{j\neq i}$ is compared with the distance $v_k(i)$ between $\mathbf{v}_i^o$ and its k-NN in $\left\{\mathbf{v}_j^q\right\}$, where $\left\{\mathbf{v}_j^q\right\}$ denotes the observations queried from the component of the model.

We retrained the model with a batch size of 16 in 1000 epochs, 20 days are used for training, 5 for validation, a Sunday set and a Wednesday set for evaluation. We take half an hour as a time step, and divide the day into 20 time steps. We employed CLiFF-Map [5] as a baseline method for comparison, which were trained in different time steps. We set k=1 in practice, which means the algorithm only considers the closest single neighbor to the new data point.

The result of the quantitative evaluation is shown in Table II. On Sundays, the observations are roughly twice as high as on Wednesdays and the time period starting at 12AM is usually the peak of crowd density. Both methods are influenced by changes in observations, and our method is less sensitive to population density than CLiFF-Map. Since this experiment is actually comparing the ability to aggregate information (either through clustering in CLiFF-Map or through the set feature extractor in our method), it can be shown that our method is more robust to the number of observations.

TABLE II
QUANTITATIVE EVALUATION RESULTS

Model	Horizon	Sun		Wed	
		Obs.	Div.[bit]	Obs.	Div.[bit]
CLiFF [5]	11AM-	1361895	0.3094	655704	0.3958
	12AM-	2136236	**0.2814**	1046258	0.3542
	Avg.		0.3024		0.3765
SSM	11AM-	1361895	0.3178	655704	0.3624
	12AM-	2136236	0.2962	1046258	**0.3266**
	Avg.		0.3094		0.3478

C. Applicability to Downstream Task

Pedestrian motion prediction is used as a case for evaluating the applicability of our model to downstream tasks. For non-myopic robotic navigation, it's important that the prediction is made over the entire duration to the destination. In practice, we try to simulate the following scene: a service robot walking down a corridor in a small room, possibly for about 4.8 seconds; and an operating robot walking down a longer corridor in a factory, possibly for about 20 seconds. Therefore, we consider the horizon length over 4.8s and 20s in the rather larger indoor setting of interest, which some current research is lacking at these time spans.

We see our work as macroscopic works, to distinguish it from some microscopic works. Traditional metrics for pedestrian trajectory prediction using microscopic features are Average Displacement Error (ADE) and Final Displacement Error (FDE). ADE is calculated by the mean square error (MSE) over all the displacement in position per person between the prediction and the ground-truth data in the whole trajectory and FDE is calculated at the final endpoint. We use the mean value of the generated distribution to calculate the error and compare it with microscopic methods.

For this task, 0.1s was chosen as a time step and the network observed 50 time steps (5s) in the experiment. We retrained the model with a batch size of 16 in 100 epochs. As shown in Fig. 5, the observations in the eastern long corridor of ATC dataset is used for training and evaluation. The results of the applicability in pedestrian motion prediction is shown in Table III. We compared our method with the state-of-the-art motion prediction algorithm Social GAN (SGAN) [32]. SGAN obtains values at every 0.4 seconds and it was designed and trained for 12 time steps (4.8s), when it can get its best performance. As a microscropic method, SGAN generates associated predictions for every pedestrians, but our method has no concept of individual pedestrians for inputs, which is somehow unfair.

In a real-world robotics application, we can easily determine the direction in which a pedestrian is moving by using sensors. Therefore, we also trained and evaluated our model in only one direction, that is, only consider the orientation ψ in domain $[0, \pi)$ to get a fair comparison. The experimental results demonstrate that our model achieves high accuracy for FDE and long-horizon ADE metrics. However, our model exhibits slight underperformance compared to SGAN for short-horizon ADE. One possible explanation is

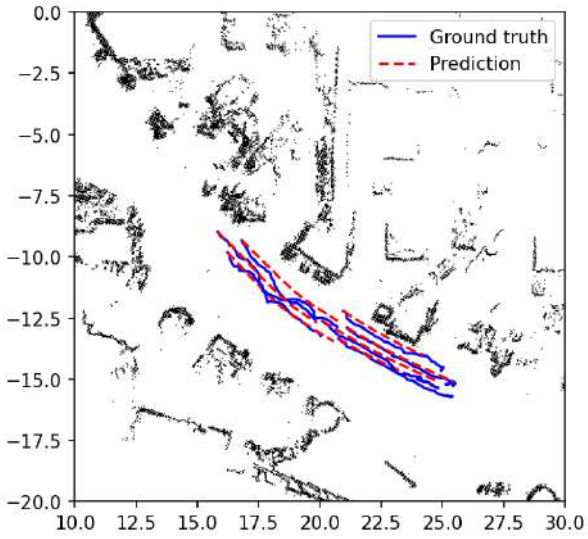


Fig. 5. Visualization of the pedestrian motion prediction. The observations in eastern long corridor of ATC dataset is used for training and evaluation.

TABLE III
RESULTS OF PEDESTRIAN MOTION PREDICTION

Model	Horizon	Sun		Wed	
		ADE(m)	FDE(m)	ADE(m)	FDE(m)
SGAN [32]	4.8s	**0.6382**	1.1896	**0.6163**	1.0758
	20s	1.8956	3.6783	1.9464	3.8433
SSM	4.8s	1.1084	2.3260	1.3941	2.7436
	20s	2.2147	3.9748	1.8420	3.3368
SSM (one direction)	4.8s	0.6824	**0.8892**	0.6761	**0.9630**
	20s	**0.7223**	**0.9908**	**0.8828**	**1.0491**

that our model outputs velocity rather than directly providing trajectory information. As a result, we need to integrate the velocity outputs of each time step to obtain the corresponding position, which is then used as input for the subsequent time step. The above process may introduce additional error, which could contribute to our model's reduced performance. In addition, note that our model was not designed for the motion prediction task, but we can still see that it maintains a certain level of accuracy, which is an encouraging indication of its applicability to downstream tasks.

V. CONCLUSION

In this paper, we presented a method for learning the motion patterns in a changeable environment. The proposed model, leverages a set feature extractor to aggregate spatial information of the input data, a variational autoencoder to encode the spatial information, and a transition module to learn the temporal information. We demonstrated the effectiveness of our method through several experiments, which shows that our model is possible to map the dynamics and be applied in downstream robotic application.

So far, we have utilized only the temporal and spatial information of dynamic objects, without considering the effects of static and semi-static objects. In the future work, we intend to integrate information from static and semi-

static objects to provide better environmental information for robotic applications. Additionally, the position-based velocity fields may vary in different environments. Therefore, introducing semantic information to model motion patterns in diverse environments is another future direction of research.

ACKNOWLEDGEMENTS

The authors express their gratitude to Xingyuan Zhang from the Machine Learning Research Lab of Volkswagen Group for his valuable discussions on topics related to this work.

REFERENCES

[1] C. S. Swaminathan, T. P. Kucner, M. Magnusson, L. Palmieri, S. Molina, A. Mannucci, F. Pecora, and A. J. Lilienthal, "Benchmarking the utility of maps of dynamics for human-aware motion planning." *Frontiers in Robotics and AI*, vol. 9, 2022.

[2] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[3] F. Surma, T. P. Kucner, and M. Mansouri, "Multiple robots avoid humans to get the jobs done: An approach to human-aware task allocation," in *2021 European Conference on Mobile Robots (ECMR)*. IEEE, pp. 1–6.

[4] M. Hanheide, D. Hebesberger, and T. Krajnik, "The when, where, and how: An adaptive robotic info-terminal for care home residents - a long-term study," in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2017, pp. 341–349.

[5] T. P. Kucner, M. Magnusson, E. Schaffernicht, V. H. Bennetts, and A. J. Lilienthal, "Enabling flow awareness for mobile robots in partially observable environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1093–1100, 2017.

[6] T. Krajník, J. P. Fentanes, J. M. Santos, and T. Duckett, "Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 964–977, 2017.

[7] T. P. Kucner, A. J. Lilienthal, M. Magnusson, L. Palmieri, and C. S. Swaminathan, *Probabilistic mapping of spatial motion patterns for mobile robots*. Springer, 2020.

[8] D. Meyer-Delius, J. Hess, G. Grisetti, and W. Burgard, "Temporary maps for robust localization in semi-static environments," in *2010 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2010, pp. 5750–5755.

[9] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," vol. 2, pp. 116–121, 1985.

[10] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[11] T. Kucner, J. Saarinen, M. Magnusson, and A. J. Lilienthal, "Conditional transition maps: Learning motion patterns in dynamic environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1196–1201.

[12] Z. Wang, R. Ambrus, P. Jensfelt, and J. Folkesson, "Modeling motion patterns of dynamic objects by iohmm," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1832–1838.

[13] M. Bennewitz, W. Burgard, and S. Thrun, "Learning motion patterns of persons for mobile service robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 4. IEEE, 2002, pp. 3601–3606.

[14] D. Ellis, E. Sommerlade, and I. Reid, "Modelling pedestrian trajectory patterns with gaussian processes," in *2009 IEEE 12th international conference on computer vision workshops, ICCV workshops*. IEEE, 2009, pp. 1229–1234.

[15] F. Verdoja, T. P. Kucner, and V. Kyrki, "Generating people flow from architecture of real unseen environments," *arXiv preprint arXiv:2208.10851*, 2022.

[16] S. Särkkä, *Bayesian filtering and smoothing*. Cambridge university press, 2013, no. 3.

[17] S. Sarkka, "On unscented kalman filtering for state estimation of continuous-time nonlinear systems," *IEEE Transactions on automatic control*, vol. 52, no. 9, pp. 1631–1641, 2007.

[18] S. Särkkä, A. Vehtari, and J. Lampinen, "Rao-blackwellized particle filter for multiple target tracking," *Information Fusion*, vol. 8, no. 1, pp. 2–15, 2007.

[19] M. S. Grewal, L. R. Weill, and A. P. Andrews, *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons, 2007.

[20] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung, "Deep state space models for nonlinear system identification," *IFAC-PapersOnLine*, vol. 54, no. 7, pp. 481–486, 2021.

[21] A. Mirchev, B. Kayalibay, P. van der Smagt, and J. Bayer, "Variational state-space models for localisation and dense 3d mapping in 6 dof," *arXiv preprint arXiv:2006.10178*, 2020.

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[24] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," in *Learning in graphical models*. Springer, 1998, pp. 105–161.

[25] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2555–2565.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[27] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[28] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[29] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*.

[30] D. Brščić, T. Kanda, T. Ikeda, and T. Miyashita, "Person tracking in large public spaces using 3-d range sensors," *IEEE Transactions on Human-Machine Systems*, vol. 43, no. 6, pp. 522–534, 2013.

[31] Q. Wang, S. R. Kulkarni, and S. Verdú, "Divergence estimation for multidimensional densities via k-nearest-neighbor distances," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2392–2405, 2009.

[32] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.

Monocular Person Localization with Lidar Fusion for Social Navigation

Sedat Dogru, Carlos A. Silva, Lino Marques *Member, IEEE*

***Abstract*— Smooth social navigation requires not only detection of the people around the robot, but also accurate localization of the people, a process difficult to achieve with a single sensing modality. Hence, literature has focused on various fusion approaches, such as RGB-D, ROI based lidar vision fusion, or Artificial Neural Network (ANN) based lidar vision fusion. However, monocular photogrammetry has always been ignored in the literature. In this work, we propose a fusion approach based on monocular positon estimation and lidar, and show the effectivenes of the approach both on a public dataset and a purpose built dataset with different cameras.**

I. INTRODUCTION

Navigation has always been an important aspect of mobile robots; and with the increasing deployment of robots in human dominated spaces, socially acceptable navigation has become more important. Socially acceptable navigation is defined as human friendly navigation around them, and it consists of a collection of behaviours: Maintaining a low speed around humans, respecting personal [1] or group space [2], respecting passage priorities, or following motion norms, such as going through the right side of a corridor when faced with an incoming person [3], [4], taking into account communication [5], and even context [6] are some example behaviours that have been studied in the literature. In order to work effectively, these behaviours require the robot be able to detect humans and estimate their position, taking them into account in the navigation algorithm.

Person detection can be done using various sensors, such as Infrared (IR) cameras, RGB cameras, RGBD cameras, radar, and lidar. Vision based approaches focus on detecting features to help identify human body and body parts, such as face, or use Deep Neural Network (DNN) based approaches [7], [8], which internally generate some features to train detection. RGBD based approaches in some cases use only dense depth data [9], which may restrict detection range due to the relatively shorter depth range of the RGBD cameras. 3D lidars, despite their relatively sparser measurements, are also used to detect humans [10]. 2D lidars provide sparser measurements than 3D ones. However, they have also proven useful in person detection through indirect measurements. 2D lidar data has been used to detect legs of people [11], to identify possible human specific features in the data [12] and commonly to detect mobile agents, assuming that the only mobile agents in the environment are the humans [13].

Among the mentioned person detection methods, 2D/3D lidar as well as depth or image based approaches using RGBD cameras allow direct measurement of the human position in 2D or 3D. Lewandowski et al. [9] used depth information from RGB-D cameras, proposing an approach which directly operates in the metric $3D$ space, extracting candidate clusters and classifying them with an SVM, using features such as VFH and 3DmFV. Hacinecipoglu et al. [14] proposed an approach that focuses on the detection of the head in the depth information acquired by an RGB-D camera, based on the rationale that a person's head is the least likely body part to be occluded. There are other approaches focused on general point cloud object detection and classification which can be used for detecting humans, such as PointNet [15], PointPillars [16] or AFDetV2 [17]. These methods rely on computationally expensive deep-learning based methodologies such as neural networks that directly process the point cloud as it is, which may fail to achieve real-time performance with modest hardware. Although lidars are able to provide highly accurate distance measurements, with increasing distance point density decreases, decreasing angular resolution and even detection probability. Particularly in close to ground 2D lidar configurations, where the lidars are used to detect legs, the legs occupy a very small angular range and hence their detection becomes probabilistic. Image based detection approaches without an integrated depth sensor can utilize a stereo system to measure distance using the stereoscopic image processing principle [18] or use monocular photogrammetry in special cases to measure distance [19], [20]. These approaches can also benefit from lidar and camera fusion, which would allow utilizing the high measurement accuracy of lidars [21]–[23]. Yet another approach in navigation context is to avoid an explicit estimate of the distance, but instead train a complex Artifical Neural Network (ANN), such as a Deep Neural Network (DNN), using a high number of training samples to infer motion commands that would result in safe navigation [24].

Sensor fusion using vision and lidar has been studied, creating a corresponding depth image, or merging features obtained in both domains. Spinello et al. [25] use extrinsic camera calibration to project 2D lidar clusters to the image. They use multidimensional features that describe geometric properties in lidar data and a grid of Histogram of Oriented Gradients (HOG) to detect humans in the image, learning both through SVM, and fusing both. Premebida et al. [26] use feature maps obtained from RGB images and upsampled

This work was supported by projects "UltraBot - Robô para desinfecção por radiação ultravioleta", under Grant CENTRO-01-0247-FEDER-072644 and "ILAF - Intelligent Logistic Autonomous Fleet", under Grant POCI-01-0247-FEDER-072534.

The authors are with Institute of Systems and Robotics, Department of Electrical and Computer Engineering, University of Coimbra, 3030-290 Coimbra, Portugal {sedat, carlos.silva, lino}@isr.uc.pt

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

lidar data and input to a deformable part model detector. González et al. [27] fuse 3D lidar and RGB image, filtering in the part of the point cloud that falls in to the camera FoV, interpolating those points to obtain a dense depth map, after which HOG and Local Binary Pattern (LBP) features over both the image and the depth are used in a detector. Schlosser at al. [28] extract depth features from upsampled lidar data and merge with RGB image in a CNN to detect pedestrians. Bozorgi et al. [22] instead of directly fusing lidar and RGBD data, merge tracks independently obtained through RGBD and lidar person detectors. Silva et al. [21] fused RGB and lidar data, keeping lidar data that falls in to the bounding box, and then studied several feature based methods to remove foreground and background points that do not belong to the human, to eventually localize it. lidar and vision have been fused to detect humans, however the fusion has been through projection and merging features, sometimes through an ANN. A problem that is faced with such a fusion is background or foreground objects, which are detected by the lidar and appear in the field of view of the camera after lidar to camera projection.

Monocular vision by itself lacks the capability to measure distance to objects, which is caused by the inherent scale ambiguity, and therefore it has been little studied. However, taking into account extra information on the size or position of the objects, helps resolve the scale ambiguity. Kundegorski and Breckon [19], using monocular infrared cameras, and the observation that statistically seen the height of most humans lie in a narrow range producing a bounded and acceptable error in human position estimation, localized people succesfully and compared it to GNSS. Niu et al. [20] used a lidar to measure the height of the ground plane on which the persons are standing to resolve the ambiguity, and used YOLO reporting a bounding box to detect people and estimate their position through the camera.

Monocular distance estimates can provide a valuable contribution to the existing fusion algorithms, helping improve quality of the lidar point cloud by reducing the outliers significantly. In this work we propose a range based fusion of lidar measurements (2D/3D) with monocular distance estimates for person position estimation, aiming particularly social navigation scenarios. In this work, person detection is performed using a vision based approach, namely a human segmentation based on YOLOv7-seg [29]. The monocular distance measurement is achieved using monocular photogrammetry principles with the assumption that both the humans and the robot occupy the same operational space, and known camera height for the robot.

In this work we show a fusion approach utilizing the strength of each sensor. At close range, where the target occupies a large portion of the field of view of the camera, monocular distance estimate is not possible. However, we can use the transformation between the two to identify the clusters and hence estimate the distance using lidar only. In the mid-range, monocular distance estimates have relatively low error, and the lidar is still dense, hence a position estimate is provided using both. At long-range, the point density of the lidar decreases, giving only random detections particularly in a 2D lidar placed close to the ground. Hence, despite the decreased accuracy, monocular estimates are used. We show that although the error in the monocular estimate increases with the square of the distance of the target from the object, vision based human detection performance, which may suffer due to poor lighting, is also a contributor to the overall error. At mid-range, the monocular localization helps filter out unrelated clusters that fall in to the ROI, which would normally require a classification step. In this work, we study the performance of the proposed approach, and compare it to the state of the art, using data from the JRDB dataset [30], and also data collected by us with fully known ground-truth using two different cameras in a corridor, measuring distances up to 25 m. We use the recently proposed RGB lidar fusion approach [21].

II. Method

A. Monocular Position Estimate

1) Mathematical Background: Monocular position estimation algorithm assumes that the person and the robot are on the same ground surface, and the height of the camera, h_c, in addition to the camera calibration matrix, $\mathbf{K}$, are known (Fig. 1). Assuming (x, y, z) is a point in 3D world in camera coordinates, and (u, v) is its corresponding image on the image plane of the camera, with the origin of the image plane being the upper left corner of the image, for Pinhole camera $\mathbf{K}$ is given by

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

and the following relationship holds [18]

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} -x/z \\ +y/z \\ 1 \end{bmatrix} \tag{2}$$

In the above relationship, z is the vertical distance from the camera, and it is the reason for the scale ambiguity in monocular position estimation. Focusing on estimation position of the feet, rather than the torso, which are equivalent in terms of navigation on a plane, it can be assumed that at least one feet is on surface, and hence the vertical distance between the camera and the feet, y, is equivalent to the height of the camera h_c. This in turn allows solving the above equation as

$$y = h_c \tag{3}$$

$$z = f\frac{h_c}{v - c_y} \tag{4}$$

$$x = \frac{z}{f}(u - c_x) \tag{5}$$

for the position of the feet in camera coordinates.

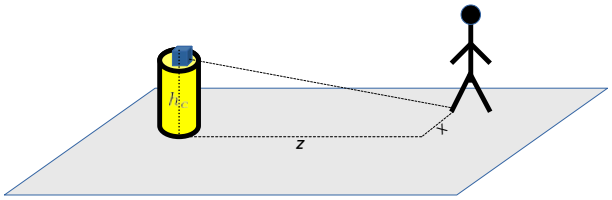


Fig. 1. Photogrammetric Pose Measurement, the robot (yellow) is on the left with a camera (blue) at a height of h_c. In a social navigation scenario, the people and the robot occupy the same floor.

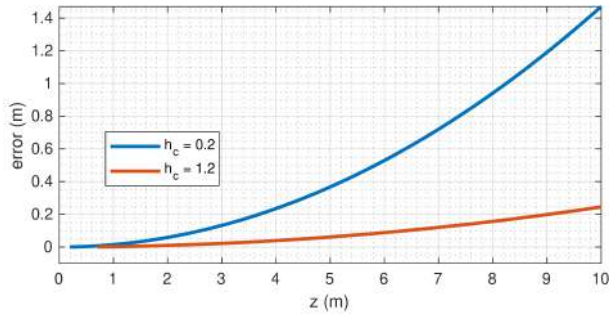


Fig. 2. Error in monocular position estimate as a function of distance and height (h_c) of a camera with 60^o vertical FoV.

2) Error Analysis for Monocular Position Estimate: The above approach is usable when the feet are in the FoV of the camera, i.e. they are far enough with $z \geq h_c * \tan\frac{\theta}{2}$, with θ being the vertical FoV. Hence, larger θ or smaller h_c improve the close range of the method (Fig. 3). In order to find the long range sensitivity of the method, we rewrite (4) as

$$v = \frac{fh_c}{z} + c_y \tag{6}$$

and calculate $\frac{d}{dz}$ and re-arrange the terms, giving in magnitude

$$|dz| = \frac{1}{fh_c} z^2 |dv| \tag{7}$$

This implies that the estimation error in z increases quadratically with distance from the camera. Note that, the (u, v) space is quantized, hence a point in 3D is anywhere within a cell, with the cell coordinates representing the position of the object with at most half cell size of error. Additional errors in cell boundaries introduced due to the camera optics, or image processing algorithms, will increase dv. Increasing h_c can be seen to be decrease the error in z estimate. However, this comes at a price of increasing the non-usable space close to the camera.

The error can also be seen in Fig. 2, where it is calculated for an Intel RGBD camera. The closer field of view deteriorates with moving the sensor up, but the distance measurement accuracy improves, particularly for the long range.

3) Pipeline: The pipeline is summarized in Alg. 1, and the intermediate steps for a sample are shown in Fig. 4. The process starts with the calculation of a lookup table of depth (i.e. z-distances in camera coordinates) for the lower half of the camera image, to improve computational performance by reducing the need to recompute the z-distances again and again for different frames along the life time of the localization process. Due to 1-point projection, the upper half of the image cannot contain any point from the ground plane. Later, as images are received, person detection is run on the image using a bounding box or a segmentation approach, giving a mask for each person. Then the edges of the masks are found and the corresponding depths are looked up, forming a parametric curve of depth. The minima of this curve are expected to correspond to the closest points of the footprint with respect to the camera. Depending on the posture of the person, and hence the corresponding mask, these points can correspond to two individual feet, or just one foot, and at times it may also contain some errors due to the segmentation process. Then the corresponding x and y are calculated, and the final positions (x, y, z) are returned.

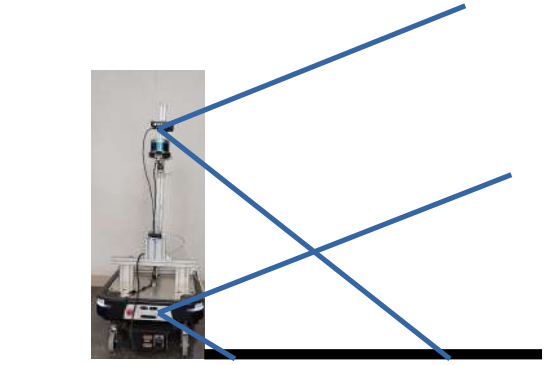


Fig. 3. Field of view of a top and a bottom camera on our experimental platform.

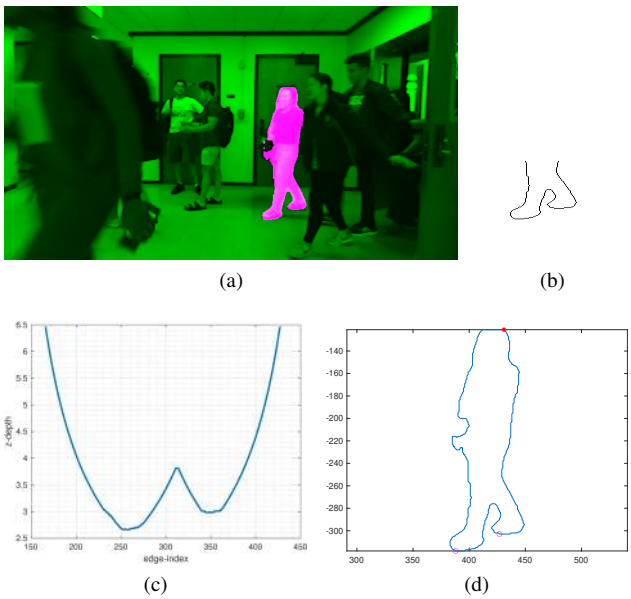


Fig. 4. (a) The original image, with the segmentation masks overlayed. (b) The corresponding edge with depth lookup (c) The depth of the edge as a function of the edge, the minima seen correspond to the individual feet (d) The minima of (c) marked on the borders of the individual mask.

Algorithm 1 Pose Estimate From Monocular Camera

procedure CALCULATEDEPTHIMAGE($\mathbf{I}$)
  $\mathbf{I_z} \leftarrow \mathbf{0}$
  for each $(u, v) \in \mathbf{I}$ **do**
    $\mathbf{I_z}(u, v) \leftarrow fh_c/(v - c_y)$
  end for
  return $\mathbf{I_z}$
end procedure
procedure CALCULATEPOSITION($\mathbf{I}$)
  $\mathbf{I_M} \leftarrow$ DetectPerson(BoundingBox|Segmentation)
  for each $\mathbf{I_m} \in \mathbf{I_M}$ **do**
    $\mathbf{I_e} \leftarrow$ FindEdges($\mathbf{I_m}$)
    $\mathbf{I_d} \leftarrow$ LookupDepth($\mathbf{I_e}, \mathbf{I_z}$)
    $f(z) \leftarrow$ FormCurve($\mathbf{I_d}$)
    $[u, v, z] \leftarrow$ FindArgMin($f(z)$)
    $y \leftarrow h_c$
    $x \leftarrow (z/f)(u - c_x)$
  end for
  return $[x, y, z]$ ▷ List containing all detections
end procedure

B. Lidar and Camera Fusion

1) Point Cloud ROI Extraction: The process starts with the detection of people in an RGB image using a pre-trained YOLOv7-tiny or YOLOVv7-seg [29], which was trained on the MS COCO [31] dataset. YOLO returns the pixel coordinates of the 2D bounding boxes or segmentation masks corresponding to people. Using the transformation matrix ${}^{C}_{L}T$, which represents the optical frame of reference of the camera in the frame of reference of the lidar, and the intrinsic matrix of the camera K, it is possible to match the bounding boxes/masks of the RGB images and the point clouds of the lidar.

Let ${}^{C}P = \{(x_i, y_i, z_i)\}$ represent the 3D point cloud of the lidar after it is transformed to the reference frame of the camera. The point cloud is filtered, excluding the points outside the field of view of the camera, giving ${}^{C}P'$. These points can be transformed to the image plane using the camera matrix K and sequentially applying equation (2), giving a corresponding set of points $\{(u_i, v_i)\}$. By comparing these corresponding image coordinates with the bounding boxes or segmentation masks output by YOLOv7, the regions of interest of the point cloud, which lie inside the bounding boxes or the masks, are identified. Assuming that YOLO has reported N bounding boxes/masks, each 3D point cloud is represented by P_{box_j}, with each containing n_j many points $(j = 1, ...N)$.

2) Candidate Generation: The set of N 3D point clouds extracted in II-B.1 do not necessarily contain points only corresponding to the humans detected by YOLO. The clouds may include the background and in some cases the foreground of the person (Fig. 5), particularly when using bounding boxes instead of segmentation masks. Therefore, it is necessary to analyze each cloud, P_{box_i}, establish clusters and find the ones corresponding to humans. A solution to this problem could be the application of a density-based clustering algorithm, such as DBSCAN [32], to each point cloud. However, DBSCAN has a set of parameters which would not allow the algorithm to scale well for clouds with sparser or far points, or allow separation of a person standing just a few centimeters away from a wall. In order to solve these problems, we propose complementing DBSCAN with a pre-processing step named z-filtering. This pre-processing first assigns the 3D points into different layers along the $z-$axis, with respect to the camera frame. Then it aggregates consecutive layers into single clusters based on their point density and a previously defined percentage threshold DENSITY_RATIO which is calculated through the average point density across all layers with at least one point. After pre-processing, DBSCAN is applied to each of the previously formed clusters along the camera's x and z coordinates. The y coordinate is not considered for invariance to the person's distance from the sensor due to the LiDAR's vertical angular resolution, which causes the LiDAR's beams to become more vertically spaced as the distance to sensor increases. If, due to the point cloud sparseness, DBSCAN is not able to form any clusters, then the generated candidate is the cluster established by the preprocessing step. When using segmentation masks, due to their more precise cropping of the human body, it is not necessary to apply DBSCAN, only z-filtering.

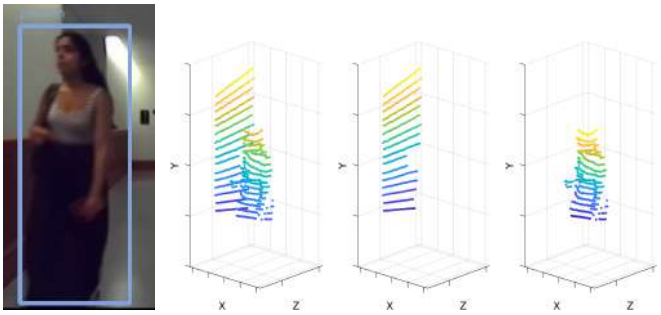


Fig. 5. Depiction of process described II-B.2. a) YOLOv7 detection; b) Resulting point cloud after pre-processing; c) and d) The two clusters extracted from Fig. 5b by DBSCAN

3) Lidar and Monocular Estimate Fusion: The cluster fusion process divides the working space into three regions, depending on the distance from the camera. The closest region corresponds to the area where the ground is not visible by the camera, and hence since the feet are not visible the monocular estimation of the pose is not possible. For this region, $z \leq h_c \tan\theta/2$. If a person's mask fills the lower end of the image, it is assumed to be in this close range. Hence, the lidar clusters obtained by ROI fusion are used. Secondary clusters are removed filtering directly by distance, i.e. requiring $z \leq h_c \tan\theta/2$. In the second region, the person's feet are visible in the camera $z \geq h_c \tan\theta/2$, and the person forms a meaningful cluster in the lidar scan. In this case after applying ROI fusion, the clusters are selected comparing the distances between the lidar estimates and the monocular estimates. In the third region, the lidar is reporting only random points, which is particularly the case with a lidar close to the ground. Hence, only the monocular estimate is used to represent the person.

C. Baseline Approach - 3D Feature Based Classification

In a recently published work [21], we have showed that after using the ROI provided by the image based person detector (YOLO) to extract the corresponding lidar point cloud, the ROI contained readings not only from the detected person but also from the the background and the foreground of the persons, at times parts of other persons, at times walls or other items in the environment. In order to solve this problem, various classifiers were tested on the 3D point cloud, and a new feature vector based on 3D Modified Fisher Vector (3DmFV) [33] was proposed. The new feature vector, called 3DmFV with Plane Inlier Percentage (3DmFV+PIP), included planarity as well, and showed considerably improved performance.

III. Validation and Experimental Work

A. Localization Performance

The robot was placed in a long corridor, marking the ground first every 0.5 m, then every 1.0 m. Then a test subject stood still at each mark, recording the corresponding images from a top and a bottom camera, at 1.25 m and 0.3 m respectively and a 3D lidar. The data then was used to measure both monocular localization performance and vision lidar fusion performance, presenting the results in Fig. 6. The data shows that monocular performance of the bottom camera was considerably better, localizing the subject mostly with an error less than 0.5 m up to 14 m. Afterwards, the error starts growing, reaching 6.0 m at 20 m. The error of the top camera however reaches 2.0 m at 6.0 m, and after 11.0 m, it exceeds 4.0 m. This behaviour is caused by the low quality person detection by YOLO for the top camera, mainly caused by the camera's incapability to adjust exposure properly (Figs. 6c, 6d). However, fusion can be seen to help keep the localization error close to zero up to 10 m.

The second test was repeated using a subset of the JRDB dataset, this time manually extracting the ground truth position by inspecting the lidar data carefully. The JRDB dataset includes a wide array of densely crowded scenarios, thus allowing for a more challenging evaluation of the proposed approach. When using the JRDB dataset, data from a camera at 1.1 m height, a 3D lidar and a 2D lidar at approximately 1.3 m and 0.3 m was used. Fusion capabilities were tested separately for both the 2D and the 3D lidar. Fig. 7 shows the histogram of the error calculated over two distance regions of fusion for the tested dataset, with the average errors presented in table I. In close range the error can be seen to be small with both lidars, with a mean value of 0.14 m. In mid range, although large errors are randomly seen, the error is concentrated in the lower 0.3 m bins, with an average of 0.15 m for the 2D, and 0.3 m for the 3D, increased mainly by the several samples with a big error. For the 2D lidar the end of mid-region was chosen as 6.0 m, because after that point persons were only randomly visible in the lidar data. However, for 3D lidar it was possible to observe meaningful data points up to 12 m, a range containing all the detected persons.

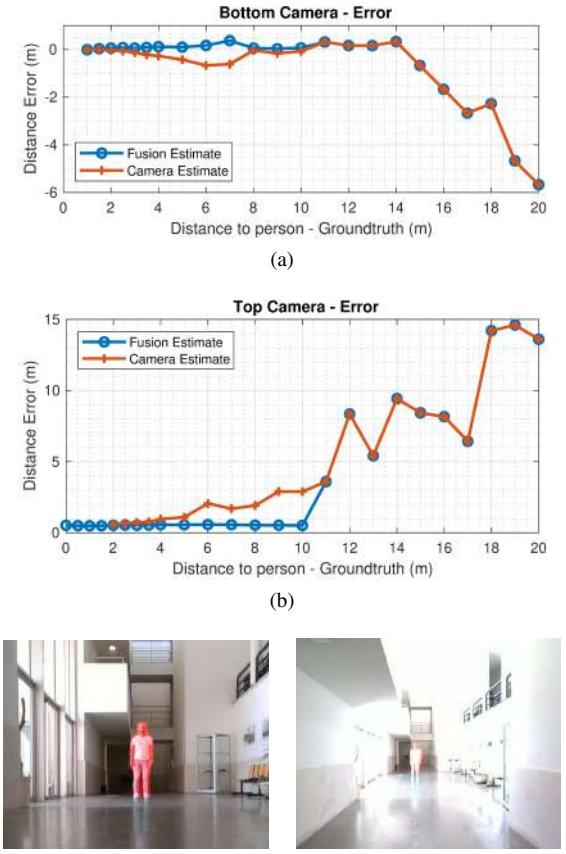


Fig. 6. (a, b) Monocular and fusion localization errors obtained using robot cameras at different heights in a corridor. The first samples do not contain monocular estimates since the subject was too close to the cameras. (c, d) Segmentation results from the two cameras when the person is 11 m away from the cameras. Due to illumination problems, the segmentation of the top camera can be seen not to include half of the legs, causing localization errors.

TABLE I
Error in position estimate

	Close Region	Mid-Region	Far Region
Monocular & 2D lidar	0.14	0.15	1.15
Monocular & 3D lidar	0.16	0.28	-

B. Classification Performance

The JRDB dataset contains many examples of point clouds that contain the detected person as well as other people and fore/background items, causing classification errors of the 3D point cloud. Therefore, a subset of the data set was used to measure the improvements to the 3D classification obtained through fusion of monocular localization. As a baseline the recently proposed [21] and in section II-C briefly described 3DmFV+PIP feature vector was used in a Support Vector Machine (SVM) classifier. Monocular localization was also fused with 2D lidar data for a different classification run, and all the results are summarized in Table II. On the tested subset, fusion of monocular localization with lidar data can

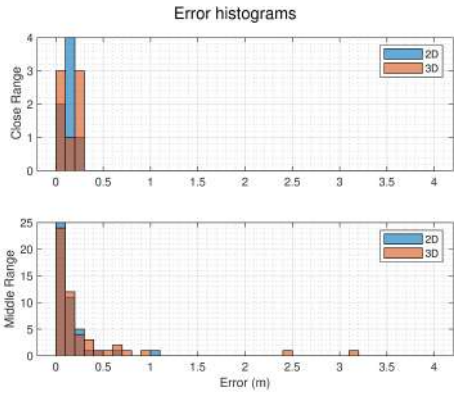


Fig. 7. Error histogram for monocular and lidar fusion.

be seen to improve the localization performance of the people considerably, with even 2D lidar's performance surpassing 3DmFV+PIP based classification approach. The state of the art 3DmFV+PIP approach was able to find the right point cloud cluster of the detected person in 72% of the frames, whereas the 2D monocular fusion and 3D monocular fusion were able to achieve rates of 83% and 86%.

TABLE II
PERFORMANCE OF DIFFERENT APPROACHES

	TP	FP	TPR	FPR
3DmFV + PIP	42	16	0.72	0.28
Monocular & 3D lidar	50	8	0.86	0.14
Monocular & 2D lidar	48	10	0.83	0.17

IV. CONCLUSIONS

In this work we have shown that monocular localization can help improve performance of lidar camera fusion, reducing the outliers and hence improving the accuracy considerably. We have also shown that, peformance of monocular localization depends heavily on segmentation performance in the visual domain, particularly light performance. As future work, we are planning to integrate the approach proposed in this paper in a social navigation pipeline, such as [34], [35] runnning on a robot.

REFERENCES

[1] R. Kirby, R. Simmons, and J. Forlizzi, "COMPANION: A Constraint-Optimizing Method for Person-Acceptable Navigation," in *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, Sept. 2009, pp. 607–612, iSSN: 1944-9437.

[2] F. Yang and C. Peters, "Social-aware navigation in crowds with static and dynamic groups," in *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, Sept. 2019, pp. 1–4, iSSN: 2474-0489.

[3] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A Human Aware Mobile Robot Motion Planner," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 874–883, Oct. 2007, conference Name: IEEE Transactions on Robotics.

[4] J. Thomas and R. Vaughan, "Right of Way, Assertiveness and Social Recognition in Human-Robot Doorway Interaction," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 333–339, iSSN: 2153-0866.

[5] Y. Che, A. M. Okamura, and D. Sadigh, "Efficient and Trustworthy Social Navigation via Explicit and Implicit Robot–Human Communication," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 692–707, June 2020, conference Name: IEEE Transactions on Robotics.

[6] P. Teja Singamaneni, A. Favier, and R. Alami, "Human-Aware Navigation Planner for Diverse Human-Robot Interaction Contexts," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2021, pp. 5817–5824, iSSN: 2153-0866.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 779–788, iSSN: 1063-6919.

[8] M. Gruosso, N. Capece, and U. Erra, "Human segmentation in surveillance video with deep learning," *Multimedia Tools and Applications*, vol. 80, no. 1, pp. 1175–1199, Jan. 2021.

[9] B. Lewandowski, J. Liebner, T. Wengefeld, S. Müller, and H.-M. Gross, "Fast and Robust 3D Person Detector and Posture Estimator for Mobile Robotic Applications," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 4869–4875, iSSN: 2577-087X.

[10] K. Kidono, T. Miyasaka, A. Watanabe, T. Naito, and J. Miura, "Pedestrian recognition using high-definition LIDAR," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, June 2011, pp. 405–410, iSSN: 1931-0587.

[11] K. O. Arras, O. M. Mozos, and W. Burgard, "Using Boosted Features for the Detection of People in 2D Range Data," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 3402–3407, iSSN: 1050-4729.

[12] C. Premebida, O. Ludwig, and U. Nunes, "Exploiting LIDAR-based features on pedestrian detection in urban scenarios," in *2009 12th International IEEE Conference on Intelligent Transportation Systems*, Oct. 2009, pp. 1–6, iSSN: 2153-0017.

[13] A. F. Foka and P. E. Trahanias, "Probabilistic Autonomous Robot Navigation in Dynamic Environments with Human Motion Prediction," *International Journal of Social Robotics*, vol. 2, no. 1, pp. 79–94, Mar. 2010.

[14] A. Hacinecipoglu, E. Konukseven, and A. B. Koku, "Pose Invariant People Detection in Point Clouds for Mobile robots," *International Journal of Mechanical Engineering and Robotics Research*, vol. 9, no. 5, 2020.

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017, pp. 652–660.

[16] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast Encoders for Object Detection from Point Clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[17] Y. Hu, Z. Ding, R. Ge, W. Shao, L. Huang, K. Li, and Q. Liu, "AFDetV2: Rethinking the Necessity of the Second Stage for Object Detection from Point Clouds," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, 2022, pp. 969–979.

[18] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[19] M. E. Kundegorski and T. P. Breckon, "A photogrammetric approach for real-time 3D localization and tracking of pedestrians in monocular infrared imagery," in *Optics and Photonics for Counterterrorism, Crime Fighting, and Defence X; and Optical Materials and Biomaterials in Security and Defence Systems Technology XI*, vol. 9253. SPIE, Oct. 2014, pp. 139–154.

[20] Y. Niu, Z. Xu, E. Xu, G. Li, Y. Huo, and W. Sun, "Monocular Pedestrian 3D Localization for Social Distance Monitoring," *Sensors*, vol. 21, no. 17, 2021.

[21] C. A. Silva, S. Dogru, and L. Marques, "Camera and LiDAR Fusion for Robust 3D Person Detection in Indoor Environments," in *2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. Tomar: IEEE, Apr. 2023.

[22] H. Bozorgi, X. T. Truong, H. M. La, and T. D. Ngo, "2D Laser and 3D Camera Data Integration and Filtering for Human Trajectory Tracking," in *2021 IEEE/SICE International Symposium on System Integration (SII)*, Jan. 2021, pp. 634–639, iSSN: 2474-2325.

[23] F. Bu, T. Le, X. Du, R. Vasudevan, and M. Johnson-Roberson, "Pedestrian Planar LiDAR Pose (PPLP) Network for Oriented Pedestrian Detection Based on Planar LiDAR and Monocular Images," *IEEE*

Robotics and Automation Letters, vol. 5, no. 2, pp. 1626–1633, Apr. 2020, conference Name: IEEE Robotics and Automation Letters.

[24] V. Tolani, S. Bansal, A. Faust, and C. Tomlin, "Visual Navigation Among Humans With Optimal Control as a Supervisor," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2288–2295, Apr. 2021, conference Name: IEEE Robotics and Automation Letters.

[25] L. Spinello and R. Siegwart, "Human detection using multimodal and multidimensional features," in *2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 3264–3269, iSSN: 1050-4729.

[26] C. Premebida, J. Carreira, J. Batista, and U. Nunes, "Pedestrian detection combining RGB and dense LIDAR data," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept. 2014, pp. 4112–4117, iSSN: 2153-0866.

[27] A. González, G. Villalonga, J. Xu, D. Vázquez, J. Amores, and A. M. López, "Multiview random forest of local experts combining RGB and LIDAR data for pedestrian detection," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 356–361, iSSN: 1931-0587.

[28] J. Schlosser, C. K. Chow, and Z. Kira, "Fusing LIDAR and images for pedestrian detection using convolutional neural networks," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2198–2205.

[29] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[30] R. Martin-Martin, M. Patel, H. Rezatofighi, A. Shenoi, J. Gwak, E. Frankel, A. Sadeghian, and S. Savarese, "JRDB: A Dataset and Benchmark of Egocentric Robot Visual Perception of Humans in Built Environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[31] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[32] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, vol. 96. AAAI Press, 1996, pp. 226–231.

[33] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "3DmFV: Three-Dimensional Point Cloud Classification in Real-Time using Convolutional Neural Networks," *IEEE Robotics and Automation Letters*, 2018.

[34] M. Kollmitz, K. Hsiao, J. Gaa, and W. Burgard, "Time Dependent Planning on a Layered Social Cost Map for Human-Aware Robot Navigation," in *2015 European Conference on Mobile Robots (ECMR)*. IEEE, 2015, pp. 1–6.

[35] C. A. Silva, S. Dogru, and L. Marques, "Mobile Robot Navigation in Dynamic Environments Taking into Account Obstacle Motion in Costmap Construction," in *ROBOT2022: Fifth Iberian Robotics Conference: Advances in Robotics, Volume 1*. Springer, 2022, pp. 235–246.

Towards Data-Driven Discovery of Governing Swarm Robots Flocking Rules

Belkacem Khaldi[1], Erhan Ege Keyvan[2], Mehmet Şahin[2], Ali Emre Turgut[2], and Erol Şahin[2]

***Abstract*— Extracting local interaction rules that govern the dynamics of a swarm is a central challenge in many swarm robotics application domains. Reverse engineer of such dynamics might be highly beneficial in preventing the serious design handcrafting errors that swarm robotics engineers may implicitly make. Advances in data-driven based systems identification techniques, called SINDy, are currently enabling the tractable identification of the equations governing the dynamics of many systems. However, they have not yet to be applied in swarm robotics systems. In this work, we aim to combine sparsity-promoting techniques with nonlinear swarm dynamical systems to develop a data-driven system identification model capable of discovering governing swarm flocking interaction rules from swarm measurement data. We particularly build and compare two SINDy flocking models: Flock-SINDy-STLSQ and Flock-SINDy-SR3. our findings suggest that the Flock-SINDy-SR3 discover better the underlying flocking dynamics rules than the Flock-SINDy-STLSQ and is expected to be further used as a controller implemented on real drones.**

***Index Terms*— System Identification, SINDy, Swarm robotics, Flocking behaviour, Reverse engineering.**

I. INTRODUCTION

Swarm robotics systems frequently employ local interaction rules at microscopic level to fulfill required mission collectively at the macroscopic level [1]. The common method for handcrafting these local interaction rules is to model and implement them at the microscopic swarm level. This, however, has a substantial effect on the swarm's macroscopic level since any alterations may severely influence the swarm's overall intended mission.

One question that comes to mind for overcoming such a challenge is can we reverse engineer the local rules governing the interactions between the swarm, so that we may avoid substantial design handcrafting problems. This might be accomplished if the fundamental procedures of reverse engineering—information extraction, modeling, and testing—are followed [2].

Advances in data-driven techniques that includes both classical machine learning and deep learning approaches, specifically those being applied for sparse identification of non-linear dynamics (SINDy) [2-6], are currently enabling the tractable identification of many equations governing various complex systems. They have recently been used to identify models ranging from predictive control systems, canonical physical systems, aerodynamic systems, chemical processes to epidemiological systems, but not to our knowledge in swarm robotics. They offer appealing qualities that make them worth considering in the context of swarm robotics.

Recent breakthroughs in data-driven techniques to system identification in general can inductively-directly discover dynamic models from data. Most successful methods in this regard include artificial neural networks [3], [4], evolutionary algorithms [5], [6] and nonlinear regression [7], [8], [9], [10]. Artificial neural networks have strong system dynamics identification abilities. However, as generalizability and interpretability are the main concerns in system dynamics, it is crucial to find sparse models that, in contrast to neural networks, require the fewest terms to express their dynamics. Evolutionary algorithms, on the other hand, are powerful bio-inspired alternative methods that have successfully been applied to system identification. They are, however, computationally demanding and hence unsuitable for real-time tracking. System identification also benefits greatly from the use of conventional regression-based model identification techniques such as the koopman operator [8], [10] and dynamic mode decomposition (DMD) [7], [9]. However, they are unable to capture structural modifications or underlying nonlinear dynamics.

The current work primarily concerned towards the adoption and adaptation of sparse-based data-driven identification methods, with the goal of following the aforementioned reverse engineering steps to discover models governing the swarm-interaction rules leading to the macroscopic swarm robotics collective behavior. This is motivated by our belief that this is crucial for designing more accurate control models for swarm robotics systems, which contributes to the improvement of swarm robotics technology. On the other hand, we believe this is an opening up of a new research direction for future perspectives in the swarm robotics field.

To this end, the current work showcases the feasibility of system dynamics discovery in a swarm drones performing a flocking behaviour that was previously successfully being implemented on a mobile swarm robotics system [11] (See Subsection II-B). The work presents in subsection III-A two data-driven discovery flocking models following the conventional SINDy method reviewed in subsection II-A. These two built models are assessed and compared further in subsection III-B to see their capabilities in predicting the swarm flocking dynamics in a forward simulation manner. Furthermore, the accurate flock SINDy model is assessed further in subsection III-C in terms on how it is expected

[1]B. Khaldi is with the LabRI-SBA Laboratory, Ecole Superieure en Informatique 8 Mai 1945, Sidi Bel Abess, Algeria b.khaldi@esi-sba.dz

[2] E.E. Keyvan, M. Şahin, A.E.Turgut, and E. Şahin are with the Center for Robotics and Artificial Intelligence (ROMER), Middle East Technical University, Ankara. [ekeyvan, mesahine, aturgut, erol]@metu.edu.tr

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

to perform if used as a controller implemented in each individual drone.

II. PRELIMINARIES

This section introduces key background material and concepts for the identification framework of flocking drones behavior. We begin by describing the SINDy with control technique, which served as the foundation for our work. Then we review the flocking behaviour re-implemented on the ROS based Kobot simulator.

A. Sparse Identification of Nonlinear DYnamics with Control inputs

Consider a nonlinear discrete-time dynamical system of the form:

$$\mathbf{x}_{t+1} = \mathbf{g}(\mathbf{x}_t, \mathbf{u}_t) \tag{1}$$

where $\mathbf{x}_t \in \mathbb{R}^n$ is the state at time t, $\mathbf{u}_t \in \mathbb{R}^q$ is the control input at time t, and $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^n$ describes the system dynamics. It is important to note that the system dimension corresponds to the variables in its dynamics. In our case, the function captures the flocking dynamics, including the states of drones and control inputs.

We review the SINDy with control, which is a technique that uses measurement data to find a sparse nonlinear dynamical system. It employs sparse regression to discover a few active terms necessary to approximate the function g. To select the optimal model, the approach assesses a library, $\Theta(x_t, u_t)$, of possible linear and nonlinear model terms in the state, x_t and control inputs u_t. This library is computed by first stacking $m-1$ time snapshots of the state x_t and the input signal u_t into two matrices: $X = \begin{bmatrix} x_1 & x_2 & \cdots & x_{m-1} \end{bmatrix}^T$ and $U = \begin{bmatrix} u_1 & u_2 & \cdots & u_{m-1} \end{bmatrix}^T$.

The associated temporal history of the state variable derivatives x_t from the data X are often estimated numerically or gotten by shifting the X by one step time and organized in a matrix, represented by $X' = \begin{bmatrix} x_2 & x_3 & \cdots & x_m \end{bmatrix}^T$. Then, a collection of potential nonlinear functions that best describe the data and which may contain constant, polynomial, trigonometric, or any customized built-in terms might be constructed and assessed as follows[12], [13]:

$$\Theta(X, U) = \begin{bmatrix} 1 & X & U & (X \otimes X) & (X \otimes U) & \cdots \end{bmatrix},$$

where $X \otimes U$ is the vector representing all product combinations of the components in X and U. Note that the optimal method for picking the right library is to begin with a straightforward alternative, such as low-order polynomials, and progressively increase the complexity of the library until sparse and precise models are produced.

The system in Eq.1 may then be written in terms of these data matrices as:

$$\mathbf{X}' = \Theta(X, U)\mathbf{\Xi} \tag{2}$$

Where $\mathbf{\Xi} = \begin{bmatrix} \xi_1 & \xi_2 & \cdots & \xi_m \end{bmatrix}$ is mostly a sparse matrix containing the coefficients of the most active terms in the generated library functions, Θ,which produce a good model fit and might be identified by employing one of the sparse regression techniques. The goal of such strategies is to reduce the residual error between the real data and the model. The sequential thresholded least-squares (STLSQ) method is used in the original SINDy approach [12], and several other variants, including the sparse relaxed regularized regression optimizer (SR3) [14], have been used to improve the SINDy's performance in identifying complex systems dynamics in terms of computational efficiency, higher accuracy, quicker convergence rates, and greater flexibility.

B. Flocking Behaviour

Our study draws inspiration from the flocking behavior exhibited by mobile robot swarms [11] and extends it to a swarm of drones using the ROS based kobot simulator. To achieve flocking behavior in the swarm, we use a weighted vector sum of heading alignment and proximal control vectors, represented as:

$$f = \frac{\alpha h + \beta p}{||\alpha h + \beta p||} \tag{3}$$

Here, h is the heading alignment vector with weight α, p is the proximal control vector, encoding the repulsion and attraction rules, with weight β, and f is the desired flocking vector.The resultant vector's Euclidean norm is represented by $||.||$. It is worth noting that the vector h enables a drone to align with the average heading of its neighbors via the onboard drone Vector Heading Sensor to obtain the neighbors' current headings. Whereas, the proximal vector utilizes data collected by the time-of-flight Range and Bearing sensor system [15] to accomplish two objectives: preventing collisions with both other robots and obstacles, as well as maintaining cohesion between the swarm members. Advanced mathematical details on how these vectors are computed can be found in [11].

III. Discovery of Governing Swarm Drones Flocking Dynamics

A. SINDy Flocking Models Learning Results

We focus on identifying the swarm flocking robotics dynamics that are induced by the local interactions rules given in Eq.3. The overall framework pursued in this study is schematized in Figure.1. To this end, we performed a number of ROS-based simulations of a swarm of $n = 5$ Tello drones performing the flocking behaviour using the Kobot simulator during a fixed time steps duration. The weighting coefficients used in this setup are $\alpha = 1$ and $\beta = 1.5$. Note that the choice of weights were carefully set to slightly prioritize the proximal control over the alignment control, while ensuring that the balance between them in the flocking behavior remains unaffected. At each simulation, the drones were randomly dispatched in a free-obstacles arena bounded by four walls. We collected at each simulation time step, t, the flocking ($f_t^i = \begin{bmatrix} f_t^i(x) & f_t^i(y) & f_t^i(z) \end{bmatrix}$), the proximal ($p_t^i = \begin{bmatrix} p_t^i(x) & p_t^i(y) & p_t^i(z) \end{bmatrix}$), and the alignment heading ($h_t^i = \begin{bmatrix} h_t^i(x) & h_t^i(y) & h_t^i(z) \end{bmatrix}$) 3D vectors for each drone. These measurements are then arranged and stacked

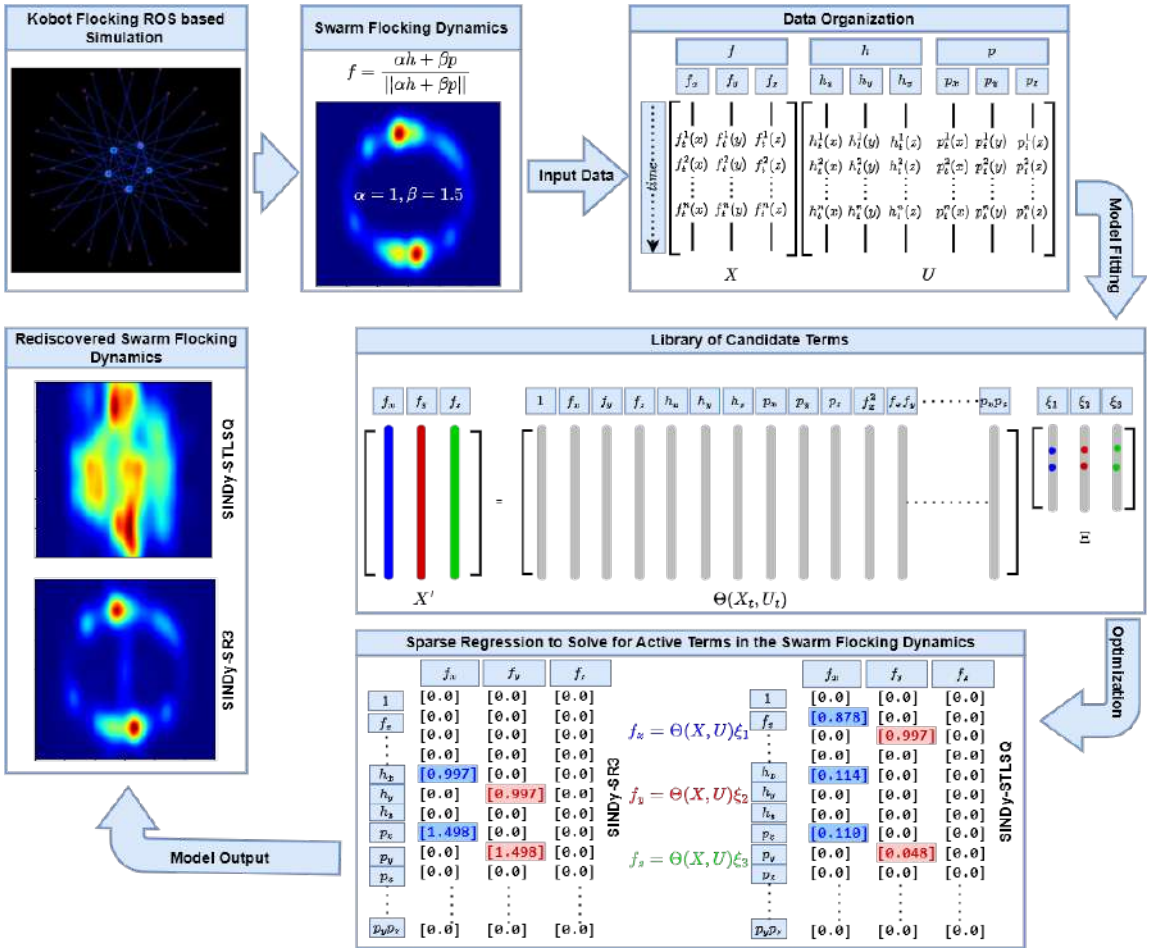


Fig. 1: Schematic of the flocking-SINDy discovery framework, demonstrated on the swarm dynamics governed by a flocking interaction rules. Data are collected from the ROS based Kobot simulator of a swarm Tello drones system performing the flocking behavior in [11], including a time history of the swarm states X, its derivative X', and the Control Inputs U.

into two big matrices of shapes $(n * TimeSteps, 3)$ and $(n * TimeSteps, 6)$ as follows:

$$X = \overrightarrow{\text{state}} \begin{bmatrix} | & | & | \\ f_t^1(x) & f_t^1(y) & f_t^1(z) \\ f_t^2(x) & f_t^2(y) & f_t^2(z) \\ \vdots & \vdots & \vdots \\ f_t^n(x) & f_t^n(y) & f_t^n(z) \\ | & | & | \end{bmatrix} \downarrow \text{time steps} * n \tag{4}$$

and

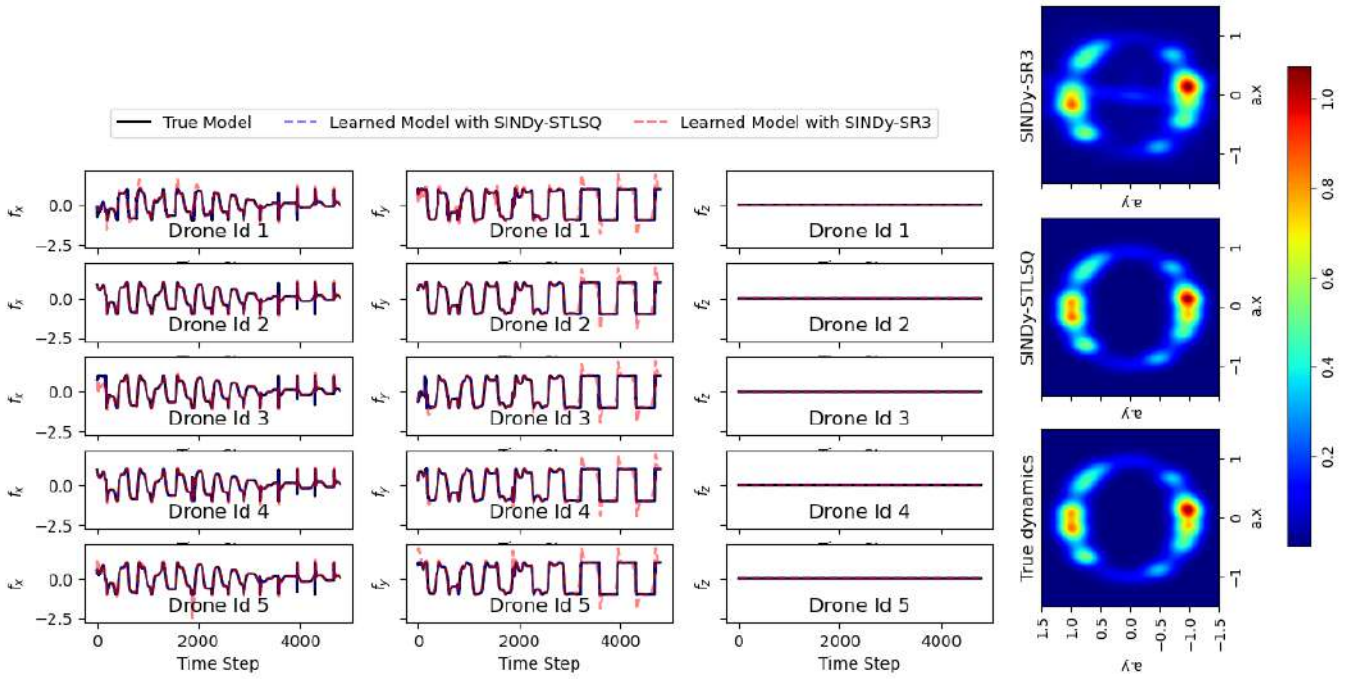
$$U = \overrightarrow{\text{Control Inputs}} \begin{bmatrix} | & | & | & | & | & | \\ h_t^1(x) & h_t^1(y) & h_t^1(z) & p_t^1(x) & p_t^1(y) & p_t^1(z) \\ h_t^2(x) & h_t^2(y) & h_t^2(z) & p_t^2(x) & p_t^2(y) & {}^2_t(z) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_t^n(x) & h_t^n(y) & h_t^n(z) & p_t^n(x) & p_t^n(y) & p_t^n(z) \\ | & | & | & | & | & | \end{bmatrix} \downarrow \text{time steps} * n \tag{5}$$

Therefore, the state of our system is $X = [f_x, f_y, f_z]$, and we assume that we can control it by the input control variables $U = [h_x, h_y, h_z, p_x, p_y, p_z]$.

First, we used 80% of the data for the training dataset and the remaining for the testing dataset. We next run the SINDy with control algorithm to identify a sparse nonlinear model. First, we build the library of candidate functions Θ. Here, we use polynomials up to second order. We set the sparsification hyperparameter to $\lambda = 0.01$ and run the SINDy algorithm using the STLSQ and the SR3 optimizer approaches.

In Figure 2, the results of the SINDy flocking dynamics identification during the learning stage are shown. We use SINDy to identify the underlying flocking rules and compare the results of the true model with the STLSQ and the SR3 optimizers. On the left of Figure 2 (in panel (a)), we show the ground truth of the flocking dynamics of each drone (see black lines) and the identification training for SINDy-STLSQ (dashed blue lines) and SINDy-SR3 (dashed red lines). The control inputs are the same for both the SINDy models.

We see that the SINDy-STLSQ model perfectly predicts the true flocking dynamics as it perfectly captures the evolution of all flocking vector components: f_x, f_y, and f_z. Similarly, the SINDy-SR3 model is almost identical to the



(a) True model vs SINDy built models for each drone

(b) True model vs SINDy built models plotted as Kernel density estimations for the entire swarm

Fig. 2: SINDy flocking dynamics identification results during the learning stage.

true flocking dynamics, and slightly diverges during few moments of the time steps training period.

The aforementioned results can be confirmed while better visualizing the flocking dynamics of the entire swarm drones using the kernel density estimation plots of the true model, and the SINDy built models (see panel(b) of Figure 2).

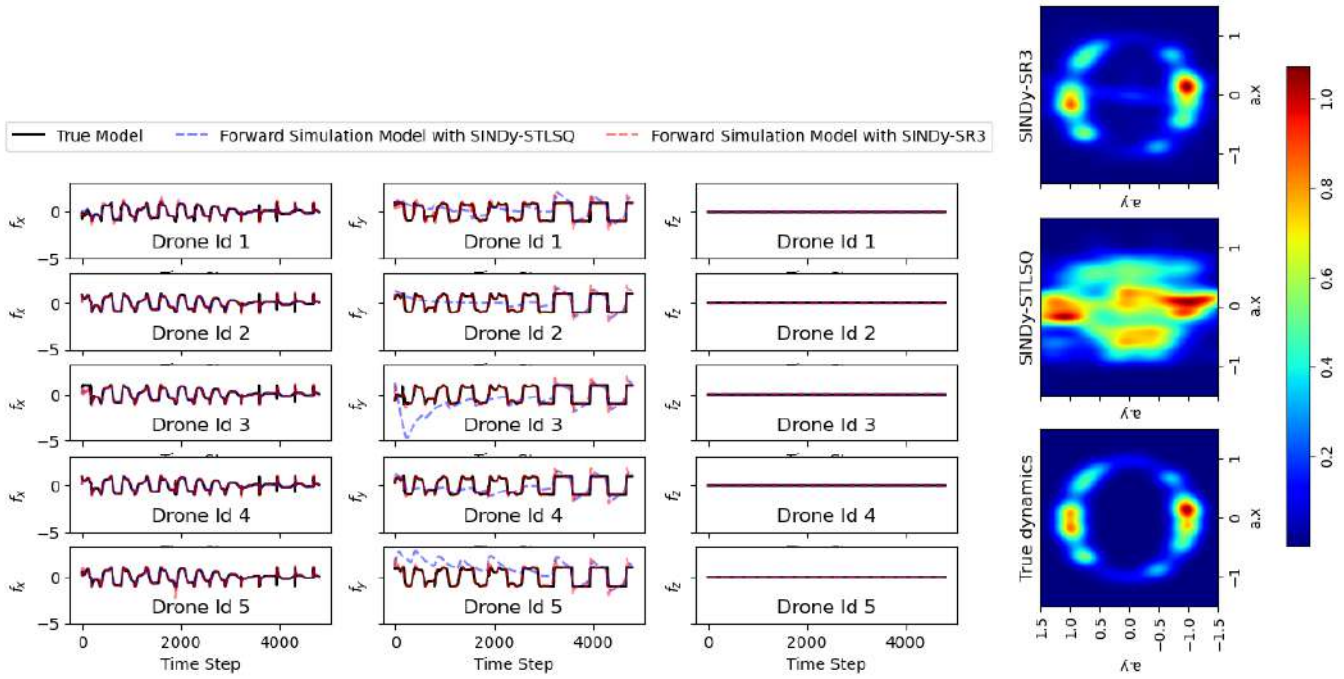
It is important to highlight that the kobot simulator is specifically designed for two-dimensional (2D) motion and interactions of swarm drones. As a result, the z-coordinate remains constant at 0 throughout the experiments. This design choice aligns with the main objective of our study, which is to showcase the effectiveness of the SINDy algorithms in discovering swarm-flocking rules within the horizontal plane.

B. SINDy Flocking Models Prediction Results

In this subsection, we use the flocking built SINDy models to assess the swarm flocking dynamics identification performances. The main challenge behind this is to see how well the models accurately predict and identify the governing flocking dynamics when performing forward simulations in the testing dataset. Similarly to the previous subsection, we compare the system identification results of the SINDy flocking models with the ground truth flocking dynamics. The obtained results are shown in Figure 3 for both individual drones and the entire swarm.

This time, the flocking model with the SINDy-SR3 sustain its performance learned previously and preforms better than the flocking model with the SINDy-STLSQ and this during all the testing time step window. As can be seen in panel (a) of Figure 3, the flocking model with the SINDy-SR3 almost captures the flocking dynamics for nearly all the x, y, and z components of the drones flocking vector, f (See dashed red lines). However, it slightly diverges during few moments of the time steps testing window. This divergence is tolerant as the swarm flocking dynamics is almost the same while compared to the true swarm flocking dynamics plotted as kernel density maps (See panel (b) of Figure 3).

On the other hand, the flocking model with the SINDy-STLSQ, accurately predicts the drones flocking dynamics of both x and z components of the flocking vector f during the entire testing time steps period. However, it fails from a long period from the start of the forward simulation window to discover the flocking dynamics of component y. But begin slightly to converging to the true y dynamics as we forward further. This leads to a fail in capturing the entire flocking swarm dynamics during the entire testing time steps window while compared to the true swarm flocking model (See 3.(b)). We may therefore presume that the SINDy-STLSQ model perform poorly in comparison to the SINDy-SR3, even-thought it perfectly perform well during the learning stage. This is due to the fact that the underlying flocking governing rules discovered by SINDy-STLSQ (See Table I) is different to the true one I). Whereas the forward simulation results with the SINDy-SR3 model performs better as it perfectly discovered the exact underlying flocking governing rules.



(a) True model vs SINDy built models for each drone

(b) True model vs SINDy models plotted as Kernel density estimations for the entire swarm

Fig. 3: SINDy flocking dynamics identification results during the testing window.

TABLE I: True vs Discovered Individual Flocking Dynamics Equations

Model	Flocking Dynamics Equations
True Model	$\begin{cases} f_x = 1 * h_x + 1.5 * p_x \\ f_y = 1 * h_y + 1.5 * p_y \\ f_z = 0 \end{cases}$
SINDy-STLSQ	$\begin{cases} f_x = 0.878 * f_x + 0.114 * h_x + 0.110 * p_x \\ f_y = 0.997 * f_y + 0 * h_y + 0.048 * p_y \\ f_z = 0 \end{cases}$
SINDy-SR3	$\begin{cases} f_x = 0.997 * h_x + 1.498 * p_x \\ f_y = 0.997 * h_y + 1.498 * p_y \\ f_z = 0 \end{cases}$

C. Swarm Flocking Revers-Engineering Evaluation

In this subsection, we aim to assess further if the obtained SINDy-SR3 flock model could be used in reverse engineering purpose. We particularly seek to see how well the forward simulation results of the obtained flocking model will perform if used as a controller implemented in each individual drone. To this end, we just used the obtained forward simulation SINDy results in estimating each drone's linear and angular velocities, which will allow us to estimate later each drone' positions over time $(x_i, y_i, z_i, \theta_i)$.

We use the same rules used in the mobile robot flocking swarms work of [11] and which is re-implemented in the Kobot based ROS simulator. The forward and angular velocities, v and ω of each drone is updated as follows:

$$v = \begin{cases} (f.h_c)^{\lambda} * v_{max}, & \text{if } f.h_c \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

and

$$\omega = (\angle h_c - \angle f) K_p \tag{7}$$

where in Eq.6, h_c is the current drone heading vector and λ is to modulate the forward velocity. More details on why the modulation has to be applied can be found in [11]. v_{max} is the maximum linear velocity allowed for the drones. While the K_p term in Eq.7 is for proportionally control ω using the deviation of the drone flocking angle from its current direction. For the estimation of drones positions given their estimated forward and angular velocities, we used the commonly used Kalman filtering estimation method.

Now, to study the flocking behaviour with the obtained SINDy-SR3 flocking model, we adopt a modified version of the angular momentum metric [16] that suggests the rotational motion of the swarm and is given by:

$$M_{ang}(t) = \frac{\sum_{i=1}^{N} |r_i(t)\omega_i(t)|}{\sum_{i=1}^{N} |r_i(t)||\omega_i(t)|}, \tag{8}$$

where $r_i(t)$ refers to the distance of the drone from its center of mass of the swarm. A swarm with $M_{ang} \approx 1$ would have individuals sharing perfectly swarm rotational motion.

Results of the angular momentum of the True flocking model (black line) against the predicted flocking SINDy-SR3 model (dashed red lines) are shown in Figure 4 in

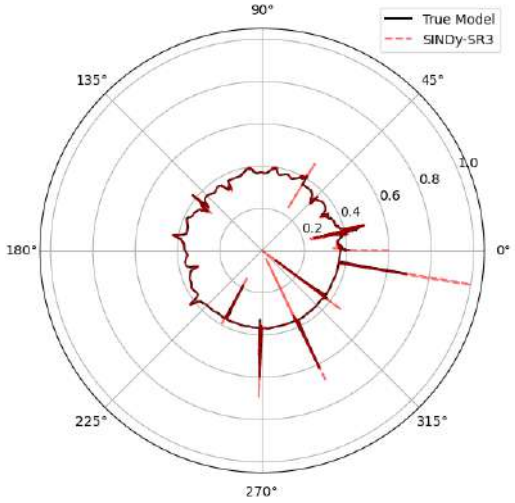


Fig. 4: Swarm angular momentum,$M_{ang}(t)$, during the testing forward simulation window.

a polar projected plot. It is shown that with the flocking SINDy-SR3 model, M_{ang} approximately fits the True one throughout almost a very long period of the testing time steps window and even can capture the sudden swarm angular rotations caused by the immediate change of the swarm angular rotation to avoid wall obstacles (See the hard picks in the plot). We may therefore presume that the SINDy-SR3 model can be used further as flocking controller implemented in each individual drone. This will hugely avoid the actual computation of the flocking vector on the actual True model at each time, and will help in revers engineering the flocking behaviour in the future.

IV. CONCLUSIONS AND FUTURE WORKS

In this paper, we adopted a data-driven methodology to discover the local interaction rules that regulate swarm dynamics. In discovering the underlying flocking dynamics rules from swarm measurement data, our proposed flocking model with the SINDy-SR3 outperformed the one with the SINDy-STLSQ model. Our findings suggest that data-driven system identification strategy has a high potential for developing more accurate control models for swarm robotics systems. The feasibility of our approaches was demonstrated through their successful assessment during the learning and testing stage.

Future research in this area could include applying our findings to other swarm behaviors. In addition, we intend to test our method's resilience and scalability on bigger swarms with more complicated dynamics, such as environmental conditions and obstacles, and varying sensor ranges. Furthermore, we intend to investigate our method's applicability to real-world swarm robotics systems and how it can be integrated as a reverse engineering approach. We aim also to investigate the performance of the SINDy flocking models by analyzing different weight configurations and their corresponding trade-offs in swarm-flocking behavior.

ACKNOWLEDGMENT

This paper is based upon a collaboration work between the Center for Robotics and Artificial Intelligence (ROMER), Middle East Technical University, Ankara and the LabRI-SBA, Ecole Superieure en Informatique, 8 Mai 1945, Sidi Bel Abess, Algeria. It belongs also to the PRFU research project N° C00L07ES220120220002 supported by General Direction of Scientific Research and Technological Development (DGRSDT).

REFERENCES

[1] B. Khaldi and F. Cherif, "An overview of swarm robotics: Swarm intelligence applied to multi-robotics," *International Journal of Computer Applications*, vol. 126, no. 2, 2015.

[2] D. Votipka, S. M. Rabin, K. Micinski, J. S. Foster, and M. M. Mazurek, "An observational investigation of reverse engineers' processes," in *Proceedings of the 29th USENIX Conference on Security Symposium*, pp. 1875–1892, 2020.

[3] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," *Nature Machine Intelligence*, vol. 3, no. 3, pp. 218–229, 2021.

[4] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations," *Journal of Computational Physics*, vol. 426, p. 109951, 2021.

[5] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *science*, vol. 324, no. 5923, pp. 81–85, 2009.

[6] B. C. Daniels and I. Nemenman, "Automated adaptive inference of phenomenological dynamical models," *Nature communications*, vol. 6, no. 1, pp. 1–8, 2015.

[7] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[8] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis, "Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 10, p. 103111, 2017.

[9] M. Habibi, S. T. Dawson, and A. Arzani, "Data-driven pulsatile blood flow physics with dynamic mode decomposition," *Fluids*, vol. 5, no. 3, p. 111, 2020.

[10] D. Bruder, C. D. Remy, and R. Vasudevan, "Nonlinear system identification of soft robot dynamics using koopman operator theory," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6244–6250, IEEE, 2019.

[11] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Sahin, "Self-organized flocking with a mobile robot swarm," in *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, May 12-16, 2008, Volume 1* (L. Padgham, D. C. Parkes, J. P. Müller, and S. Parsons, eds.), pp. 39–46, IFAAMAS, 2008.

[12] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[13] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science advances*, vol. 3, no. 4, p. e1602614, 2017.

[14] K. Champion, P. Zheng, A. Y. Aravkin, S. L. Brunton, and J. N. Kutz, "A unified sparse optimization framework to learn parsimonious physics-informed models from data," *IEEE Access*, vol. 8, pp. 169259–169271, 2020.

[15] C. Bilaloğlu, M. Şahin, F. Arvin, E. Şahin, and A. E. Turgut, "A novel time-of-flight range and bearing sensor system for micro air vehicle swarms," in *Swarm Intelligence: 13th International Conference, ANTS 2022, Málaga, Spain, November 2–4, 2022, Proceedings*, pp. 248–256, Springer, 2022.

[16] D. Bhaskar, A. Manhart, J. Milzman, J. T. Nardini, K. M. Storey, C. M. Topaz, and L. Ziegelmeier, "Analyzing collective motion with machine learning and topology," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 12, p. 123125, 2019.

Where to Place a Pile?

Miroslav Kulich[1], David Woller[1,2], Sarah Carmesin[3], Masoumeh Mansouri[3], and Libor Přeučil[1]

***Abstract*— When planning missions for autonomous machines in real-world scenarios, such as open-pit mining, painting, or harvesting, it is important to consider how the machines will alter the working environment during their operations. Traditional planning methods treat such changes, like piles built during drilling, as constraints given to the planner that depend on the machine's trajectory. The goal is to find a trajectory that satisfies these constraints. However, our approach formulates the planning problem as finding optimal positions for changes, such as piles, along the machine's trajectory. We propose a heuristic solver and provide extensive experimental evaluations.**

I. INTRODUCTION

With increasing levels of autonomy, robots are deployed in more and more complex scenarios. In a mining application, one or more drill machines operate in an open-pit mine to drill blast holes in predetermined targets. After the blast holes are drilled, they are filled with explosive material and detonated, and the ore is processed for mineral extraction. The drill machines can autonomously navigate to the targets, level themselves, drill, and retract. However, the drilling process creates piles of excess material around the hole, which must be cleared before the machine can navigate to the next target. The Drill Pattern Planning Problem (DP^3) [1] for a single drill machine or a fleet of these involves computing a time-optimal plan that ensures the machine(s) can reach each drill target, perform the defined operations, and move away from the target without colliding with obstacles, other machines, or the excess material created during the drilling process.

Mansouri et al. [1] propose a method for solving multi-vehicle DP^3 considering the dimensions of the machines, including the size of the dust guard and jacks used for leveling and the time required to perform each task. The authors break down the problem into sub-problems, identifies interdependency among the sub-problems, and interleaves reasoning within each sub-problem. The approach is further improved and defined as MVRP-DDO (Multi Vehicle Routing with Nonholonomic Constraints and Dense Dynamic Obstacles) in [2]. Carmesin et al. [3] introduce new variants of the Hamiltonian Cycle and Travelling Salesperson problems inspired by the open-pit mining application. Specifically, the authors assume dynamic graphs where edges are deleted or made untraversable depending on the already visited vertices. Besides formal definitions of the problems for such graphs, problems' properties are theoretically analyzed, and two solvers are proposed.

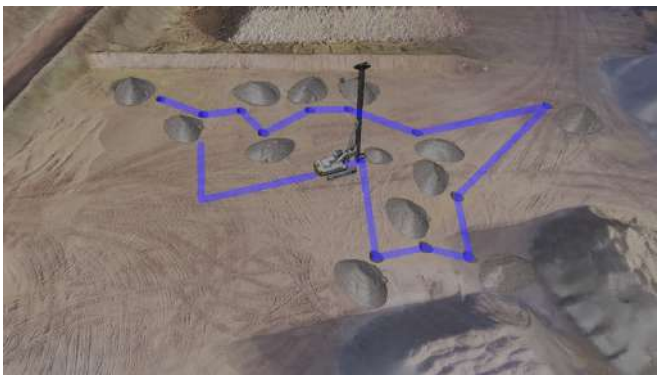


Fig. 1: Drilling scenario.

Another application where heavy vehicles are not allowed to pass already visited areas is autonomous harvesting. In the harvesting application, harvested areas limit the mobility of harvesting machines, hence affecting the reachability among the nodes representing areas to be harvested. Ullrich et al. [4] propose a graph-search planner that searches a directed graph representing the harvesting area. The designed cost function considers the number of passes through individual edges to tackle multiple passing of edges.

The aforementioned approaches assume the constraints caused by drilling/harvesting are predefined, and the planner can only affect the order in which they appear. In [1], [2], for example, piles are placed at the same positions as blast holes. Similarly, the set of edges to be deleted after visiting a vertex is predefined in [3], while edges are removed as they are traversed in [4]. Our approach is different. We assume that piles are built in the vicinity of blast holes, but their exact positions can vary, and the planning algorithm has to decide where to place them. Specifically, a trajectory for a drilling machine is given, and we ask the following questions:

- Can we place piles of a given radius so that the machine's trajectory is not obstructed by them?
- What is the largest radius for which the machine's trajectory is not obstructed by the piles?
- How should the piles be placed to ensure that the machine's path is not obstructed?

[1] Miroslav Kulich, David Woller, and Libor Přeučil are with Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, Jugoslávských partyzánů 1580/3, 160 00 Prague, Czech Republic {miroslav.kulich, david.woller, libor.preucil}@cvut.cz

[2] David Woller is also with Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo náměstí 13, 121 35 00 Prague, Czech Republic

[3] Sarah Carmesin and Masoumeh Mansouri are with School of Computer Science, University of Birmingham, Edgbaston, B15 2TT Birmingham, United Kingdom sxc1431@student.bham.ac.uk, m.mansouri@bham.ac.uk

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

We formulate two problems related to the above questions formally in Section II and propose a solver for both problems (Section III). The performance of the solver is extensively evaluated through experiments with a specially designed dataset, and the results are discussed in Section IV. The concluding remarks are presented in Section V.

II. PROBLEM FORMULATION

Let $P = \langle p_1, p_2, \dots p_n \rangle$ be a sequence of points in $\mathbb{R}^2$ forming a polygonal path, i.e., a curve consisting of line segments connecting the consecutive points. The *Path Conforming Circles Placement Problem* (PCCP) is to find a set of circles $\mathcal{K} = \{\kappa_i\}_{i \in I}$, where $I = \{1, \dots n\}$, and κ_i is a circle with center c_i and radius r_i, such that:

(C1) the radii of all circles are equal: $r_i = r \quad \forall i \in I$,
(C2) i^{th} point lies on i^{th} circle: $|p_i c_i| = r \quad \forall i \in I$,
(C3) intersection of any two circles is empty: $\kappa_i \cap \kappa_j = \emptyset \quad \forall i, j \in I, i \neq j$,
(C4) intersection of any circle with the path is empty: $\kappa_i \cap P = \emptyset \quad \forall i \in I$, and
(C5) r is maximal.

We say that κ_i is *associated* with p_i.

Let $head_k(P) = \langle p_1, p_2, \dots, p_{k-1} \rangle$ be a head of polygonal path $P = \langle p_1, p_2, \dots p_n \rangle$ and $tail_k(P) = \langle p_k, p_{k+1}, \dots, p_n \rangle$ its tail. The *Weak PCCP* (WPCCP) relaxes condition C4 by allowing for the nonempty intersection of the i^{th} circle with $head_i(P)$. The modified condition for the WCCP is thus:

(4*) $\quad \kappa_i \cap tail_i(P) = \emptyset \quad \forall i \in \langle 1, n \rangle$

Examples of the problem instances and their solutions are shown in Fig. 2. As the WPCCP is less constrained, the maximum radius found for it is higher than the one for the PCCP.

Although the PCCP and WPCCP are new, we can take inspiration from a class of problems seeking the largest empty circle. The classic example of this class is the Largest Empty Circle Problem (LEC) which consists in finding the largest circle C centered in the convex hull of a set of points such that no point lies in the interior of the circle. Shamos [5] propose an algorithm solving the LEC based on an effective search of Voronoi diagrams. Thoussaint [6] subsequently corrected the algorithm showing that Shamos made a wrong assumption about the intersection of a convex hull with a Voronoi diagram, while [7] further improved the algorithm complexity to $O(n[h \log n])$, where n is the number of points and h is the number of convex hull edges. The query variant of the LEC is addressed in [8]. The aim is to preprocess the input points to identify the largest empty circle efficiently. Finally, Augustine et al. [9] address the constrained variant where the circle has to be centered on a given line. All the aforementioned formulations search for a *single* circle while we seek a set of circles. This makes our formulation novel and challenging.

III. APPROACH

In this section, we introduce a solver for both PCCP and WPCCP. We start with the description of a general structure which is the same for both problems. The next subsections III-A to III-C then detail the individual parts of the algorithm. Finally, subsection III-D introduces modifications for the WPCCP.

The algorithm shown in Alg. 1 is motivated by the bisection method [10]. It starts with the estimation of the lower and upper bound of the radius (lines 1 and 2). Then, the bounds are modified by the iterative procedure (lines 3-10). At each iteration, the interval between the bounds is split into two halves by computing the midpoint radius (line 4) and finding the optimal placement of circles with this radius fixed (line 5). If the found placement is valid, the lower bound is replaced by the radius (line 7), and the solution is stored. If the placement is invalid, the upper bound is replaced by the radius (line 8). The process stops when the upper and lower bound difference is below the predefined limit. The stored solution and radius are returned then (line 11).

Algorithm 1: Interval bisection algorithm for the PCCP/WPCCP.

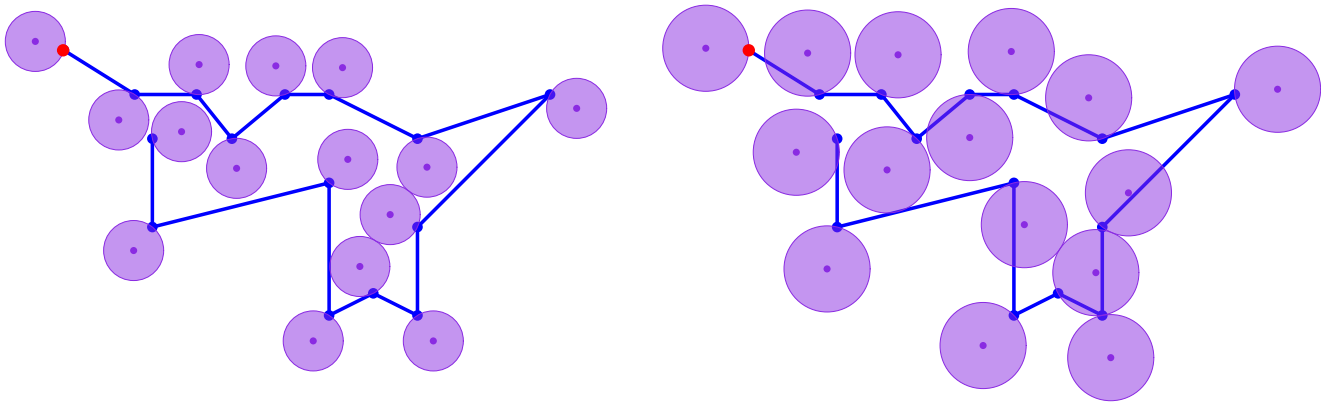
Input: $\mathcal{C}$ – set of cells

```
1  lb ← lower_bound()
2  ub ← upper_bound()
3  while (ub − lb) < ϵ do
4      radius ← (lb+ub)/2
5      (P, valid) ← find_placement(radius)
6      if valid then
7          lb ← radius
8          (P_best, radius_best) ← (P, radius)
9      else
10         ub ← radius
11 return (P_best, radius_best)
```

A. Upper Bound

Circle center c_i has to be closer to p_i than to any other point and line segment on P; otherwise, the circle would intersect P. The valuable tool for determining possible positions of circles' centers satisfying this condition is a Voronoi diagram (VD): the VD for a set of geometries (points and line segments in our case) is a partition of the plane into cells such that each cell contains exactly one input geometry and all points in the plane are closer to the geometry than to any other geometry. $VD(P)$, a Voronoi diagram of points and line segments can be computed by Fortune's sweepline algorithm in $O(n \log(n))$ time and use $O(n)$ space [11].

Fig. 3a shows the VD of a path from Fig. 2. Boundaries of Voronoi cells are formed by points equidistant from two or more input geometries. Boundary curves between two points or two segments are segments, while edges between a point and a segment are parabolic arcs. The center of the largest circle κ_i thus lies on a boundary of i^{th} cell, i.e. the cell VC_i containing i^{th} geometry. Moreover, maximal distances on boundary curves are reached at their end vertices [7].



(a) (b)

Fig. 2: Example solutions of (a) the PCCP, and (b) the WPCCP. The paths (in blue) start at the red points.

Given $VC = \{VC_i\}_{i \in I}$ a set of Voronoi cells containing points on path P we can thus determine upper bound ub of (W)PCCP as:

$$ub = \min_{i \in I} \max_{v \in \Delta(VC_i)} |vp_i|, \tag{1}$$

where $\Delta(VC_i)$ is a set of VC_i's vertices and $|\cdot|$ is the Euclidean distance. In other words, the radius of the largest circle touching a point is the distance of the point to the most distant vertex on its Voronoi cell boundary (see Fig. 3b for an example). The upper bound is the smallest of these radii.

By contradiction, we prove there is no valid solution with a radius larger than ub. Assume p_m for which $|p_m c_m| = ub$. c_m is thus the vertex maximizing the distance in Eq. 1, and m is the cell index for which this maximum is minimal. The solution for radius $r > ub$ contains circle $C_m(\bar{c}_m, r)$ associated with p_m. As c_m is the farthest point of VC_m and $\bar{c}_m$ is farther from p_m than c_m, $\bar{c}_m$ lies outside VC_m. There is, therefore, a point or segment on the path closer to $\bar{c}_m$ than to c_m, i.e., circle $\kappa_m(\bar{c}_m, r)$ has a nonempty intersection with the path. The solution is thus invalid, which is a contradiction.

B. Lower Bound

We determine lower bound lb by finding *some* solution. Assume that a circle center for each cell VC_i lies on a ray α_i which bisects the angle formed by p_i and two edges of VC_i's boundary incident to p_i as shown in Fig. 3c. For each pair $p_i, p_j \in P, i \neq j$ we determine circles' centers c_i, c_j such that:

(1) the centers lie on the bisecting rays: $c_i \in \alpha_i$ and $c_j \in \alpha_j$,
(2) the points lie on the circles with the same radius r_{ij}: $|p_i c_i| = |p_j c_j| = r_{ij}$,
(3) the circles touch: $|c_i c_j| = 2r_{ij}$.

The above conditions lead to a quadratic equation for r_{ij} with one positive solution. Nevertheless, one or both centers can lie outside the individual cells, making the solution invalid. The valid radius $\bar{r}_{ij}$ is thus limited:

$$\bar{r}_{ij} = \min\{r_{ij}, |p_i x_i|, |p_j x_j|\}, \tag{2}$$

where x_i is the intersections of ray α_i with $\delta(VC_i)$, the boundary of VC_i). Similarly, $x_j = \alpha_j \cap \delta(VC_j)$.

The lower bound is the smallest valid radius of all pairs:

$$lb = \min_{i \in I, j \in I, i \neq j} \bar{r}_{ij}.$$

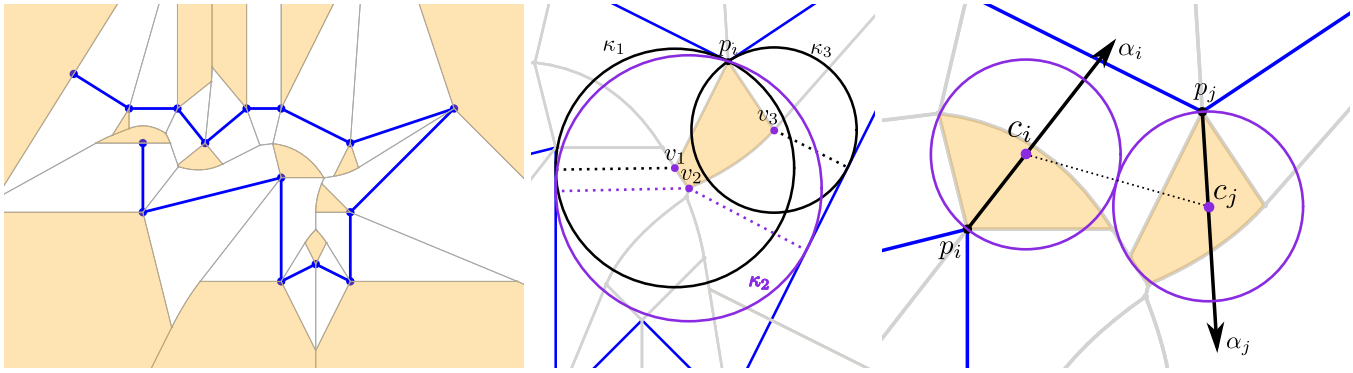
Circle center c_i is computed as the intersection of α_i with a circle centered in p_i with radius lb.

To prove the validity of the solution constructed by this procedure, we must ensure that the constraints C1–C4 from the problem formulation are satisfied. C1 and C2 are met trivially, and C4 holds as c_i lies in VC_i due to Eq. 2. C3 is proved by contradiction using the observation that given two circles touching the same point, a circle with a smaller radius is entirely inside a circle with a larger radius. This means that if the smaller circle intersects with another circle, the larger circle also intersects with the same circle. Assume now that two circles $\kappa_i(lb)$ and $\kappa_j(lb)$ with radii lb associated to some points c_i and c_j intersect. According to the above observation, $\kappa_i(lb)$ and $\kappa_j(\bar{r}_{ij})$ thus intersect as well as $\kappa_i(r_{ij})$ and $\kappa_j(r_{ij})$. This is not possible as $|c_i c_j| = 2r_{ij}$.

C. Placement for a fixed radius

In this section, we describe the algorithm which finds a valid placement of circles for given radius r or reports that such placement does not exist. We formulate this problem as a discrete optimization problem and solve it by a local search heuristic – an initially generated solution is iteratively improved by local optimization. Realize that validity of the initial solution is not guaranteed; thus, the search is done in the space of *all* solutions, not only valid ones. Invalid solutions are penalized in the designed objective function forcing the solver to find a valid one if it exists.

The algorithm shown in Alg. 2 starts by generating a set of valid circles' centers for each Voronoi cell of a point on path P (lines 1–2). Given cell VC_i, the set equals the intersection of the cell and a circle with radius r centered in p_i. To determine the intersection efficiently, the cell is split into sectors according to boundary edges as shown in Fig. 4a. The sectors are processed sequentially in clockwise order. Each sector is described by two angles – directions of rays starting



(a) (b) (c)

Fig. 3: (a) Voronoi diagram (in gray) of the blue path with highlighted cells of points (in orange). (b) Upper bound computation: Voronoi cell of p_i bounded by closed curve $p_1v_1v_2v_3p_1$, circles κ_1, κ_2, and κ_3 going through p_1 with centres v_1, v_2, and v_3, with κ_2 the largest one. The dotted lines connect the circles' centers with the points where the circles touch the path. (c) Lower bound computation: α_1 and α_2 are axes of cells associated with p_i and p_j respectively, s_i and s_j are centers of largest touching circles.

in p_i and passing through endpoints of the boundary edge e. A sequence of points on circle $\kappa(p_i, r)$ lying in the sector is generated for each sector. A point from the sequence lies in VC_i iff it is closer to p_i than to g, where g is a geometry (point or edge) that shares the boundary edge e with p_i.

The initial solution is generated next (line 3) by selecting one circle center from each $\mathcal{V}_i$. Preliminary experiments show that the selection does not influence the solution quality; we thus simply select the center randomly.

The iterative improvement is made in the loop in lines 4-11. In each iteration, points on the path are processed in a random order (lines 8-9). The randomness of the order is ensured by shuffling the indexes (line 7). When processing i^{th} point, all circles' centers are fixed except the i^{th} one and new c_i is selected from $\mathcal{V}_i$ that minimizes the designed objective function (line 9). It consists of two parts.

The first part penalizes candidate centers of circles having a non-empty intersection with other circles:

$$f^i_{int}(c) = \sum_{k \in I \setminus \{i\}} (\text{Area}(\kappa(c, r) \cap \kappa(c_k, r)) + \varepsilon_k), \qquad (3)$$

where

$$\varepsilon_k = \begin{cases} \varepsilon & \text{if } \kappa(c, r) \cap \kappa(c_k, r) \neq \emptyset \\ 0 & \text{otherwise,} \end{cases}$$

ε is a constant (see its meaning bellow), and Area is the area of circles' intersection.

Assume the optimal placement according to f_{int} in Fig. 4b for motivation of the second part. The intersection of κ_1 and κ_2 is small but nonempty. Center c_1 of circle κ_1 is the most left possible, i.e., in the best position. As the intersection of κ_2 and κ_3 is empty, $f^i_{int}(c_3) = 0$ and moving c_3 farther from κ_2 does not improve f^i_{int}. Moving c_2 farther from κ_1 shrinks κ_1–κ_2 intersection, but enlarges the intersection of κ_2 and κ_3 increasing f^i_{int} in total (notice that c_2 is in the optimal position). The solution is to move c_3 to provide space for moving c_2.

The second part of the objective function thus aims to penalize a circle (a center) with circles in its close vicinity even if it does not intersect them:

$$f^i_{dist}(c) = \sum_{k \in N} \gamma(\mu r - |cc_k|),$$

where $N = \{k | k \in I, k \neq i, |cc_k| \leq \mu r\}$, μ and γ are constants. μ specifies the size of the vicinity as the multiple of r and ensures that f^i_{dist} is non-negative. Even a tiny intersection of circles should be penalized more than many non-intersecting circles close to other circles. f^i_{int} should thus always dominate over f^i_{dist} which is the purpose of γ and ε from Eq. 3. We set $\mu = 2.2$, $\varepsilon = 10^{-5}$, and $\gamma = 10^{-10}$.

The new position of c_i minimizes the sum of the two parts:

$$c_i = \min_{c \in \mathcal{V}_i} (f^i_{int}(c) + f^i_{dist}(c)) \qquad (4)$$

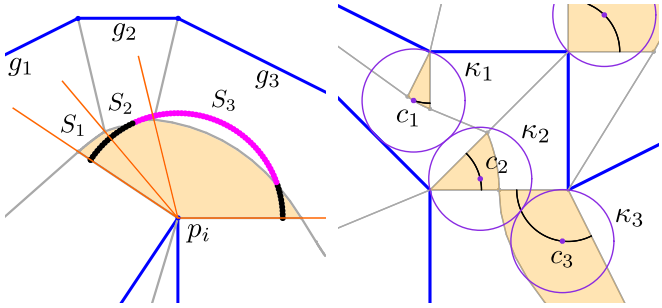
The validity of the solution is determined during the evaluation of f^i_{int}. Simply, the solution is valid if all intersections in Eq. 3 are empty.

We monitor whether the position of some center changed (line 10). If there is no such change during the processing of the entire path, the iterative process finishes (line 11), and the algorithm outputs the result (line 12).

Unfortunately, the cost function in Eq. 4 is not convex, i.e., it can have many local minima. It is thus not guaranteed that Alg. 2 finds a global optimum or even a valid solution if it exists. On the other hand, the algorithm is relatively fast and stochastic. Therefore, we run Alg. 2 several times to increase the probability of a correct result.

D. Modifications for the WPCCP

The presented algorithm is the same for both PCCP and WPCCP, with one exception. The difference lies in the way VC_i , an area of possible positions of circles' centers, is determined for a given point p_i. While a Voronoi diagram (VD) of points and line segments forming the path is computed for the PCCP, and the set of possible centers for a



(a) (b)

Fig. 4: (a) Determination of valid circle centers for a given radius. The orange lines delimit three sections S_1, S_2, and S_3 of point p_i. The black points are valid centers while the magenta ones are invalid. A center from S_k is valid iff it is closer to p_i than to geometry g_k. (b) Situation where the optimization of f_{int} does not find a valid solution.

Algorithm 2: Local search algorithm for a fixed radius.

Input: VC – set of Voronoi cells of points on path P
r – radius
Output: $\langle valid, \Upsilon \rangle$, where $\Upsilon = \{c_i\}_{i \in I}$
$valid$ – flag whether the solution is valid
$\Upsilon = \{c_i\}_{i \in I}$ – set of circles' centers

foreach $i \in I$ **do**
  $\mathcal{V}_i \leftarrow VC_i \cap C(p_i, r)$
  $c_i \leftarrow$ random($\mathcal{V}_i$)
repeat
  $modified \leftarrow false$
  $valid \leftarrow true$
  shuffle(I)
  foreach $i \in I$ **do**
    $\langle valid_i, c_i \rangle \leftarrow$ optimal_center($\mathcal{V}_i$)
    $valid \leftarrow valid \wedge valid_i$
    $modified \leftarrow modified \vee$ changed(c_i)
until $!modified$
return $\langle valid, \Upsilon \rangle$

point is directly its Voronoi cell, the process for the WPCCP is more complex. The placement of a circle for p_i in the WPCCP is influenced only by points and segments not yet visited. This means that VC_i is a Voronoi cell of p_i in the VD constructed for unvisited points and segments. The naïve approach to determine VC_i for all points is to construct the VD of relevant geometries for each point and take its Voronoi cell. The time complexity of this approach is $O(n^2 \log n)$ as a single VD is constructed in $O(n \log n)$ time. Allen et al. [12] propose the VD construction algorithm that incrementally adds new geometries for which the VD is constructed. The amortized complexity of one such insertion is $O(\sqrt{n})$, and thus total complexity of constructing all VC_i's is $O(n\sqrt{n})$. However, we use the naïve approach in our implementation.

IV. EXPERIMENTAL RESULTS

We evaluated the method's performance for both problems on two datasets we created for this purpose. The first dataset consists of 10 instances where the points are distributed evenly. Specifically, we generated hexagonal grids of various sizes, took vertices of these grids as cities, and solved the Travelling Salesman Problem for these cities by the Concorde solver [13]. The TSP solutions specify the instances `hexaXXX`, where `XXX` is the number of points in the instance.

The second dataset (instances `meshXXX`) is generated similarly, but the points are generated as vertices of a conforming constrained Delaunay triangulation (CCDT) generated by [14]. The CCDT distributes points in the plane randomly but tries to keep some minimal distance between them. Examples of both datasets are shown in Fig. 5 together with their solutions.

All experiments were performed within the same computational environment: a notebook with the Intel®Core i5-8250U CPU@1.6 Ghz. The algorithm has been implemented in C++. Twenty runs were run for each instance to provide statistically significant results.

The results are presented in Table I. Each row summarized values of 20 runs for each instance for both problems. lb and up are the lower and upper bounds, r, min, max stand for average, minimal, and maximal found valid radius, σ is the standard deviation of the radius, and $time$ is computational time in seconds.

Several observations can be made from the table. First, the mean radii are closer to the lower bounds than to the upper bounds. If we set $lb = 0\%$ and $ub = 100\%$, the radius is 10-35% for the PCCP, and 13-37% for the WPCCP. This suggests that lb is a better estimate of the optimal radius than ub.

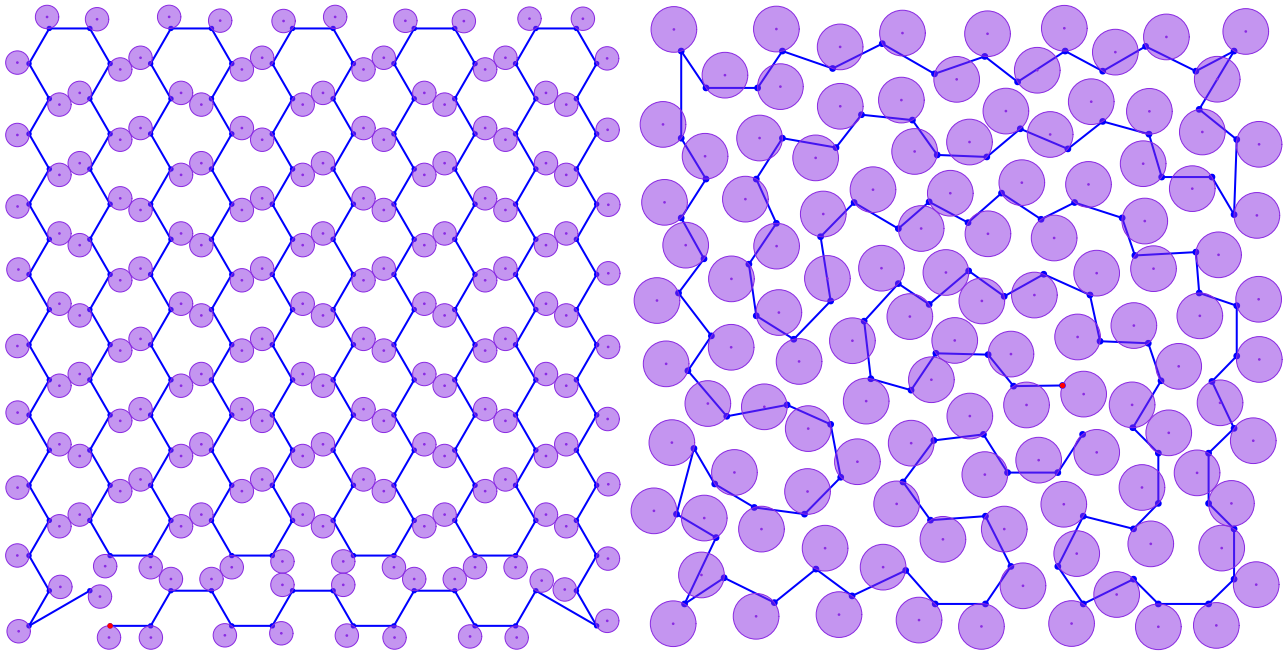
Second, the computational time is two orders higher for the WPCCP. This is caused mainly by the fact that Voronoi cells are larger than in the PCCP, and thus more centers have to be evaluated in Alg.2. Moreover, the convergence of this algorithm is slower due to a higher number of centers.

Although we run Alg. 2 twenty times, the solutions found by the solver differ as the values of σ show. The values are lower for the PCCP, meaning this problem is simpler to solve. Nevertheless, the deviations are relatively low, even for the WPCCP.

V. CONCLUSIONS

We study two problems inspired by an open-pit mining scenario. Contrary to the current approaches, we address the problems of where to place piles that do not obstruct a planned trajectory. Together with a heuristic solver for the problems, we provide upper and lower bounds for a given instance. The experimental evaluation shows that the solver finds good solutions for instances of hundreds of piles/circles in a reasonable time.

While the computational time for the PCCP is sufficient for real deployment, there is space for improvement for the WPCCP. We will thus focus on a more efficient generation of



(a) (b)

Fig. 5: Example instances and theirs solutions. (a) PCCP solution of `hexa180` (b) WPCCP solution of `mesh115`.

	PCCP							WPCCP						
problem	lb	ub	r	min	max	σ	time	lb	ub	r	min	max	σ	time
hexa180	15.00	30.00	17.31	17.31	17.31	0.00	0.50	18.02	103.90	30.66	28.75	31.39	0.64	48.82
hexa240	13.68	27.56	15.62	15.62	15.62	0.00	0.65	16.78	55.14	28.20	26.37	28.77	0.55	60.28
hexa308	12.43	25.00	14.35	14.35	14.35	0.00	0.85	15.02	86.45	24.88	24.14	25.65	0.41	99.93
hexa336	11.75	23.32	13.48	13.47	13.51	0.02	0.93	14.22	80.17	23.73	22.43	24.28	0.47	115.93
hexa416	10.64	21.00	12.23	12.23	12.23	0.00	1.14	13.05	57.95	21.32	20.50	22.00	0.46	134.56
hexa448	9.50	18.75	10.99	10.99	10.99	0.00	1.16	11.52	64.63	19.06	18.12	19.84	0.46	172.08
hexa540	9.50	18.99	10.96	10.96	10.96	0.00	1.50	11.43	65.77	18.63	18.00	19.50	0.41	213.87
hexa576	8.62	17.36	9.90	9.90	9.91	0.00	1.55	10.57	35.34	17.24	16.37	17.82	0.44	196.54
hexa836	7.50	15.00	8.66	8.66	8.66	0.00	2.16	9.04	40.41	14.76	13.93	15.41	0.41	338.74
hexa1144	6.48	13.00	7.37	7.37	7.37	0.00	3.05	7.84	34.95	12.43	11.96	12.92	0.27	413.71
mesh115	14.90	31.04	18.68	18.68	18.68	0.00	0.27	20.72	65.84	34.01	31.75	35.55	1.34	16.82
mesh244	9.25	19.23	10.47	10.29	10.72	0.14	0.37	12.17	38.87	20.22	19.57	20.90	0.49	35.75
mesh268	8.61	19.14	9.95	9.95	9.95	0.00	0.45	11.77	46.47	19.66	17.69	20.62	0.98	54.06
mesh293	7.87	17.87	8.87	8.75	8.96	0.08	0.45	10.98	38.87	17.55	17.07	17.84	0.22	41.52
mesh343	7.78	16.00	8.99	8.94	9.00	0.02	0.42	11.16	45.10	17.40	16.45	18.20	0.56	64.43
mesh374	6.99	14.15	8.76	8.65	8.85	0.06	0.47	9.49	31.50	17.40	16.32	18.11	0.42	66.10
mesh400	7.51	15.98	8.70	8.70	8.70	0.00	0.64	9.72	34.28	15.40	14.70	15.72	0.33	65.52
mesh449	7.06	11.78	8.68	8.66	8.70	0.01	0.45	9.00	26.10	15.32	14.86	15.61	0.18	87.37
mesh686	5.49	12.02	6.47	6.46	6.50	0.01	1.05	6.78	25.22	11.35	11.09	11.82	0.17	112.87
mesh1337	3.51	6.77	4.21	4.20	4.22	0.01	1.21	4.71	17.72	7.85	7.55	8.03	0.12	215.91

TABLE I: Experimental evaluation.

candidate centers in the future. Instead of generating them equidistantly, new candidates will be generated adaptively at promising positions based on the cost value of already evaluated centers.

As Voronoi diagrams can be generated for general geometries, a natural extension of the problem is to consider trajectories of other shapes. An interesting example is Dubin's car, for which optimal paths consist of straight lines and circular turns.

ACKNOWLEDGMENT

The research was supported by Czech Science Foundation Grant No. 23-05104S. The work of David Woller has also been supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS23/122/OHK3/2T/13. Masoumeh Mansouri is a UK participant in Horizon Europe Project CONVINCE, and her work is supported by UKRI grant number 10042096.

REFERENCES

[1] M. Mansouri, H. Andreasson, and F. Pecora, "Hybrid reasoning for multi-robot drill planning in open-pit mines," *Acta Polytechnica*,

vol. 56, no. 1, pp. 47–56, 2016.

[2] M. Mansouri, F. Lagriffoul, and F. Pecora, "Multi vehicle routing with nonholonomic constraints and dense dynamic obstacles," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3522–3529.

[3] S. Carmesin, D. Woller, D. Parker, M. Kulich, and M. Mansouri, "The hamiltonian cycle and travelling salesperson problems with traversal-dependent edge deletion," *SSRN Electronic Journal*, 2023. [Online]. Available: https://ssrn.com/abstract=4410404

[4] A. Ullrich, J. Hertzberg, and S. Stiene, "Ros-based path planning and machine control for an autonomous sugar beet harvester," in *Proceedings of International Conference on Machine Control & Guidance, (MCG-2014)*, 2014.

[5] M. I. Shamos, "Computational geometry." Ph.D. dissertation, Yale University, 1978.

[6] G. T. Toussaint, "Computing largest empty circles with location constraints," *International Journal of Computer & Information Sciences*, vol. 12, pp. 347–358, 10 1983. [Online]. Available: https://link.springer.com/article/10.1007/BF01008046

[7] M. Schuster, "The largest empty circle problem," in *Proceedings of the Class of 2008 Senior Conference*, 2008, pp. 28—-37.

[8] J. Augustine, S. Das, A. Maheshwari, S. C. Nandy, S. Roy, and S. Sarvattomananda, "Localized geometric query problems," *Computational Geometry*, vol. 46, no. 3, pp. 340–357, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925772112001198

[9] J. Augustine, B. Putnam, and S. Roy, "Largest empty circle centered on a query line," *Journal of Discrete Algorithms*, vol. 8, no. 2, pp. 143–153, 2010, selected papers from the 3rd Algorithms and Complexity in Durham Workshop ACiD 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570866709000847

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical recipes: The art of scientific computing," in *Finding Roots of Equations*. Cambridge University Press, 1992, ch. 9.2, pp. 352–355.

[11] S. Fortune, "A sweepline algorithm for voronoi diagrams," in *Proceedings of the Second Annual Symposium on Computational Geometry*, ser. SCG '86. New York, NY, USA: Association for Computing Machinery, 1986, p. 313–322. [Online]. Available: https://doi.org/10.1145/10515.10549

[12] S. R. Allen, L. Barba, J. Iacono, and S. Langerman, "Incremental Voronoi Diagrams," *Discrete & Computational Geometry*, vol. 58, no. 4, pp. 822–848, 2017. [Online]. Available: https://doi.org/10.1007/s00454-017-9943-2

[13] D. Applegate, R. Bixby, V. Chvatal, and W. J. Cook, "The Concorde tsp solver," http://www.math.uwaterloo.ca/tsp/concorde.html, 2003, accessed: 2023-05-01.

[14] J. R. Shewchuk, "Delaunay refinement algorithms for triangular mesh generation," *Computational Geometry*, vol. 22, no. 1, pp. 21–74, 2002, 16th ACM Symposium on Computational Geometry. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925772101000475

An EKF-based Multi-Object Tracking Framework for a Mobile Robot in a Precision Agriculture Scenario

Andrea Arlotta, Martina Lippi, Andrea Gasparri

***Abstract*— Many robotic applications require the ability to locate multiple objects in the environment, but the use of instant-by-instant identification techniques may be unreliable in variable and poorly structured contexts, such as for the majority of precision agriculture settings. Inspired by the needs of the H2020 CANOPIES projects, where robotic platforms are required to perform harvesting operations in table-grape vineyards, in this paper, we propose a framework for tracking objects of interest over time using a mobile robotic platform equipped with RGB-D camera. Specifically, we design a multi-object tracking module based on an Extended Kalman Filter (EKF) which takes into account the motion of the robot to update the estimate of the localization of the objects. We validate the approach in a realistic Unity-based simulator, where a mobile robot is tasked with tracking table-grape bunches within a vineyard environment. Additionally, we conduct preliminary tests in a laboratory setup.**

I. Introduction

In many robotic applications, the ability to locate objects within the environment is a fundamental skill that enables various tasks such as monitoring and performing operations on such objects [1]–[3]. For instance, in autonomous vehicles, the ability to identify and localize other vehicles, pedestrians, and obstacles is crucial for safe navigation and collision avoidance. However, several application scenarios, such as in agricultural settings, are often characterized by complex,dynamical, and poorly structured environments, which can pose significant challenges for object detection algorithms, thus resulting in inadequate or unreliable performances. For example, occlusions, changes in lighting conditions, and other environmental factors can hinder the accuracy of object detection algorithms, leading to incorrect or incomplete identification of objects.

To overcome these challenges, multi-object tracking systems can be realized [4]. These techniques allow updating the estimate of the objects' location, even when they are temporarily occluded or move out of view. The classical approach for multi-object tracking systems relies on the use of cameras and tracking-by-detection paradigm [5], where an object detector is usually employed to detect the objects of interest and then a tracker is used to estimate their trajectories over time. One of the seminal works in this regard is [6] which relies on a multiple hypothesis tracking (MHT) approach. Specifically, this is based on creating multiple hypotheses for the identity and trajectory of each target, and then updating these hypotheses as new observations become available.

A. Arlotta, M. Lippi, and A. Gasparri are with the Department of Civil, Computer Science and Aeronautical Technologies Engineering, Roma Tre University, Italy, 00146 e-mail: *andrea.arlotta,martina.lippi@uniroma3.it, gasparri@inf.uniroma3.it*.

This work was supported by the European Commission under the grant agreement number 101016906 – Project CANOPIES.

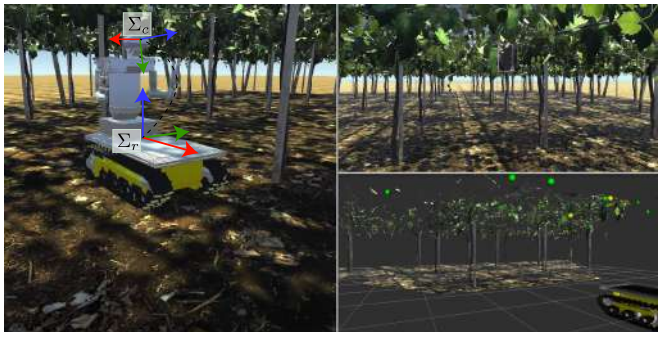


Fig. 1: Representation of the case study within a precision agriculture context: the simulated vineyard with the robot (left), the RGB image as seen from the robot (top right) and the point cloud obtained from the robot RGB-D camera (bottom right) are reported.

The algorithm uses a Bayesian probabilistic framework to calculate the likelihood of each hypothesis given the current observations and previous hypotheses. Further examples of tracking-by-detection methods include SORT (Simple Online and Realtime Tracking) [7], DeepSORT [8], and IOU Tracker [9]. Regarding the object detector, many popular approaches can be used for tracking-by-detection, such as Fast Region-based Convolutional Neural Network (R-CNN) [10] or Faster R-CNN [11] as dual-stage detectors, and You Only Look Once (YOLO) [12] or Single Shot Multibox Detector (SSD) [13] as single stage detectors. More recently, another category of methods for multi-object tracking is emerging which is based on the tracking-by-regression paradigm, where the object's state is directly regressed from the image data. Examples of direct regression methods include the recent CenterTrack [14] and FairMOT [15]. However, these methods usually require large amounts of labeled training data and can be computationally expensive.

Despite the impressive results of recent methods for object detection and localization, they mainly rely on computer vision techniques and do not model the fact that the camera may be mounted on a mobile robot, whose motion should be taken into account for tracking. In this work, we design a tracking-by-detection framework for multi-object tracking using a mobile robot equipped with an RGB-D camera. Our framework employs an Extended Kalman Filter (EKF) for each tracked object, which is updated with velocity commands from the robot and instantaneous measurements. Given a detection module based on RGB-D data, we measure the relative positioning of the detected objects with respect to the robot. This instantaneous data is then processed through a data association procedure, which determines whether each measurement is associated with a tracked object or a new

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

one. Finally, the EKFs are updated using the respective measurements and the robot's velocity. This allows to address potential inaccuracies in the detection process due to, for instance, noise or occlusions. We validate the approach in a precision agriculture context, as shown in Figure 1, where the robot is deployed into a simulated realistic vineyard and tasked with locating bunches. Furthermore, we perform preliminary evaluations in a laboratory environment. The framework proposed in this work is a crucial step toward developing an autonomous mobile robot, with manipulation capabilities, for harvesting applications. By accurately detecting and tracking multiple objects, the robot can effectively execute trajectories to harvest ripe fruits, even in situations where they may be temporarily occluded by leaves or other fruits, or are outside the camera's field of view.

II. Preliminaries

A. State and observation models

Let us consider a frame Σ_r attached to the mobile robot. We denote the frame where a variable is computed using the subscript index. If no frame is explicitly specified, we assume that the reference frame is Σ_r. Based on the requirements of the precision farming application of the H2020 project CANOPIES, which involves the detection of grape bunches to improve harvesting applications, we make the assumption that the objects of interest are static while the robot navigates within the environment. We denote the state of an object o by the vector $p_o = [p_{x,o}, p_{y,o}, p_{z,o}]^T \in \mathbb{R}^3$, which represents the object's position with respect to the robot frame. Thus, p_o denotes the location of the object o with respect to the robot. The evolution of the state variables over time is governed by nonlinear equations, which can be written as follows:

$$p_{o,k} = f(p_{o,k-1}, u_{k-1}) + w_{o,k-1}, \tag{1}$$

where $p_{o,k}$ is the state of the object at time step k, $u_{k-1} \in \mathbb{R}^m$ is the *robot* control input at time step $k-1$, f represents the state dynamics function relating the current state to the previous state and the control input, and $w_{o,k-1} \in \mathbb{R}^3$ is the process noise at time step $k-1$. The latter is modeled as a zero-mean Gaussian process with covariance $Q_{o,k-1} \in \mathbb{R}^{3\times3}$. We assume to have noisy measurements of the object state. In detail, let $z_{o,k} \in \mathbb{R}^3$ be the measurement vector of the object o at time k. The following observation model is considered:

$$z_{o,k} = h(p_{o,k}) + v_{o,k}, \tag{2}$$

where $h(\cdot)$ is a nonlinear function that maps the state variables to the measurements, and $v_{o,k} \in \mathbb{R}^3$ is a zero-mean Gaussian measurement noise with covariance $R_{o,k} \in \mathbb{R}^{3\times3}$. The specific state dynamics $f(\cdot)$ and observation model $h(\cdot)$ functions will be elaborated in the following sections with regard to the considered case study.

B. Extended Kalman Filter

The Extended Kalman Filter (EKF) is a Bayesian filter commonly used for state estimation of systems with nonlinear dynamics and measurement models [16]. Briefly, it relies on linearizing the nonlinear dynamics around the current state estimate and then propagating the state estimate forward in time using a recursive process. Two phases are recursively executed: *1) Predict,* where the current state estimate and system dynamics in (1) are exploited to predict the next state estimate; *2) Update,* where new measurements and the observation model in (2) are exploited to improve the state estimate. In detail, let $\hat{x}_{o,k}$ be the object state estimate at time k, and $P_{o,k} \in \mathbb{R}^{3\times3}$ the error covariance matrix associated with the state estimate. The prediction equations are

$$\begin{aligned} \hat{x}^-_{o,k} &= f(\hat{x}^-_{o,k-1}, u_{k-1}) \\ P^-_{o,k} &= F_{o,k-1} P_{o,k-1} F^T_{o,k-1} + Q_{o,k-1}, \end{aligned} \tag{3}$$

where $\hat{x}^-_{o,k}$ represents the predicted state estimate at time k, ${P_{o,k}}^-$ is the predicted error covariance matrix, and $F_{o,k-1}$ is the Jacobian matrix of the state model evaluated at $\hat{x}_{o,k-1}$. The update equations are

$$\begin{aligned} K_{o,k} &= P^-_{o,k} H^T_{o,k} \left(H_{o,k} P^-_{o,k} H^T_{o,k} + R_{o,k} \right)^{-1} \\ \hat{x}_k &= \hat{x}^-_{o,k} + K_{o,k} \left(z_{o,k} - h(\hat{x}^-_{o,k}) \right) \\ P_{o,k} &= \left(I_3 - K_{o,k} H_{o,k} \right) P^-_{o,k} \end{aligned} \tag{4}$$

where $H_{o,k}$ is the Jacobian matrix of the observation function evaluated at $\hat{x}^-_{o,k}$, $K_{o,k}$ is the Kalman gain and I_3 is the 3×3 identity matrix.

C. Point cloud from RGB-D camera

RGB-D cameras are sensors that provide both color (RGB) and depth (D) information about a scene. This information can be used to generate a 3D point cloud, which represents the spatial locations of points in the scene. Each point in the point cloud corresponds to a pixel in the RGB image, and its position in 3D space is determined by the depth value at that pixel. More in detail, let Σ_c be the camera frame, $f = [f_x, f_y]^T$ be the camera focal length, $c = [c_x, c_y]^T$ be the principal point and s_d be the the depth scale factor. We consider a generic pixel i with coordinates $[u_i, v_i]^T$ and depth value d_i. The respective 3D point $p^c_i = [x^c_i, y^c_i, z^c_i]^T$ in the camera frame is computed as follows:

$$\begin{bmatrix} x^c_i \\ y^c_i \\ z^c_i \end{bmatrix} = d_i \, s_d \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{c_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{c_y}{f_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}. \tag{5}$$

The overall camera point cloud $\mathcal{P}^c$ is then formed by collecting all such p^c_i points associated with pixels having non-zero depth value d_i.

D. Problem statement

Based on the above quantities, we are now ready to state the main problem that we address in this work. Let us consider a mobile robot, with frame Σ_r, which is equipped with an RGB-D camera, with frame Σ_c and let $T^r_c \in \mathbb{R}^{4\times4}$ be the homogeneous matrix of the camera frame with respect to the robot one. Let us consider that the robot navigates in an environment where n objects of interest are present, e.g., bunches in a table-grape vineyard. We aim to

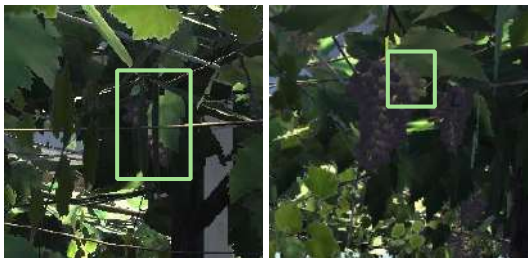


Fig. 2: Examples of occlusions in a table-grape vineyard. On the left leaves occlude a bunch, on the right a black grape bunch occludes a white grape bunch behind.

develop a real-time framework for reliably estimating $p_{o,k}$ $\forall o \in \{1, ..., n\}$, which represent the location of objects with respect to the robot frame, even when object measurements are intermittent or affected by outliers. We reiterate that such measurement inaccuracies are very frequent in unstructured outdoor environments, such as in precision agriculture settings. Figure 2 illustrates examples of occlusions in a table-grape vineyard. In the first scenario (left), a grape bunch (highlighted in a rectangle) is occluded by leaves. In the second scenario (right), a white grape bunch (highlighted in a rectangle) is occluded by a black grape bunch.

III. Proposed Framework

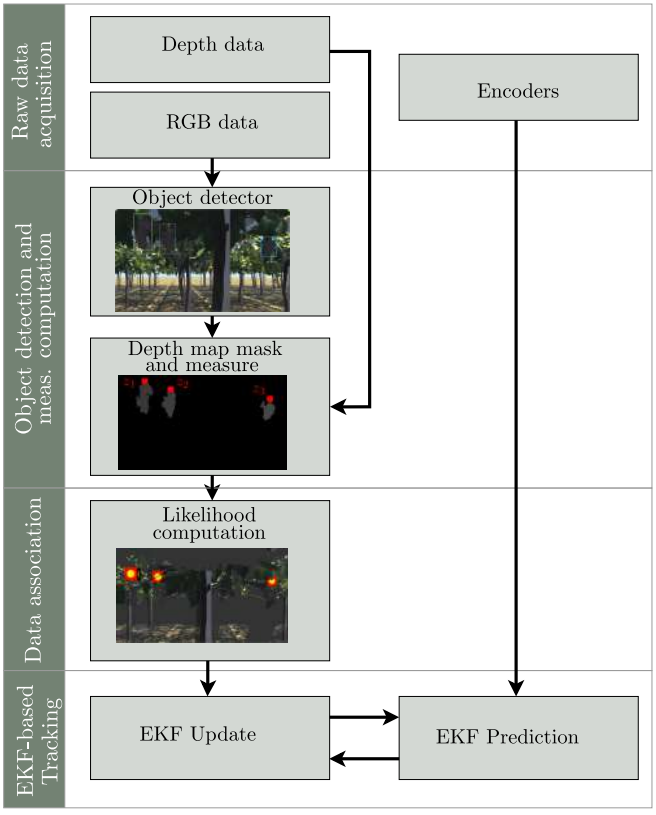


Fig. 3: Architecture overview.

The proposed framework to solve the above problem is represented in Figure 3 and structured into four main modules: *i)* raw data acquisition, *ii)* object detection and relative measurement computation, *iii)* data association, and *iv)* EKF for relative multi-object localization tracking. In the remainder of the paper, we omit the time dependency if not strictly necessary to simplify the notation. In the following, a description of each module is provided.

1) Raw data acquisition: This preliminary module, as illustrated by the first block on the left in Figure 3, gathers at each time step all the necessary data for multi-object tracking on the mobile robot. Specifically, encoder measurements are collected, which allow deriving the robot control input u_k in (3). This control input typically comprises the linear and angular velocities of the robot, as well as color and depth data from the RGB-D camera mounted on the robot.

2) Object detection and relative measurement computation: This module, shown in the second block from the left in Figure 3, is in charge of detecting the objects that are in the robot field of view at each time step k and providing the relative measurement vector $z_{o,k}$ for each detected object o.

Regarding the object detector, we consider that any off-the-shelf object detector can be employed. For instance, in our prior works for precision agriculture settings such as [2], [3], [17], we employed the YOLO architecture, while, for the H2020 CANOPIES project, a grape cluster detection module was developed in [18] that is available for use. The detector produces two primary outputs: bounding boxes that enclose the objects of interest and a segmentation mask that identifies, for each detected object, the associated pixels. These outputs are leveraged to extract measurements for each object, which are then utilized for tracking purposes. As measurement function, we use

$$h(p_{o,k}) = p_{o,k}, \tag{6}$$

i.e., we measure the entire tracking vector. Measurements are obtained as follows. For each detected object o, the respective portion of the segmentation mask is used to obtain a *masked depth map*, such that zero values are assigned to the pixels not belonging to the object, while depth values are preserved for the object pixels. Let $\mathcal{I}_o$ be the set of pixel indices with non-zero values in the masked depth map. The object point cloud $\mathcal{P}_o^c$ is generated by applying (5) for the pixels in $\mathcal{I}_o$. At this point, to retrieve the measurement vector $z_{o,k}^c$ in the camera frame, we select the pixel $[u_o, v_o]^T$ at the center of the top edge of the bounding box that surrounds the object and approximate its depth value as $\bar{d}_o = \frac{1}{|\mathcal{I}_o|} \sum_{i \in \mathcal{I}_o} d_i$, where $|\mathcal{I}_o|$ denotes the cardinality of $\mathcal{I}_o$. Then, the object measurement z_o^c is obtained by using u_o, v_o, and $\bar{d}_o$ in (5). Note that the choice of the center of the top edge of the bounding box is arbitrary and other choices might be made depending on the task at hand. For instance, the center of the bounding box might be used. In our case study, the ultimate goal is to enable a robotic hand to execute harvesting, and this choice of bunch position allows us to more likely locate and target the bunch peduncle, that is the desired cutting point for the robotic hand. Additionally, we use the average depth value $\bar{d}_o$ instead of the value d_o to calculate the object position because depth maps can be noisy, and the value d_o could even be zero. By utilizing this preprocessing step, we enhance the robustness of the system, and as we approach closer to the object of interest, more accurate perception data can be expected. As a final step, a coordinate transformation is applied to retrieve the object measurement z_o^r in the robot

frame. Specifically, it holds

$$\begin{bmatrix} z_o^r \\ 1 \end{bmatrix} = T_c^r \begin{bmatrix} z_o^c \\ 1 \end{bmatrix},$$

where T_c^r is the homogeneous matrix of the camera frame with respect to the robot one as stated in Section II-D. The measurements z_o^r obtained for all the objects detected at time step k are collected in the set of current measurements $\mathcal{M}_k$.

3) Data association: This module associates measurements with specific tracked objects in the environment. More in detail, given the set of tracked objects up to time k $\mathcal{T}_k$, collecting the states of the tracked objects, and the set of current measurements $\mathcal{M}_k$, it enables us to establish whether the measurements are associated with any tracked object and if so, the specific corresponding one. To this aim, we compute a likelihood matrix $L_k \in \mathbb{R}^{|\mathcal{M}_k| \times |\mathcal{T}_k|}$, where each element l_{ij} represents the Euclidean distance between measurement i, i.e., z_i, and tracked object j, i.e., p_j. By computing the minimum likelihood value l_i^* in each row i of L_k, i.e., $l_i^* = \min_{j \in \mathcal{T}_k} l_{ij}$, we can identify the best match between each new measurement i and the existing tracked objects. If the minimum value l_i^* is below a predefined likelihood threshold τ, we preliminarily assign the new measurement i to the tracked object $j^* = \text{argmin}_{j \in \mathcal{T}_k} l_{ij}$. If, on the other hand, the minimum value is above the threshold, we add the measure $z_{i,k}$ to the set $\mathcal{N}_k$, collecting all the measures which need to be associated with new objects. Once all measurements have been analyzed, if multiple measurements are associated with the same object, we only preserve the association with minimum likelihood value. The third block from the left in Figure 3 illustrates the current module, showing the tracked objects in yellow and the measurements in red.

4) EKF-based Multi-Object Tracking: The objective of this module is to update the set of tracked objects $\mathcal{T}$. An EKF is associated with each tracked object, as reported in Section II-B. For each vector $z_n \in \mathcal{N}_k$, a new EKF is initialized with state $p_{n,0} = z_n$ and the respective state is added to the set of tracked objects $\mathcal{T}$. Then, the prediction and update steps for each tracked object o are executed independently: the prediction step is triggered whenever a control input u_k is available, while the update step is triggered whenever a new measurement $z_{o,k}$ is available. This enables the estimation of the relative object localization information even when a measurement is unavailable at a certain time k. To define the prediction model in (3), let us consider the expression of the object position with respect to a fixed world frame Σ_w, i.e.,

$$p_o^w = p_r^w + R_r^w p_o^r,$$

where p_r^w is the robot position with respect to frame Σ_w, R_r^w is the rotation matrix between frame Σ_w and robot frame Σ_r, while p_o^r is the relative position of the object with respect to Σ_r. By deriving the previous equation, one obtains

$$\dot{p}_o^w = \dot{p}_r^w + R_r^w \dot{p}_o^r + \omega_r^w \times R_r^w p_o^r,$$

where $\dot{p}_r^w$ and ω_r^w denote the linear and angular velocities of the robot with respect to Σ_w, the operator $\times$ denotes the vector product, and the expression of the time derivative of rotation matrices has been exploited. By recalling the assumption made for our application scenario of having static objects, i.e., $\dot{p}_o^w = 0$, the relative velocity of the object with respect to frame Σ_r can be computed as

$$\dot{p}_o^r = R_w^r[-\dot{p}_r^w - \omega_r^w \times R_r^w p_o^r].$$

At this point, by considering as frame Σ_w the frame of the robot at time step $k-1$, we derive the following dynamics

$$p_{o,k+1}^r = p_{o,k}^r - R_{r,k-1}^{r,k}(\dot{p}_{r,k}^{r,k-1} + \omega_{r,k}^{r,k-1} \times R_{r,k}^{r,k-1} p_{o,k}^r) \cdot \Delta t,$$

where Δt represents the sampling time, $R_{r,k-1}^{r,k}$ expresses the rotation of the robot frame at time $k-1$ with respect to the one at time k, and $\dot{p}_{r,k}^{r,k-1}$ and $\omega_{r,k}^{r,k-1}$ denote the linear and angular velocities of the robot at time k with respect to frame at time $k-1$. For instance, if the robot frame Σ_r is oriented with the x-axis aligned with the sagittal axis in the direction of motion (as shown in red in Figure 1) and the z-axis aligned with the vertical axis (in blue), the vector $\dot{p}_{r,k}^{r,k-1}$ have zero components along y and z axis, while the vector $\omega_{r,k}^{r,k-1}$ will have zero components along x and y axis.

As far as the update phase is concerned, if a measurement in $\mathcal{M}_k$ is associated with the object o, the update equations in (4), using measurement model in (6), are used.

IV. Validation results

To evaluate the effectiveness of the proposed approach, we firstly conducted a numerical validation within a simulated vineyard environment, purposely designed for the H2020 CANOPIES project, as depicted in Figure 1. Then, we performed a preliminary validation with a real robot in a laboratory setting. In both scenarios, the mobile robot was required to track table-grape bunches in its own (moving) frame. Validation results are shown in the video at the link[1].

A. Setup description

1) Simulation setup: The Unity-based simulation setup used in this study realistically simulates a vineyard with table-grape pergola system and $3\text{m} \times 3\text{m}$ planting pattern. Different dimensions, grape densities, light orientation and intensity can be set to reproduce different operating conditions. Mobile robots, and in particular Alitrak DCT-300P mobile bases, can be included and equipped with a PAL Robotics humanoid robotic torso. Notably, the simulated environment, including the robotic platforms, is a reliable reproduction of the real-world one available for the experiments of CANOPIES project. Finally, the simulator generates ground truth data for object positions and is integrated with the Robot Operating System (ROS) middleware. In our experimental setup, we equipped the robot with an RGB-D camera mounted as shown in Figure 1. We validated the framework by having the robot navigate through the vineyard. Specifically, we created a grid of waypoints positioned at the center of each cell between the vines, as depicted in Figure 4 where triangles denote waypoints and the square denotes the start and final

[1] https://youtu.be/2s1Tz4dUbZ0

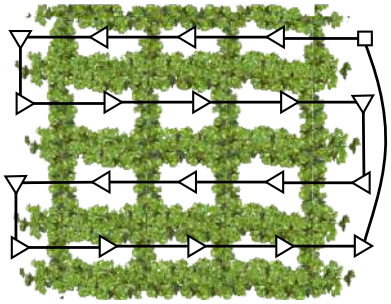


Fig. 4: Example of path employed for testing tracking system. The square marker indicates the initial waypoint and the triangle markers indicate intermediate waypoints.

configuration. These waypoints were used to plan a path that ensured complete coverage of the field.

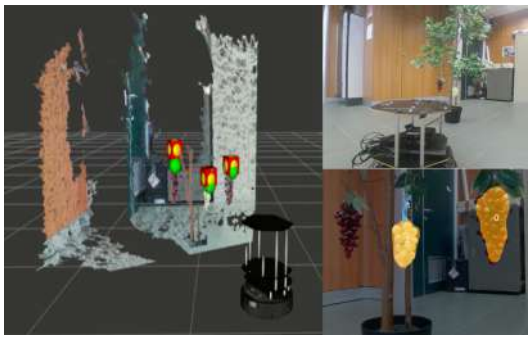


Fig. 5: Rviz environment screenshot from the laboratory validation session (left). View of the laboratory setup (top-right). Visual output of the object detection node (bottom-right).

2) Laboratory setup: Preliminary tests were conducted in the laboratory to evaluate the algorithm in a real environment, as shown in Figure 5. The setup included a synthetic tree with three artificial grape bunches and a Turtlebot 2 robot, equipped with a Realsense D435 RGB-D sensor. In addition, an Optitrack motion capture system was employed to obtain ground truth data. The robot was controlled using a joystick and moved in a random manner around the tree.

B. Implementation details

The proposed framework has been fully realized within ROS middleware with the following main nodes:
`Robot odometry node`, receiving velocity commands for the mobile base as *Twist* messages and providing *Odometry* messages containing the state from the encoders.
`Trajectory generator node`, producing the desired waypoint as *PoseStamped* message.
`Robot controller`, producing velocity commands in the form of *Twist* messages based on the desired current waypoint.
`Camera node`, handling the RGB-D camera and publishing RGB and depth as *Image* messages. Concurrently, the camera intrinsic parameters are transmitted as *CameraInfo* messages.
`Object detection node`, performing the object detection and segmentation and publishing detected bounding boxes and segmentation mask via custom messages.
`Measurement computation node`, generating the measurements of the detected bunches with respect to the robot frame. It listens to the object detection node output and, when a detection message is available, it computes the set of 3D points associated to the bunches. The output of this computation is then published via a *MarkerArray* message, which provides the list of positions of detected bunches.
`Data association node`, responsible for possibly associating sensor measurements to tracked objects. Once the association process is complete, the measurements are tagged with an ID and published in a *MarkerArray* message.
`Tracking node`, executing the EKFs associated with tracked objects. This node listens to the robot odometry topic and the output of the data association node: every time an odometry message is available, a prediction step is made for every object; while every time new measurements are available, a correction step is executed for the respective objects. The output of this computation provides a *MarkerArray* message which contains the set of the tracked objects.

Regarding the considered parameters, we set the measurement noise covariance matrix, $R_{o,k} = \text{diag}\{0.5, 0.5, 0.5\}$, and the process noise covariance matrix, $Q_{o,k} = 10^{-4} I_3$, $\forall o, k$. Additionally, we initialized the error covariance matrix at time 0, $P_{o,0} = 0.05 I_3$, $\forall o$. The similarity threshold parameter for data association was set to $\tau = 0.35$ m.

Size	# Obj.	Mean [m]	Std. Dev. [m]	Min. [m]	Max. [m]	F.P.
3x3	15	0.075	0.010	0.043	0.359	0
4x4	33	0.076	0.027	0.011	0.378	0
5x5	51	0.169	0.067	0.017	0.506	0
6x6	81	0.208	0.075	0.020	0.583	0

TABLE I: For each field dimension, the number of initialized filters, the values of the mean error, mean standard deviation, minimum error, maximum error, and number of false positives are reported.

C. Simulation results

We validated the framework with different field sizes, from 3 rows and 3 columns (denoted as 3×3) to the case of 6 rows and 6 columns (denoted as 6×6). Specifically, for each field size and for each waypoint provided by the trajectory generator, we collected the estimation errors, i.e., the distances, between ground truth data and estimated object positions through the EKFs. Table I summarizes the obtained results. Specifically, each row is associated with a field size, while columns report the number of instantiated EKFs, the average error, average standard deviation, minimum and maximum errors, and the number of False Positives (F.P., i.e., the number of active filters for the same object) obtained considering all waypoints. It can be observed that the mean and maximum error values increase with the size of the field. This is motivated by the fact that, according to our planned path, the higher the field size, the more likely it is that numerous bunches have had their latest measurements recorded long ago. Consequently, the estimates for these bunches are updated solely with predict equations in (3), resulting in an expected drift effect over time. Indeed, it is worth noting that the proposed approach is primarily well-suited for *local* multi-object tracking within the robot surroundings. The whole-field coverage simulation was merely conducted to showcase the results. In our future work, for instance, we intend to utilize the multi-object tracking system to guide the manipulator's movement for harvesting

applications. Therefore, only the information of the bunches in the vicinity of the robot are needed for an effective planning. In addition, the table reports a minimum error always greater than 0.01 m. This is motivated by the fact that, while the ground truth values correspond to the precise location of the peduncles of the bunches, we estimate their location by selecting the central points of the top edges of the bounding boxes associated with the detected bunches. This heuristics is based on the typical structure of grape bunches, where the peduncle is generally placed in the top-middle part of the bunch. Finally, the table shows that no false positives were recorded with the considered scenario.

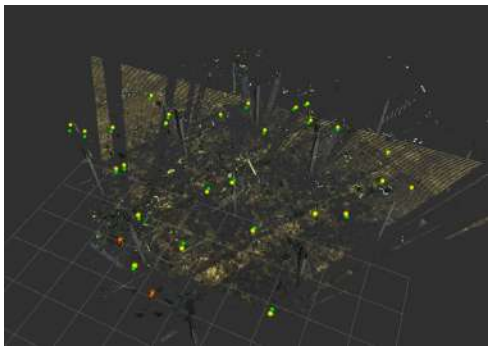


Fig. 6: Top-view screenshot in RViz environment reporting the RGB-D depth cloud, measurements (red squares), ground truth data (green spheres), and estimated positions of the objects (yellow spheres).

Figure 6 shows a visual representation (generated in RViz) of the results obtained with the coverage of a vineyard composed of 3 rows and 3 columns. It reports the depth cloud generated by the RGB-D camera as well as the markers representing the ground truth positions of grape clusters (in green), the markers depicting poses generated by the measurement node (in red), and the markers obtained by the EKFs of the tracked objects (in yellow). From the image, it can be observed that, in general, the EKF markers are relatively close to the ground truth markers. Certain EKF markers exhibit a more significant error, which can be attributed to the fact, as mentioned earlier, that the last measurement was obtained a long time ago, preventing the execution of the update equations.

D. Laboratory results

Figure 5 reports the laboratory setup (in the top right), the detection output, where the detected clusters are masked, (in the bottom right) as well as a graphical representation of the measurements, tracking instances, and ground truth (on the left). We reiterate that the detection module developed within the H2020 CANOPIES project, detailed in [18], was used. The robot moved for a total of 80 seconds. The figure, accompanied by the video, showcases the capabilities of the system to operate in a (simplified) real-world setup. Specifically, we can observe that the EKF markers (in yellow) closely approximate the ground truth markers (in green). On a quantitative basis, we recorded mean error value equal to 0.043 m with a standard deviation of 0.03 m, while maximum and minimum errors were 0.148 m and 0.003 m, respectively. All bunches were successfully tracked with no false positives.

V. Conclusions

In this work, inspired by the needs of the H2020 CANOPIES project, we presented a framework for tracking table grape bunches in a vineyard using a mobile robotic platform equipped with an RGB-D camera. The proposed multi-object position tracking module is based on an Extended Kalman Filter which takes into account the robot motion for estimating the location of the objects of interest. As future work, we aim to validate the approach in a real setup and exploit the proposed framework for driving the motion of a mobile manipulator to carry out fruit harvesting operation, during which occlusion and clutter problems occur.

References

[1] R. Verschae and J. Ruiz-del Solar, "Object detection: current and future directions," *Frontiers in Robotics and AI*, vol. 2, p. 29, 2015.

[2] M. Lippi, N. Bonucci, R. F. Carpio, M. Contarini, S. Speranza, and A. Gasparri, "A yolo-based pest detection system for precision agriculture," in *Mediterranean Conf. Control and Automation*, 2021, pp. 342–347.

[3] A. Arlotta, M. Lippi, and A. Gasparri, "A ROS-based architecture for object detection and relative localization for a mobile robot with an application to a precision farming scenario," in *Mediterranean Conf. Control and Automation*, 2023.

[4] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," *Artificial intelligence*, vol. 293, p. 103448, 2021.

[5] Z. Sun, J. Chen, L. Chao, W. Ruan, and M. Mukherjee, "A survey of multiple pedestrian tracking based on tracking-by-detection framework," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 5, pp. 1819–1833, 2020.

[6] D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[7] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *IEEE Int. Conf. Image Processing*, 2017, pp. 3645–3649.

[8] N. Wojke and A. Bewley, "Deep cosine metric learning for person re-identification," in *IEEE Winter Conf. Applications of Computer Vision*, 2018, pp. 748–756.

[9] E. Bochinski, T. Senst, and T. Sikora, "Extending iou based multi-object tracking by visual information," in *IEEE Int. Conf. Advanced Video and Signals-based Surveillance*, 2018, pp. 441–446.

[10] R. Girshick, "Fast R-CNN," in *IEEE Int. Conf. Computer Vision*, 2015, pp. 1440–1448.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[12] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Computer Vision–ECCV*. Springer, 2016, pp. 21–37.

[14] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *European Conf. Computer Vision*, 2020, pp. 474–490.

[15] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *Int. J. Computer Vision*, vol. 129, pp. 3069–3087, 2021.

[16] M. S. Grewal and A. P. Andrews, *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014.

[17] M. Lippi, R. F. Carpio, M. Contarini, S. Speranza, and A. Gasparri, "A data-driven monitoring system for the early pest detection in the precision agriculture of hazelnut orchards," *IFAC-PapersOnLine*, vol. 55, no. 32, pp. 42–47, 2022.

[18] T. A. Ciarfuglia, I. M. Motoi, L. Saraceni, M. Fawakherji, A. Sanfeliu, and D. Nardi, "Weakly and semi-supervised detection, segmentation and tracking of table grapes with limited and noisy data," *Computers and Electronics in Agriculture*, vol. 205, p. 107624, 2023.

Stereo Visual Localization Dataset Featuring Event Cameras

Antea Hadviger[1], Vlaho-Josip Štironja[1], Igor Cvišić[1], Ivan Marković[1], Sacha Vražić[2], Ivan Petrović[1]

***Abstract*—Visual odometry and SLAM methods are facing increasingly complex scenarios and novel solutions are needed to offer more accurate and reliable results in challenging environments. Standard cameras are challenged under low light conditions or very high-speed motion, as they suffer from motion blur and operate at a limited frame rate. These problems can be alleviated by using event cameras – asynchronous visual sensors that offer complementary advantages compared to standard cameras, as they do not suffer from motion blur and support high dynamic range. Although there are a number of existing datasets intended for visual odometry and SLAM that contain event data, most of them are collected using monocular sensors and limited either in terms of camera resolution or ground truth availability. Our work aims to complement this by further supporting the development of robust stereo visual odometry and SLAM algorithms, allowing to exploit both event data and intensity images. We provide both indoor sequences with 6-DoF motion and outdoor vehicle driving sequences that additionally contain 3D lidar data. All sequences contain data from a synchronized high-resolution stereo event and standard cameras, whereas ground truth trajectories are provided by either a motion capture system or a highly accurate GNSS/INS and AHRS that combines the fibre-optic gyro IMU with a dual antenna RTK GNSS receiver.**

***Index Terms*—event cameras, stereo cameras, visual odometry and SLAM, sensor fusion**

The dataset is available at http://www.bitbucket.com/unizg-fer-lamor/event-dataset.

Fig. 1: Sample intensity images (left) and accumulated events (right; color indicates event polarity) from the outdoor sequences. Second sample shows a situation where the left part of the intensity image is overexposed due to bright sunlight as the car is passing through the shaded area, while the event cameras can handle this scenario well, given their high dynamic range. Third sample is captured during a night drive.

I. Introduction

Event cameras, also known as dynamic vision sensors (DVS), are asynchronous biologically inspired sensors that detect changes in brightness intensity on a pixel-by-pixel basis. Intensity changes are reported as events as they occur, rather than capturing whole scenes with a fixed framerate like traditional cameras. Event cameras have numerous advantages over traditional cameras, including high temporal resolution in microseconds, low latency, high dynamic range up to 120 dB, and low power consumption. These sensors have a great potential to be particularly useful in challenging scenarios, such as scenes with dynamic illumination or high-speed motion. However, given that the output of event cameras is fundamentally different from traditional cameras, the full potential of these sensors can only be realized by developing new asynchronous processing methods. Nevertheless, event cameras have already proven to be valuable for various robotic perception tasks such as depth estimation, visual odometry, and simultaneous localization and mapping (SLAM). More details on event cameras, event-based methods, and results can be found in a detailed survey [1].

Even though state-of-the-art visual SLAM algorithms based on intensity images achieve high accuracy in favorable conditions, there remains a challenge to achieve robust visual SLAM under low light or high dynamic range conditions, as well as

This research has been supported by the European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS), the Croatian Science Foundation under contract No. I-3137-2019, and Rimac Technologies under the project *Visual Localization of Sports Cars*.

[1] Antea Hadviger, Vlaho-Josip Štironja, Igor Cvišić, Ivan Marković, and Ivan Petrović are with the University of Zagreb Faculty of Electrical Engineering and Computing, Laboratory for Autonomous Systems and Mobile Robotics, Croatia. {antea.hadviger, vlaho-josip.stironja, igor.cvisic, ivan.markovic, ivan.petrovic}@fer.hr

[2] Sacha Vražić is with Rimac Automobili, Croatia. sacha.vrazic@rimac-automobili.com

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

very high-speed motions, making it imperative to investigate alternative sensors. Using the complementary advantages of event and standard cameras for SLAM, visual odometry, and depth estimation has already been proven to achieve interesting results, as reviewed in a recent comprehensive survey on event-based SLAM [2]. The first attempt to combine events with intensity images and inertial measurements for robust visual SLAM was presented in [3]. More recently, [4] presents a visual-inertial odometry method that tightly-coupled the events, intensity images, and IMU by leveraging point and line features. This method based on monocular cameras was subsequently extended to support stereo cameras in [5]. Similarly, the work in [6] explores the possibility of using event-based visual-inertial odometry for navigation of planetary robots.

As the complexity of problems addressed by event-based odometry and SLAM increases in terms of the type of scene, illumination, or speed of motion, there is a rising interest for datasets featuring event cameras coupled with other sensors to enable researchers to explore novel methods with proper evaluation benchmarks at their disposal. Visual SLAM datasets such as KITTI [7], EuRoC [8], and TUM-RGBD [9] have pushed the field forward by providing high-quality data and ground truth with clear quantitative evaluation guidelines. Even though there is a number of existing datasets intended for visual odometry and SLAM that contain event-based data, most of them are monocular and limited either in terms of camera resolution or ground truth availability. Our work aims to provide a dataset to support development of robust stereo visual odometry and SLAM algorithms, allowing to exploit both event data and intensity images. We offer both indoor sequences with 6-DoF motion and outdoor sequences collected with a car driving through urban areas. All sequences contain data from a stereo high-resolution event and standard camera, while ground truth trajectories are provided from either a motion capture system or an RTK GNSS receiver. Outdoor sequences also feature 3D lidar data to support depth estimation development.

The rest of the paper is organized as follows. Section II presents related event-based datasets intended for visual odometry and SLAM. In Section III we present utilized sensors along with describing synchronization and calibration methods used for collecting the data. Finally, we conclude the paper in Section V.

II. Related Work

The first event camera simulator was presented in [10], as well as a collection of datasets including data from a DAVIS event camera (240×180 resolution), which also provides intensity images and IMU measurements. Both synthetic and real-world sequences are available. For indoor sequences, ground truth for 6-DoF motion is recorded using a motion capture system. DDD17 [11] and DDD20 [12] are both large scale datasets collected with a monocular DAVIS346 camera (346×240) mounted on a driving car. Since these datasets are intended for automated driving applications, additional data from the vehicle, such as steering angle and speed, are also provided. However, 6-DoF pose ground truth is not available, as only 2D translation can be inferred from the provided GPS latitude and longitude. The UZH-FPV Drone Racing Dataset [13] is a specialized visual-inertial odometry dataset intended for 6-DoF flying drone localization in high-speed scenarios. It features event data from the miniDAVIS346, intensity images, and IMU measurements, along with precise ground truth poses.

MVSEC is the first dataset with synchronized stereo event cameras [14]. It features a variety of data captured in different illumination levels and environments. The indoor sequences were collected with a handheld rig and a hexacopter, while sensors were mounted on top of a car and a motorcycle for outdoor driving sequences. Each DAVIS346 camera provides event streams, grayscale intensity images, and IMU readings. Additionally, 3D lidar, motion capture, and GPS data are utilized to provide ground truth trajectories and depth. However, low resolution of the DAVIS346 camera (346×240), as well as smaller stereo baseline, are limiting factors for odometry and SLAM accuracy, especially for outdoor driving scenarios.

Similarly, the DSEC dataset [15] contains data from a stereo event camera, a stereo standard camera, and 3D lidar, but with the added benefit of a higher event camera resolution. DSEC addresses the problem of low event camera resolution and small stereo baseline as the main drawbacks of MVSEC by featuring Prophesee Gen3.1 event cameras with 640×480 pixels. Even though DSEC provides a very rich amount of data collected by driving in a variety of illumination conditions, it does not contain ground truth trajectories. Furthermore, it does not cover very high speed scenarios where standard cameras would suffer from motion blur.

The TUM-VIE dataset [16] consists of a large variety of head-mounted sequences, targeting VR applications, and hand-held sequences, in both indoor and outdoor environments. The dataset contains high-resolution stereo event data (1280×720), stereo grayscale frames, as well as IMU data. However, ground truth poses are only available at the beginning and the end of each sequence, therefore limiting trajectory evaluation.

The VECtor dataset [17] is captured by a full hardware-synchronized sensor suite that includes a Prophesee Gen4 event stereo camera (1280×720), a regular stereo camera, an RGB-D sensor, a lidar, and an IMU, while ground truth trajectories are provided by a motion capture system, or by matching motion-compensated lidar scans with a dense point cloud of the environment captured by the laser scanner. Sequences are collected in large and small-scale environments. Each small-scale sequence comes in two variants (normal or fast), depending on the speed of exerted motion. The ViViD++ dataset [18] provides data from a sensor system including RGB, thermal, event, depth, and inertial measurements, along with ground truth RTK-GPS trajectory in outdoor driving scenarios. However, only a single event and standard camera is deployed.

Even though HD event cameras with a resolution of 1280×720 are available on the market and provide more

TABLE I: Properties of the sensors used for data collection

Sensor	Description
2× DVXplorer event camera	640 × 480 pixels 200μs temporal resolution up to 110 dB 165 Mega events/s
2× FLIR Blackfly S camera BFS-PGE-31S4C-C	2048 × 1536 pixels up to 60 dB global shutter 10 Hz
3× IMU integrated in DVXplorer and Ouster	3D accelerometer 3D gyroscope
MoCap OptiTrack Flex13 indoor only	accurate 6-DoF pose 850nm IR light 100 Hz
Spatial FOG Dual GNSS/INS and AHRS outdoor only	3D pose and orientation up to 8mm accuracy (with RTK) 40 Hz
Ouster OS1-128 Lidar outdoor only	128 channels Vertical FoV: 45° Angular resolution: 0.35° 10 Hz

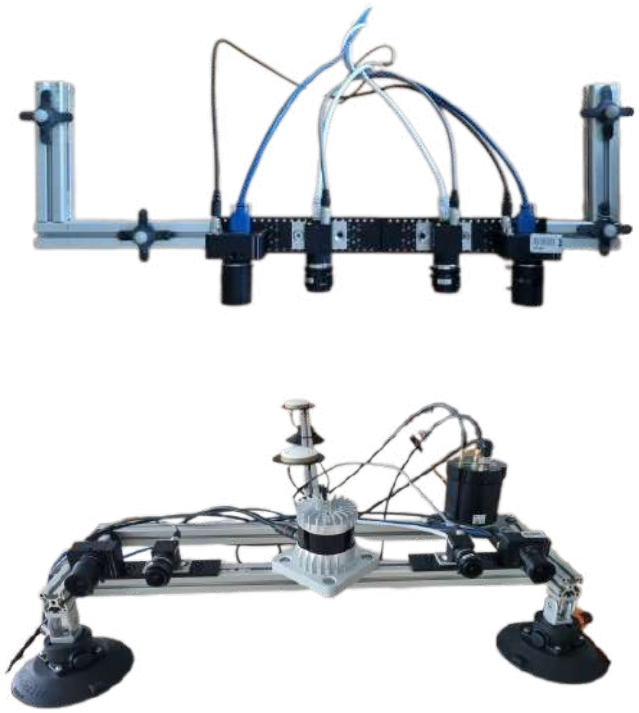


Fig. 2: Sensor setups for collecting indoor (upper image) and outdoor sequences (lower image). Event and standard stereo cameras are mounted on an aluminum rig. Indoor setup includes motion capture markers, while the GNSS dual antenna, AHRS and lidar are mounted on the outdoor rig.

detailed information about the scene, we found that the event rate exceeds the bandwidth limit during very rapid motions in scenes with rich texture, leading to significant loss of events during the recording. Bandwidth limitation is the main reason why we opted for using event cameras with 640×480 resolution for recording our dataset, as we wanted to prevent data loss and preserve all the events generated during very high speed motions, thus enabling more accurate performance of the event-based algorithms. Furthermore, the main motivation for collecting this dataset was the lack of ground truth trajectories in stereo datasets that feature high-resolution event cameras, e.g., DSEC and TUM-VIE, which we believe can complement and serve greatly visual odometry and SLAM development and evaluation.

III. The Proposed Dataset

Both our indoor and outdoor setup (Figure 2) feature a stereo event camera (with an integrated IMU) and a stereo standard camera, but with different baselines and intrinsic parameters. Additionally, the indoor rig contains motion capture markers, while the outdoor rig equips a 3D lidar (with an integrated IMU) and a highly accurate GPS aided inertial navigation system and AHRS. This section provides details about the individual sensors (summarized in Table I) and describes how we approached synchronization and calibration. All sequences are collected and published in the ROS bag format.

A. Sensor Setup

For both indoor and outdoor data acquisition we used the DVXplorer stereo event camera kit. The cameras have a spatial resolution of 640×480 pixels (VGA) and offer asynchronous output as a stream of events with temporal resolution of $200\,\mu$s, while intensity frames are not available. Their dynamic range is between 90 and 110 dB (3-100k lux with 99.9% of pixels respond to 27.5% contrast, 0.3-100k lux with 50% of pixels respond to 80% contrast). The two event cameras are enclosed in anodized aluminum cases and mounted on the rig in a horizontal stereo setup, with a baseline of 30 cm for the indoor and 57 cm for outdoor sequences. The cameras are connected to the computer using USB 3.0. The maximum throughput of events that they can handle is 165 Mega-events per second, but in practice it is also limited by the USB 3.0 bandwidth. Both event cameras are equipped with 4-12 mm varifocal CS-mounted lenses. To suppress erratic events generated by infrared (IR) flashes from the motion capture system, we additionally placed IR cut filters, in the form of thin rectangular pieces of glass, directly on top of the sensor, under the lens. To capture and visualize event data, we used the ROS event camera driver developed by iniVation [1]. The DVXplorer sensor also encapsulates a 6-axis IMU (gyro and accelerometer) with a sampling rate of 800 Hz. The event cameras parameters, usually referred to as biases, regulate the signal-to-noise ratio and the event rate of the generated stream. For outdoor and indoor sequences with brighter lighting, we set the DVXplorer biases to default values as suggested by

[1] https://gitlab.com/inivation/dv/dv-ros

the manufacturer (sensitivity level 3 out of 5). For indoor sequences with less lighting and the outdoor night sequences, the biases are set to promote slightly higher sensor sensitivity level (4 out of 5), since there is less contrast in the scene. Intensity color images are obtained using a stereo pair of FLIR Blackfly S GigE global shutter cameras, with a spatial resolution of 2048×1536 pixels with auto-exposure enabled. Even though the cameras can operate at a higher frequency, we had to lower their frame rate to 10 Hz due to bandwidth limitations. The cameras are powered through a Power-Over-Ethernet (PoE) switch and connected to the computer via Ethernet. They are rigidly mounted on the rig in a horizontal stereo setup along the same line as the event cameras. The baselines for the indoor and outdoor setup are 10 cm and 40 cm, respectively. We used an inhouse ROS camera driver that extended the existing camera drivers and wrapped them in a ROS node. Unfortunately, even with the frequency lowered to 10 Hz, frames are occasionally dropped at random from either left or right camera. In case of a dropped frame, we still retain the corresponding stereo frame in the ROS bag. Lidar data was collected using the Ouster OS1-128 high-resolution scanning lidar sensor. It has a 128 channel vertical resolution that gives 2.62 million points per second, with the vertical field of view of $45°$. Angular resolution is $0.35°$ with the output rate of 10 Hz. Additionally, Ouster OS1-128 is equipped with an IMU MPU 9250 that has an output rate of 100 Hz.

To provide ground truth poses for the indoor sequences, we used the OptiTrack motion capture system with 12 Flex13 cameras installed in our laboratory. Four reflective infrared markers were placed on the rig. The motion capture system reports both position and orientation of the rigid body defined by the markers with millimeter level precision. In our setup, the motion capture software is ran on a different computer than the one used for recording the data. The poses are transmitted to the destination computer via WiFi and captured using the dedicated ROS driver at a frequency of 100 Hz.

For outdoor sequences, we use the Spatial FOG Dual, a highly accurate GNSS/INS and AHRS that provides accurate position, velocity, acceleration, and orientation as the ground truth proxy. It combines the fibre-optic gyro IMU, which provides very accurate inertial data, accelerometers, magnetometers, and a pressure sensor with a dual antenna RTK GNSS receiver. These are coupled in a sophisticated fusion algorithm to deliver accurate and reliable 3D position and orientation. We used a TopCon base station with the quad-constellation RTK Net G5 receiver and a G5-A1 antenna mounted on the top of a 13-story building. The used GNSS has factory claimed horizontal and vertical position accuracy of 0.008 m and 0.015 m, respectively, with an output frequency of 40 Hz. On the vehicle we placed the GNSS antennas away from the other sensors to ensure that there is no interference due to multiple running data cables.

B. Indoor Sequences

Indoor sequences (Figure 3) are acquired in our laboratory with rich scene setup. The OptiTrack motion capture system is used to record the ground truth trajectories. The rig with stereo event and standard cameras was handheld (6-DoF motion) in all the indoor sequences. One part of the dataset was recorded in bright daylight with all the blinds open (event camera settings set to default). In total, there is around 7 minutes of indoor data available, divided into 8 sequences. Given that the standard cameras operate with auto-exposure settings, the exposure changes according to the amount of light coming into the sensor. Thus, there might be some overexposed or underexposed frames as the cameras are rapidly changing poses, from pointing towards the windows to away from the windows. The other part of the dataset is collected with the blinds closed and with less natural light (event camera sensitivity set to *high*). We provide several sequences where speed of motion is limited so that intensity images do not suffer from motion blur. In contrast, other sequences contain rapid motions, inducing motion blur in the intensity images, thus making them more challenging for standard visual algorithms.

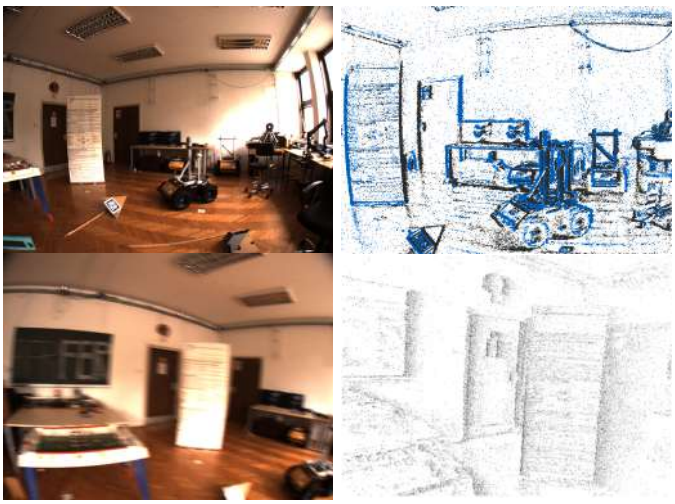


Fig. 3: Sample intensity images and accumulated events (first) and event time surfaces (second) of indoor sequences. In the second sample, motion blur is present in the intensity image, whereas the event time surface remains sharp.

C. Outdoor Sequences

To collect the data for outdoor driving sequences (Figure 1), we mounted the multi-sensor rig on the car roof. The car was driven in the central area of the city of Zagreb, Croatia. Given the high traffic in the urban settings, most sequences contain moving objects such as pedestrians and other vehicles. Note that the traffic lights, some illuminated billboards, and the lights of the other vehicles generate bursts of events due to flickering. During daytime, this does not cause any additional noise or bandwidth issues, but the lights significantly increase noise in the night sequences, especially when the streetlights are positioned directly above and next to the road. Due to auto-exposure of the standard cameras, parts of some frames can be overexposed or underexposed, given the bright daylight conditions (third sample in Fig. 1). Due to the lens and camera

Fig. 4: Ground truth trajectory for a part of our outdoor driving sequences, obtained with a GNSS and AHRS system mounted on the sensor rig placed on the top of the car.

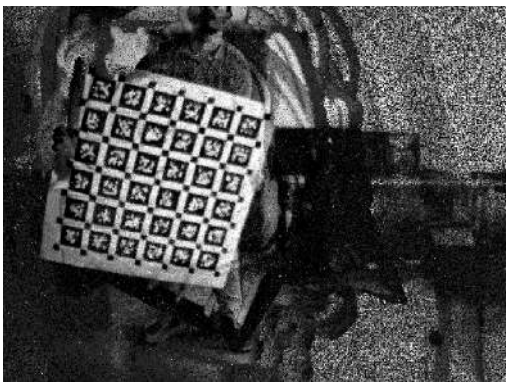


Fig. 5: Sample image used for event camera calibration, obtained by accumulating events based on their polarities within a predefined interval, and using an exponential decay kernel to determine each event's contribution to the image intensity. The image is an output of the iniVation ROS toolbox.

positioning, a part of the car hood is visible in the images. Sample ground truth trajectory obtained with the GNSS RTK and AHRS system can be seen in Fig. 4. The first of the outdoor sequences includes information from the lidar and contains around 17 minutes of data recorded in the central area of the city. This long sequence is uninterrupted, meaning that we retain the parts where the car is stopped on the traffic lights for completeness. Similarly, the night sequence contains around 12 minutes of uninterrupted recording. Additionally, there are 5 other outdoor sequences containing 17 minutes of data, including a 4-minute sequence recorded while driving through the woods, in a very highly textured environment with repetitive patterns, posing a challenge for visual localization.

IV. Synchronization and Calibration

A. Camera Synchronization and Temporal Calibration

Two standard FLIR Blackfly S cameras are synchronized to capture images at the same time by physically connecting their GPIO pins with a synchronization cable. Similarly, two DVXplorer event cameras are also synchronized from the hardware side using a dedicated synchronization cable. Since they do not capture intensity images, we do not need to worry about triggering their exposure at the same time, but their synchronization allows us to keep the timestamps of the generated events consistent across both sensors, on a microsecond precision level.

Since the clocks between the standard and event cameras are not synchronized on the hardware level, we performed temporal calibration to ensure timestamp consistency across all visual sensors. To estimate the time delay between sensors, we used the open-source ROS toolbox called Calirad [19], which implements a method for multisensor calibration based on Gaussian processes estimated moving object trajectories, resulting in reliably estimated temporal parameters. Specifically, for visual sensor calibration we used a square AprilTag [20] marker with a side length of 14.35 cm printed on a planar rigid board. Using the AprilTag detector and intrinsic camera parameters, which need to be estimated prior to this procedure, we obtain continuous 3D positions of the target in the coordinate systems of the cameras, which Calirad uses to estimate the time delay between sensors.

B. Camera Calibration

To perform intrinsic and extrinsic calibration of all four visual sensors, we use a grid of 6x6 AprilTags as a calibration target that allows for robust data association. We move the AprilTag grid in front of the static sensor rig to capture enough data that is subsequently used as the input for the Kalibr toolbox [21]. The intensity images from the standard cameras can be directly used as the input, but DVXplorer event cameras only capture events, which need to be processed in a way that would allow for the tool to detect AprilTags. Therefore, we make use of the intensity image reconstruction tool available in the iniVation event camera driver. Namely, the events are accumulated during the time interval set using the configuration parameters and reported in the form of intensity frames at the configured frame rate. Instead of naïvely accumulating events based on their polarity within a certain interval, the contribution of events to the intensity of the image is calculated by an exponential decay kernel that accounts for the events' timestamps. The resulting images, as seen in Fig. 5, can be used as an input for Kalibr to perform intrinsic and extrinsic calibration. Even though frame acquisition between the standard and event cameras is not synchronized, being able to choose an arbitrarily high event frame rate enables us to have all four intensity images acquired at approximately the same time to allow correct extrinsic calibration.

C. IMU-Camera Calibration

The iniVation ROS event camera driver provides a tool which takes as input the stream of the IMU data coming from the camera capture node and estimates the IMU biases. The camera must be placed on a stable and level surface with the gravity vector in the same direction as the Y axis of the camera frame, e.g., a table or floor. After performing intrinsic camera calibration, we collected images of a static

AprilTag calibration pattern while moving and rotating the camera around all the axes to obtain extrinsic and temporal parameters between the IMU and the event camera.

D. Lidar Extrinsics Calibration

The lidar and camera are calibrated using the open-source package [22]. The package evaluates calibration parameters using Perspective-n-Point RANSAC with Levenberg-Marquardt refinement. In order to perform the calibration, various target-based sequences with a checkerboard moving and rotating around all axes were recorded. The extrinsic calibration between lidar and its integrated IMU was assumed to be equal to the manufacturer specifications.

E. Motion Capture to Camera Calibration

OptiTrack motion capture system provides 3D position and orientation of the center of the rigid body defined by the markers placed on our rig. However, the center of the markers is not fully aligned to any camera frame, thus requiring extrinsic calibration to obtain the transformation from the camera coordinate system to the coordinate system of the motion capture markers. For this purpose, we use the same AprilTag as for the camera temporal calibration described in Section IV-A. The AprilTag detector gives us 3D positions and orientations of the target in the camera frame ${}^{\text{cam}}\mathbf{T}_{\text{tag}}$ while the motion capture system provides the position of the rig set in the motion capture frame ${}^{\text{mocap}}\mathbf{T}_{\text{rig}}$. We used these transforms, simultaneously obtained by moving the rig in front of the AprilTag, to finally estimate the position of the rig marker with respect to the camera ${}^{\text{rig}}\mathbf{T}_{\text{cam}}$ by solving the hand-eye calibration problem defined in [23] by the following equations:

$$ {}^{\text{mocap}}\mathbf{T}_{\text{rig}}^{(i)}\,{}^{\text{rig}}\mathbf{T}_{\text{cam}}\,{}^{\text{cam}}\mathbf{T}_{\text{tag}}^{(i)} = {}^{\text{mocap}}\mathbf{T}_{\text{rig}}^{(0)}\,{}^{\text{rig}}\mathbf{T}_{\text{cam}}\,{}^{\text{cam}}\mathbf{T}_{\text{tag}}^{(0)} \tag{1} $$

$$ \left({}^{\text{mocap}}\mathbf{T}_{\text{rig}}^{(0)}\right)^{-1}\,{}^{\text{mocap}}\mathbf{T}_{\text{rig}}^{(i)}\,{}^{\text{rig}}\mathbf{T}_{\text{cam}} = {}^{\text{rig}}\mathbf{T}_{\text{cam}}\,{}^{\text{cam}}\mathbf{T}_{\text{tag}}^{(0)}\left({}^{\text{cam}}\mathbf{T}_{\text{tag}}^{(i)}\right)^{-1} \tag{2} $$

We used the OpenCV *calibrateHandEye* function to solve this problem.

V. Conclusion

In this paper we have proposed a dataset aimed for developing robust stereo visual odometry and SLAM algorithms using synchronized stereo event and standard cameras of high resolution and sufficient baselines. We included 6-DoF sequences collected indoors with a handheld rig, as well as outdoor driving sequences obtained with a sensor rig mounted on the car roof. In the former case, a motion capture system provided ground truth trajectories, whereas in the latter, a highly accurate GNSS/INS and AHRS that combines the fibre-optic gyro IMU with a dual antenna RTK GNSS receiver is deployed.

References

[1] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis *et al.*, "Event-based vision: A survey," *arXiv preprint arXiv:1904.08405*, 2019.

[2] K. Huang, S. Zhang, J. Zhang, and D. Tao, "Event-based simultaneous localization and mapping: A comprehensive survey," *arXiv preprint arXiv:2304.09793*, 2023.

[3] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high speed scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[4] W. Guan, P. Chen, Y. Xie, and P. Lu, "Pl-evio: Robust monocular event-based visual inertial odometry with point and line features," *arXiv preprint arXiv:2209.12160*, 2022.

[5] P. Chen, W. Guan, and P. Lu, "Esvio: Event-based stereo visual inertial odometry," *IEEE Robotics and Automation Letters*, 2023.

[6] F. Mahlknecht, D. Gehrig, J. Nash, F. M. Rockenbauer, B. Morrell, J. Delaune, and D. Scaramuzza, "Exploring event camera-based odometry for planetary robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8651–8658, 2022.

[7] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[8] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[9] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 573–580.

[10] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, 2017.

[11] J. Binas, D. Neil, S.-C. Liu, and T. Delbruck, "Ddd17: End-to-end davis driving dataset," *arXiv preprint arXiv:1711.01458*, 2017.

[12] Y. Hu, J. Binas, D. Neil, S.-C. Liu, and T. Delbruck, "Ddd20 end-to-end event camera driving dataset: Fusing frames and events with deep learning for improved steering prediction," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[13] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6713–6719.

[14] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multi vehicle stereo event camera dataset: An event camera dataset for 3D perception," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2032–2039, 2018.

[15] M. Gehrig, W. Aarents, D. Gehrig, and D. Scaramuzza, "Dsec: A stereo event camera dataset for driving scenarios," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4947–4954, 2021.

[16] S. Klenk, J. Chui, N. Demmel, and D. Cremers, "Tum-vie: The tum stereo visual-inertial event dataset," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8601–8608.

[17] L. Gao, Y. Liang, J. Yang, S. Wu, C. Wang, J. Chen, and L. Kneip, "Vector: A versatile event-centric benchmark for multi-sensor slam," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8217–8224, 2022.

[18] A. J. Lee, Y. Cho, Y.-s. Shin, A. Kim, and H. Myung, "Vivid++: Vision for visibility dataset," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6282–6289, 2022.

[19] J. Peršić, L. Petrović, I. Marković, and I. Petrović, "Spatiotemporal multisensor calibration via gaussian processes moving target tracking," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1401–1415, 2021.

[20] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 3400–3407.

[21] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1280–1286.

[22] V. Heethesh, "Ros camera lidar calibration package, available: https://github.com/heethesh/lidar_camera_calibration," 2019.

[23] K. Daniilidis, "Hand-eye calibration using dual quaternions," *The International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, 1999.

Analyzing Data Efficiency and Performance of Machine Learning Algorithms for Assessing Low Back Pain Physical Rehabilitation Exercises

Aleksa Marusic[1], Louis Annabi[1], Sao Mai Nguyen[1,2] and Adriana Tapus[1]

***Abstract*— Physical rehabilitation focuses on the improvement of body functions, usually after injury or surgery. Patients undergoing rehabilitation often need to perform exercises at home without the presence of a physiotherapist. Computer-aided assessment of physical rehabilitation can improve patients' performance and help in completing prescribed rehabilitation exercises. In this work, we focus on human motion analysis in the context of physical rehabilitation for Low Back Pain (LBP). As 2D and 3D human pose estimation from RGB images had made impressive improvements, we aim to compare the assessment of physical rehabilitation exercises using movement data acquired from RGB videos and human pose estimation from those. In this work, we provide an analysis of two types of algorithms on a Low Back Pain rehabilitation datasets. One is based on a Gaussian Mixture Model (GMM), with performance metrics based on the log-Likelihood values from GMM. Furthermore, with the recent development of Deep Learning and Graph Neural Networks, algorithms based on Spatio-Temporal Graph Convolutional Networks (STGCN) are taken as a novel approach. We compared the algorithms in terms of data efficiency and performance, with evaluation performed on two LBP rehabilitation datasets: KIMORE and Keraal. Our study confirms that Kinect, OpenPose, and BlazePose data yield similar evaluation scores, and shows that STGCN outperforms GMM in most configurations.**

I. INTRODUCTION

Physical rehabilitation has a very important role in post-operative recovery and in the restoration of body functions [3]. Usually, during the rehabilitation process, patients performing exercises are monitored in a clinical setting by a medical professional, such as a physiotherapist. During a rehabilitation exercise session, patients' behavior reflects their health status and is an important indicator of the treatment outcome. However, patients often have a limited number of supervised sessions, and they need to continue the rehabilitation process at home without any supervision. In these cases, a physiotherapist makes a rehabilitation plan consisting of several recommended exercises. Patients are typically responsible for performing their exercises regularly at home and periodically visiting the hospital for progress assessment. However, a lack of supervision and timely feedback from healthcare professionals can reduce patient's engagement during the rehabilitation process. Lower motivation and poor supervision can increase the chances of incorrect exercise performance, which can slow down the recovery process and increase the risk of re-injury [22].

Low back pain (LBP) is a major cause of disability worldwide, with more than 50% of the global population experiencing LBP at some point in their lives [3]. This is especially concerning as LBP disproportionately affects elderly individuals, whose percentage in European societies is steadily increasing. As a result, medical staff are under significant strain to manage the growing number of patients suffering from LBP.

Automatic physical rehabilitation monitoring can significantly improve patients' progress during at-home rehabilitation. The goal of such a system is to recognize the activity being performed, the intensity with which it is performed, and its quality, thus helping monitor patients' progress. In general, human activity analysis is a very active research topic today and one of the most important and challenging areas in AI. It involves analyzing human body movements based on the motions of different body joints, skeletons, and muscles [29]. It also has applications in several domains such as sports sciences, action or gesture recognition [10], [17], [9], [2], and range-of-motion estimation [1].

Developing an effective system for movement assessment highly depends on a few factors including motion sensors, precise movement data and its pre-processing, and evaluation techniques. In recent years, there were several studies that employed machine learning methods to classify individual repetitions into correct or incorrect classes of movements. Some of the first methods proposed for this task included distance function-based algorithms such as Dynamic Time Warping and Mahalanobis distance or probabilistic models such as hidden Markov models and Gaussian mixture models [31], [7]. The outputs in these approaches are discrete class values of 0 or 1 (i.e., incorrect or correct).

Naturally, with recent developments in Neural Networks and Deep Learning (DL), there is a big interest in their application for modeling and analysis of human motions. There are already numerous papers on general Human Action Recognition (HAR) systems that utilize various DL frameworks ranging from Convolutional Networks and Long Short-Term Memory (LSTM) [15] and encoder-decoder networks to even more novel architectures such as Spatio-Temporal Graphs [34], and Attention models [23].

Furthermore, a large number of datasets related to HAR fields are freely available for analysis. These datasets are

*This work was supported by ENSTA Paris

[1]Autonomous Systems and Robotics Lab, Computer Science and System Engineering (U2IS), ENSTA Paris, Institut Polytechnique de Paris, 828 Blvd des Maréchaux, 91120 Palaiseau, France, name.surname@ensta-paris.fr

[2]Dep. Informatique, IMT Atlantique, nguyensmai@gmail.com

979-8-3503-0704-7/23/$31.00

extensively used for benchmarking algorithms for action recognition, gesture recognition, or pose estimation. However, in the medical domain, collecting large data sets of rehabilitation exercise data from patients faces multiple challenges such as impairment, unlabeled data, or privacy and safety concerns. Consequently, only a few public datasets for rehabilitation evaluation are currently available, and they are still quite smaller than more general ones, e.g. the ones used for benchmarking HAR algorithms.

Our study focuses on evaluating the performance and data efficiency of two distinct algorithms used to assess the effectiveness of rehabilitation exercises. Specifically, we investigate the Gaussian Mixture Model and the Spatio-Temporal Graph Convolutional Network algorithm. Our evaluation is conducted on two datasets that contain rehabilitation exercises for Low Back Pain patients, namely Kimore [8] and Keraal dataset.

The rest of the paper is structured as follows. Section II presents an overview of the different approaches for motion analysis and physical rehabilitation assessment. Section III describes the used datasets, while Section IV details the implemented methods for rehabilitation assessment. The results are summarized in Section V. Finally, the conclusions and discussions are presented in Section VI.

II. RELATED WORK

This section presents related work in deep learning for motion analysis in general and in the assessment of physical rehabilitation exercises.

A. Deep learning for motion analysis

Several deep learning approaches have been applied on skeleton data, in particular on the task of action recognition. Motion data can be seen as 3D tensors with one temporal dimension (the timeframe of the movement), one spatial dimension (the skeleton joint), and one feature dimension (the XYZ Euclidean position). Early deep learning approaches for motion processing focused on the temporal processing, using recurrent neural networks [12], or 1D convolutional neural networks [19]. Other approaches explored the idea of representing motion as images, in order to exploit the performances of 2D and 3D convolutional neural network for image processing [33], [18], [20], [6], [14].

With the recent development of graph neural networks, there is a new wave of algorithms that are able to properly take into account the skeleton structure using graph convolutions [34], [30], [24]. These neural network layers perform an operation that can be seen as a message passing between adjacent joints in the skeleton graph, thus properly exploiting this prior knowledge. More recently, self-attention mechanisms have been added to allow graph convolutions to span across non-adjacent joints based on dynamically computed attention coefficients [28], [27].

B. Assessment of physical rehabilitation exercises

Movement assessment is typically accomplished by comparing a patient's performance of an exercise to the desired performance as specified by therapists. A sequence of body movements is provided as input for a machine or deep learning algorithm, which should assess that exercise with a quality score. This thus requires a more precise model of the movement than most gesture classification models.

At first, studies on exercise evaluation employed more traditional machine learning methods for classification, such as Adaboost classifier, K-Nearest Neighbors, Bayesian classifier, or ANNs [4]. Others tried using distance function based models like [16]. However, classifiers only provide correct or incorrect labels, not providing any additional information or score, while distance functions solve that problem but are not able to learn from the rehabilitation data.

Further, some of the research tried using probabilistic approaches, like Hidden Markov models [7] or Gaussian Mixture Model [25]. Such models provide an assessment that is based on the likelihood that the given exercises are being drawn from a trained model. These models were able to solve previous problems, and stochastic character of human movements goes hand in hand with models nature, but they are not able to extract all the information from the data, such as joint or spatial connections among body parts.

Liao et al. [21] created a deep neural network model to generate quality scores of input movements. They proposed deep learning architecture for hierarchical spatio-temporal modeling combining GMMs, CNNs, and LSTM to provide a quality score. However, with recent development of Graph Neural Networks, it is possible to extract even more information from spatio-temporal features of the exercise. In [11] and [13], Graph Convolutional Networks (GCN) are used to assess physical rehabilitation, obtaining state-of-the-art scores on commonly used KIMORE and UI-PRMD datasets. Last but not least, [36] et al. used an ensemble of two GCN, one for position and for orientation features of the skeleton joints.

III. DATASET

This section explains the type of data used and presents two datasets used for the evaluation of the algorithms.

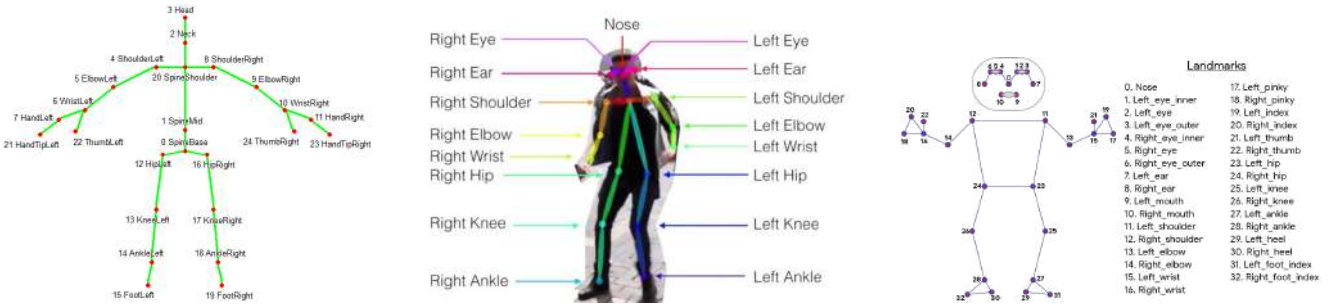
A. Skeleton data

A Human Pose Skeleton represents the orientation of a person in a graphical format. Depth cameras, like the Microsoft Kinect, can provide position and orientation of skeleton joints. They had become very popular due to their price and ease of use over optical motion tracking systems, which place a set of markers on the body. More recently, a standard vision camera can be used with deep learning techniques that estimate skeleton joints positions from plain RGB images. In this work, we will consider the algorithms OpenPose and BlazePose [37], [32]. Figure 1 displays the joints of Kinect [1], OpenPose [2] and BlazePose [3] skeletons.

[1] https://www.sealeftstudios.com/blog/blog20160708.php

[2] https://maelfabien.github.io/tutorials/open-pose/

[3] https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html



(a) Kinect v2 - 3D positions and orientations of each joint of Kinect skeleton

(b) OpenPose - 2D positions of each joint of OpenPose skeleton

(c) BlazePose - 3D positions of each joint of BlazePose skeleton

Fig. 1: Skeleton format for the three pose estimation algorithms used in the Keraal dataset. For the Kimore dataset, we only have Kinect data.

B. Keraal dataset

The Keraal dataset is a medical database of clinical patients carrying out low back-pain rehabilitation exercises. The data includes recordings from healthy subjects but, more importantly, of rehabilitation patients, extracted from a 4 weeks evolution of each patient. The centrally randomized, controlled, single-blind, and bi-centric study was conducted from October 2017 to May 2019. The rehabilitation program includes a group of 31 patients, aged 18 to 70 years, recruited in the double-blind study. 12 patients suffering from low-back pain were included in the Robot Supervised Rehabilitation Group, and were asked by a humanoid robot coach to perform each of the three predefined exercises the best they can from its demonstration. The details of this clinical trial, including the patient care, the rehabilitation sessions, the robot coach, the inclusion and exclusion criteria, the characteristics of the patients, and the efficiency of the care have been reported in [5]. Details can be read on `http://nguyensmai.free.fr/KeraalDataset.html`. A list of three exercises has been chosen in conjunction with therapists as common rehabilitation exercises that are also used for low-back pain treatment.

Videos collected from patients and healthy subjects were annotated by two physiotherapists. In this study, the labels are obtained by merging the assessments of two physicians, and we process the videos to obtain the BlazePose and OpenPose skeletons. Each exercise was labeled as either correct or incorrect. The dataset used in this study comprises Kinect (3D) v2 skeleton data , Blazepose (3D) and OpenPose (2D) skeletons obtained from videos and annotations.

C. Kimore dataset

The Kimore dataset [8] includes RGB-D videos and score annotations of five exercises for LBP rehabilitation, selected by physicians. The exercises are performed by two groups of participants: a control group (44 participants) and a group of patients (34 participants). The dataset also contains an assessment of the performed exercises, provided by two physicians. More details can be found here `https://vrai.dii.univpm.it/content/kimore-dataset`.

IV. METHODOLOGY

This section provides the technical overview of two algorithms used in this study.

A. Gaussian Mixture Model

Gaussian mixture models (GMMs) belong to a group of probabilistic models used to classify data into different categories based on probability distribution. GMM models the dataset as a mixture of several Gaussian distributions. As in [25], we encode the movement point positions as a Gaussian Mixture Model (GMM): $\theta = [t, x]$, where t is the timestamp and x the joints positions.

$$p(\theta) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\mu_i, \Sigma_i) \tag{1}$$

where the i^{th} vector component is characterized by normal distributions with weights ϕ_i, means μ_i, and covariance matrices Σ_i. Each Gaussian of the mixture is thus defined by:

$$\mu_i = \begin{bmatrix} \mu_i^t \\ \mu_i^x \end{bmatrix}, \Sigma_i = \begin{bmatrix} \Sigma_i^t & \Sigma_i^{xt} \\ \Sigma_i^{xt} & \Sigma_i^x \end{bmatrix} \tag{2}$$

where the indices t and x refer to respectively time and position.

The parameters ϕ_i, μ_i, Σ_i are learned by Expectation-Maximisation (EM) from the skeleton data of the movements captured by the Kinect or estimated with OpenPose or BlazePose.

B. Graph Convolutional Networks

Graph Neural Networks (GNNs) are a class of deep learning models that are specifically designed to operate on graph-structured data [13]. These models leverage the graph topology to learn meaningful representations of the nodes and edges of the graph.

Given our focus on skeletons, let us examine how they can be integrated into graph data. Skeleton-based data can be obtained from motion-capture devices or pose estimation algorithms from videos. Usually, the data is a sequence of frames, each frame will have a set of joint coordinates. Each joint in the given skeleton can be represented as a node in

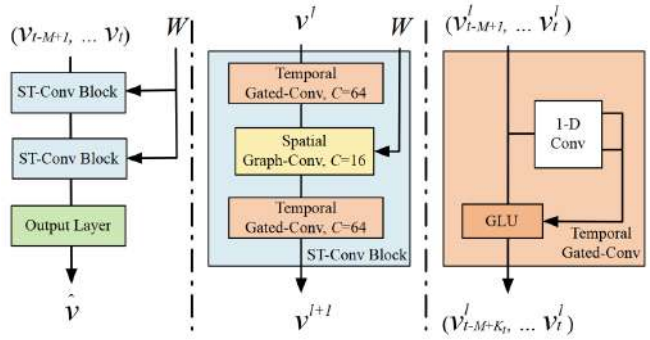


Fig. 2: Architecture of spatio-temporal graph convolutional networks. The network chains two spatio-temporal convolutional blocks (ST-Conv blocks) and a fully-connected output layer. Each ST-Conv block contains two temporal gated convolution layers and one spatial graph convolution layer in the middle. Image taken from [35].

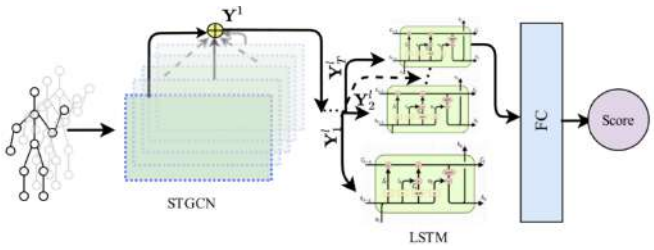


Fig. 3: Architecture of STGCN for physical rehabilitation assessment. The network combines STGCNs with LSTM layers as suggested in [11]. Image taken from [11].

a graph, while connections between the joints represent the edges in the graph (e.g., right hip to right knee). In this way, the graph provides information about the hierarchy of the human skeleton, starting from one joint as a root (e.g., mSpine) and expanding further to hands and feet, which would be the leaves of the graph. Although GNNs have been extensively used in various domains, they were first used on static graph data, where graph structure does not change once data is fitted. In recent years, there has been an increased interest in systems with temporal dimension, meaning graph data would change over time. To address this need, a new family of GNNs has emerged: Spatio-Temporal GNNs, which take into account both the spatial and temporal dimensions of the data by learning temporal representations of the graph structure.

Such architecture was first introduced in [35]. The authors proposed the architecture of spatio-temporal graph convolutional networks (STGCN). As shown in Figure 6, STGCN is composed of several spatio-temporal convolutional blocks, each of which is formed as a "sandwich" structure with two gated sequential convolution layers and one spatial graph convolution layer in between.

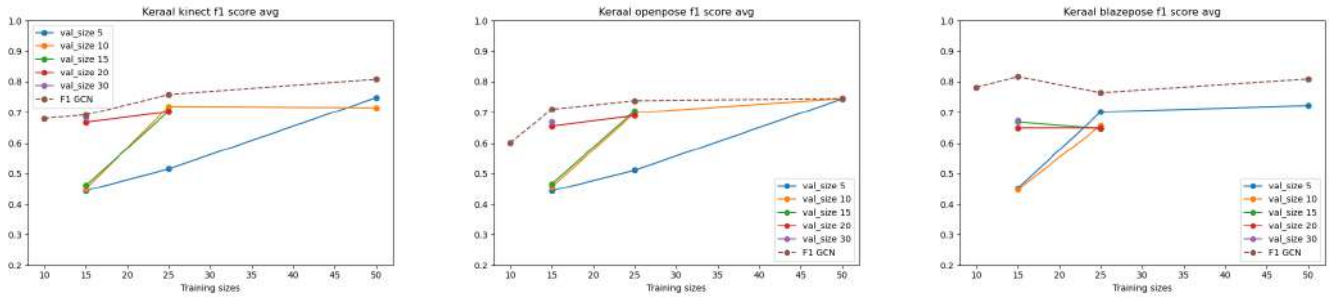
Further, in [34] this model was applied to skeleton-based action recognition, while [11] modified that algorithm for the task of assessing physical rehabilitation exercises. Since it is a very novel approach obtaining state-of-the-art results, we decided to their algorithm as the base for our analysis here. An overview of this model can be seen in Figure 3.

V. RESULTS

We experimented with the STGCN and GMM algorithms and compared their performances and sample efficiency on the two physical rehabilitation exercises datasets. The GMM is trained on correct demonstrations of the exercises, and then a classification threshold is determined based on validation data containing both correct and incorrect demonstrations. In contrast, the STGCN method needs to be trained on both correct and incorrect demonstrations. Even though validation data could be used to optimize hyperparameters or to perform early stopping, we did not use it with this method. To compare the data efficiency of both algorithms, we thus need to take into account both the number of training and validation examples needed for the GMM method and compare it with the number of training examples needed for the STGCN based method. We report the scores after training, which takes a couple of seconds for GMM to train, while STGCN, for one training of 250 epochs, takes 20 - 70 minutes depending on the setup (which dataset, skeleton type, and number of training examples). All models have been trained on CPU Intel Core i9-9900KF.

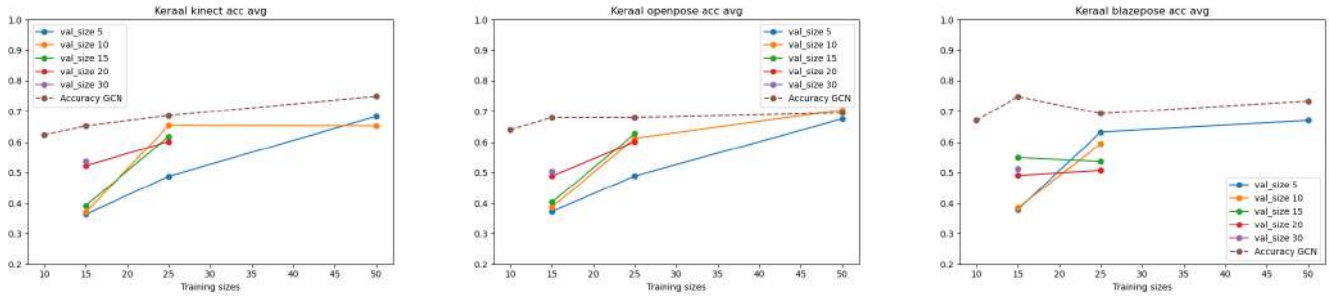
Figure 4 shows the F1 scores obtained with two methods on the Keraal dataset, while 5 provides the F1 scores. The scores are averaged across the 3 exercises of the dataset, with Kinect, OpenPose, and BlazePose poses. We can notice a slight but not significant improvement in these scores as the training set size increases. For both GMM and STGCN, we note that the scores with Kinect, Openpose and BlazePose are similar : the use of depth sensors (Kinect v2) does not seem to improve significantly the performance of the algorithm, which corroborates the conclusions presented in [26]: for low-back rehabilitation exercises, previous GMM obtained through Kinect, OpenPose and BlazePose data revealed comparable results. These new results extend the same conclusion to variations in the sizes of the training and validation sets, and to another evaluation algorithm : STGCN. This indicates that independently of the size of the dataset and the machine learning algorithm, simple RGB cameras have the potential to be used as the main sensor for collecting movement data. On Kinect and BlazePose data, the STGCN method seems to outperform the GMM method, especially when a large number of training examples are available. These results advocate in favor of using the STGCN method, even when few training examples are available.

Results for the Kimore dataset are presented in Figure 6. While GMM can achieve an F1 score of nearly 0.9 with 100 training examples, we can see that more than 200 examples are needed to achieve such classification precision with STGCN. Even when taking into consideration the additional validation examples needed for GMM, this model seems to perform significantly better when a small number of examples is available. This result contradicts what was observed for the Keraal dataset, where the STGCN method



(a) F1 scores with Kinect v2 (b) F1 scores with OpenPose (c) F1 scores with BlazePose

Fig. 4: F1 scores of GCN and GMM on the Keraal dataset, with different training sizes used. The data is averaged across exercises or groups. For GMM various sizes of validation sizes (for threshold defining) are deployed.



(a) Kinect v2 (b) OpenPose (c) BlazePose

Fig. 5: Accuracy of STGCN and GMM on the Keraal dataset.

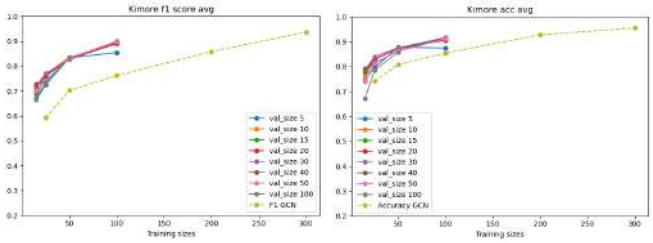


Fig. 6: F1 score and accuracy scores of STGCN and GMM on the Kimore dataset.

outperforms the GMM method even with a few training examples. Our supposition is that this could be due to the absence of a strong agreement between the physicians for Keraal (Cohen's $\kappa = 0.63$ and Krippendorff's $\alpha = 0.62$). In comparison, the Kimore dataset uses a questionnaire containing 10 questions to assess the quality of the performed exercises, which could lead to more robust labeling.

VI. CONCLUSIONS

In this work, we have compared two algorithms for LBP physical rehabilitation assessment on two datasets with several human pose estimation methods. We can draw several conclusions from the observed results, that can provide useful insights in order to further develop the use of automated physical rehabilitation methods :

- While more experiments could be done in order to confirm this result, we observed that using more expensive depth cameras does not seem to impact the performance of the assessment method. This study confirms the conclusion presented in [26] over a more extensive study using more sizes of training and validation sets and, additionally, using a more efficient evaluation algorithm. Similarly to that evaluation, we see that the use of 3D inputs (Kinect and BlazePose), compared to 2D inputs (OpenPose) does not improve the results obtained on the Keraal dataset.
- More training examples lead to a better assessment. We recommend collecting data from as many participants as possible when recording exercises.
- Label quality is essential. We observe significantly better accuracy on the Kimore dataset, where the labels were obtained by merging the answers to ten questions given by two physicians, compared to the Keraal dataset, where only two evaluations are combined. Although this reveals that assessing rehabilitation movements is a difficult task, we suggest having as many annotators as possible and monitoring a measure of their agreement to ensure high label quality.
- Finally, we recommend using the STGCN algorithm instead of the GMM algorithm in most situations. The GMM algorithm should still be useful in special cases when we need a fast (real-time) learning system or when gathering incorrectly performed exercises is difficult. It can be trained using only correct demonstrations and only needs a few incorrect demonstrations to optimize the threshold value used for classification.

REFERENCES

[1] Miron A, Sadawi N, Waidah I, Hussain H, and Grosan C. Intellirehabds (irds)—a dataset of physical rehabilitation movements. *Data*, 6(5), 2021.

[2] Tapus A., Bandera A., Vazquez-Martin R., and Calderita L. V. Perceiving the person and their interactions with the others for social robotics - a review. *Pattern Recognition Letters*, 2018.

[3] Wu A, March L, Zheng X, Huang J, Wang X, Zhao J, Blyth FM, Smith E, Buchbinder R, and Hoy D. Global low back pain prevalence and years lived with disability from 1990 to 2017: estimates from the global burden of disease study 2017. *Ann Transl Med.*, 8(6):299, January 2020.

[4] Ilktan Ar and Yusuf Sinan Akgul. A computerized recognition system for the home-based physiotherapy exercises using an rgbd camera. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(6):1160–1171, 2014.

[5] Agathe Blanchard, Sao Mai Nguyen, Maxime Devanne, Mathieu Simonnet, Myriam Le Goff-Pronost, and Olivier Rémy-Néris. Technical feasibility of supervision of stretching exercises by a humanoid robot coach for chronic low back pain: The r-cool randomized trial. *BioMed Research International*, 2022:1–10, mar 2022.

[6] Carlos Caetano, François Brémond, and William Robson Schwartz. Skeleton image representation for 3d action recognition based on tree structure and reference joints. In *2019 32nd SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*, pages 16–23. IEEE, 2019.

[7] Marianna Capecci, Maria Gabriella Ceravolo, Francesco Ferracuti, Sabrina Iarlori, Ville Kyrki, Andrea Monteriù, Luca Romeo, and Federica Verdini. A hidden semi-markov model based approach for rehabilitation exercise assessment. *Journal of Biomedical Informatics*, 78:1–11, 2018.

[8] Marianna Capecci, Maria Gabriella Ceravolo, Francesco Ferracuti, Sabrina Iarlori, Andrea Monteriu, Luca Romeo, and Federica Verdini. The kimore dataset: Kinematic assessment of movement and clinical scores for remote monitoring of physical rehabilitation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(7):1436–1448, 2019.

[9] Glowinski D, Dael N, Camurri A, Volpe G, Mortillaro M, and Scherer K. Toward a minimal representation of affective gestures. *IEEE Transactions on Affective Computing*, 2(2):106–118, 2011.

[10] Kulić D, Venture G, and Nakamura Y. Detecting changes in motion characteristics during sports training. *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4011–4014, 2009.

[11] Swakshar Deb, Md Fokhrul Islam, Shafin Rahman, and Sejuti Rahman. Graph convolutional networks for assessment of physical rehabilitation exercises. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 30:410–419, 2022.

[12] Maxime Devanne, Panagiotis Papadakis, et al. Recognition of activities of daily living via hierarchical long-short term memory networks. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 3318–3324. IEEE, 2019.

[13] Chen Du, Sarah A Graham, Colin A. Depp, and Truong Q. Nguyen. Assessing physical rehabilitation exercises using graph convolutional network with self-supervised regularization. *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 281–285, 2021.

[14] Haodong Duan, Yue Zhao, Kai Chen, Dahua Lin, and Bo Dai. Revisiting skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2969–2978, June 2022.

[15] Fuqiang Gu, Mu-Huan Chung, Mark Chignell, Shahrokh Valaee, Baoding Zhou, and Xue Liu. A survey on deep learning for human activity recognition. *ACM Computing Surveys*, 54, 08 2021.

[16] Roshanak Houmanfar, Michelle Karg, and Dana Kulić. Movement analysis of rehabilitation exercises: Distance metrics for measuring patient progress. *IEEE Systems Journal*, 10:1014–1025, 2016.

[17] Aggarwal J.K. and Ryoo M.S. Human activity analysis: A review. *ACM Comput. Surv.*, 43(3), apr 2011.

[18] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. A new representation of skeleton sequences for 3d action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3288–3297, 2017.

[19] Tae Soo Kim and Austin Reiter. Interpretable 3d human action analysis with temporal convolutional networks. In *2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pages 1623–1631. IEEE, 2017.

[20] Chuankun Li, Yonghong Hou, Pichao Wang, and Wanqing Li. Joint distance maps based action recognition with convolutional neural networks. *IEEE Signal Processing Letters*, 24(5):624–628, 2017.

[21] Yalin Liao, Aleksandar Vakanski, and Min Xian. A deep learning framework for assessing physical rehabilitation exercises. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, PP:1–1, 01 2020.

[22] Yalin Liao, Aleksandar Vakanski, Min Xian, David Paul, and Russell Baker. A review of computational approaches for evaluation of rehabilitation exercises. *Computers in biology and medicine*, 119:103687, 2020.

[23] Yanan Liu, Hao Zhang, Dan Xu, and Kangjian He. Graph transformer network with temporal kernel attention for skeleton-based action recognition. *Knowledge-Based Systems*, 240:108146, 2022.

[24] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 143–152, 2020.

[25] Nguyen S M and Tanguy P. Cognitive architecture of a humanoid robot for coaching physical exercises in kinaesthetic rehabilitation. In *International Workshop on Cognitive Robotics*, 2016.

[26] Aleksa Marusic, Sao Mai Nguyen, and Adriana Tapus. Evaluating kinect, openpose and blazepose for human body movement analysis on a low back pain physical rehabilitation dataset. In *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, pages 587–591, 2023.

[27] Vittorio Mazzia, Simone Angarano, Francesco Salvetti, Federico Angelini, and Marcello Chiaberge. Action transformer: A self-attention model for short-time pose-based human action recognition. *Pattern Recognition*, 124:108487, 2022.

[28] Chiara Plizzari, Marco Cannici, and Matteo Matteucci. Skeleton-based action recognition via spatial and temporal transformer networks. *Computer Vision and Image Understanding*, 208:103219, 2021.

[29] Sara Sardari, Sara Sharifzadeh, Alireza Daneshkhah, Bahareh Nakisa, Seng W. Loke, Vasile Palade, and Michael J. Duncan. Artificial intelligence for skeleton-based physical rehabilitation action evaluation: A systematic review. *Computers in Biology and Medicine*, 158:106835, 2023.

[30] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7912–7921, 2019.

[31] Chuan-Jun Su, Chang-Yu Chiang, and Jing-Yan Huang. Kinect-enabled home-based rehabilitation system using dynamic time warping and fuzzy logic. *Appl. Soft Comput.*, 22:652–666, 2014.

[32] Bazarevsky V, Grishchenko I, Raveendran K, Zhu T, Zhang F, and Grundmann M. Blazepose: On-device real-time body pose tracking. *CoRR*, abs/2006.10204, 2020.

[33] Pichao Wang, Zhaoyang Li, Yonghong Hou, and Wanqing Li. Action recognition based on joint trajectory maps using convolutional neural networks. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 102–106, 2016.

[34] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[35] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3634–3640, 2018.

[36] Bruce X.B. Yu, Yan Liu, Xiang Zhang, Gong Chen, and Keith C.C. Chan. Egcn: An ensemble-based learning framework for exploring effective skeleton-based rehabilitation exercise assessment. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3681–3687. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

[37] Cao Z, Hidalgo Martinez G, Simon T, Wei S, and Sheikh Y. A. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

Enhanced Visual Predictive Control Scheme for Mobile Manipulator

H. Bildstein[†], A. Durand-Petiteville[‡] and V. Cadenat[†]

***Abstract*—This paper proposes a multi-camera visual predictive control strategy for a mobile manipulator allowing to position the end-effector camera with respect to a landmark. Several issues are considered: (i) the visual landmark possible loss during navigation, (ii) the realization of large displacements which implies a large prediction horizon and impacts the closed-loop stability, (iii) the robot's high redundancy which may lead to a large search space and potential non-relevant solutions, (iv) the processing time. To cope with these challenges, the proposed strategy relies on (i) the use of two complementary cameras, (ii) the definition of a cost function depending on both the vision-based task and the manipulability, (iii) the integration of constraints allowing to prioritize the former against the latter. The strategy has been simulated and compared using ROS and Gazebo, showing its efficiency.**

I. Introduction

In this paper, we tackle the problem of controlling a mobile manipulator using a multi-camera Visual Predictive Control (VPC) [1] scheme. VPC combines the advantages of Image-Based Visual Servoing (IBVS) [2], *i.e.,* reactivity and absence of metric localization, with the ones of Nonlinear Model Predictive Control (NMPC) [3], *i.e.,* the possibility to take into account constraints such as joints limits and camera field of view during the minimization process. For these reasons, numerous VPC schemes were designed to control robotic arms [4] [5] [6], quadrotor UAVs [7], mobile robots [8], or autonomous underwater vehicles [9]. However, concerning mobile manipulators, NMPC schemes usually express the task using the end-effector pose [10] or the generalized coordinates [11] [12] [13]. Cameras are sometimes used to control mobile manipulators but the task is not defined in the image space [14] [15]. In such cases, the end-effector pose estimation accuracy has a significant impact on the control performances [2].

The design of a VPC scheme to control a mobile manipulator brings together the challenges related to mobile robots and robotic arms. First, the whole system contains many degrees of freedom, leading to a large search space for the NMPC optimization problem. We must then rely on an efficient solver in order to compute an optimal solution in a very short time. Next, the system is redundant and the end-effector pose can be obtained with an infinite number of configurations. However, these configurations are not equally suitable for the task to perform, and it is necessary to be able to select the most relevant ones. Then, if the mobile manipulator is equipped with a single camera to perform both navigation and manipulation tasks, it is challenging to keep the landmark in the field of view while performing an efficient trajectory. Thus, it might be interesting to consider a second camera to guarantee landmark visibility. Finally, unlike fixed robotic arms, a camera attached to a mobile manipulator has to perform a large displacement to reach the desired pose. This impacts the stability of the closed-loop system and it might be necessary to use large prediction horizons. To our knowledge, the works presented in [16] and [17] are among the few ones tackling some of the aforementioned challenges. In [16] the nominal VPC scheme was introduced, while [17] was a first attempt to navigate with a tucked arm. It relied on a two-step control scheme, which, despite promising results, suffered from a slow convergence rate and needed to be carefully set up.

In this paper, we present a VPC scheme taking into account the aforementioned challenges. First, the robot is equipped with two cameras, one on the end-effector and one on the head. Thus, when the end-effector camera cannot perceive the landmark, the head camera computes the visual features which are then projected on the end-effector image sphere to manage the classical perspective projection issue, *i.e.,* without projection singularities. Moreover, the positioning task is defined in the optimization problem using image moments [18], which facilitates the mapping between the task and the pose spaces. In addition to the positioning task, we also include in the optimization problem a measure of manipulability. This latter must deal with the redundancy of the robotic system by promoting configurations far from singularities. We then propose to extend the problem by adding a set of constraints, such as the classical visibility and joint limits constraints, similarly to [16]. Finally, we present the positioning constraint set guaranteeing the end-effector positioning despite the use of a local server and the presence of the manipulability measure in the optimization cost function. This method first includes the prediction-reference equality constraint, which is a modified version of the terminal constraint [3]. Next, the velocity constraint on the last predicted step is relaxed to ensure the problem's feasibility. Finally, we include a novel logarithm-based [19] constraint prioritizing the visual task over the manipulability maximization. Last but not least, the optimization problem is implemented using a symbolic representation to reduce the processing time while computing a solution sufficiently relevant to successfully achieve the task.

The rest of the paper is organized as follows. First, the different models are introduced before detailing the proposed VPC strategy and its simulation validation on TIAGo robot. Finally, the obtained results are thoroughly discussed.

[†]H. Bildstein and V. Cadenat are with CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France and Univ. de Toulouse, UPS, LAAS, F-31400, Toulouse, France {cadenat,hugo.bildstein}[at]laas.fr

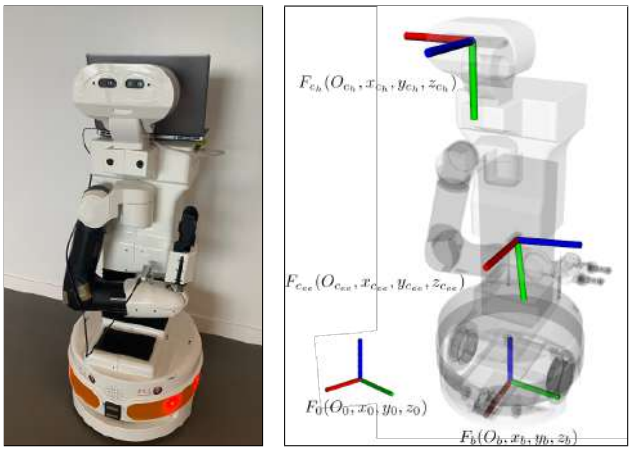
[‡]A. Durand-Petiteville is with Universidade Federal de Pernambuco UFPE, Departamento de Engenharia Mecânica, Av. da Arquitetura, 50740-550, Recife - PE, Brazil adrien.durandpetiteville[at]ufpe.br

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

II. Preliminaries

A. Robotic system description and modeling

The objective is to position a camera mounted on the end-effector of a mobile manipulator relatively to a specific landmark. The considered system is the TIAGo robot from PAL Robotics, which consists of an upper body attached to a differential mobile base (cf. Fig. 1a). The upper body includes a 2-degree-of-freedom (DoF) head and a 7-DoF arm, with two RGB-D cameras fixed on the head and wrist. As a result, the wrist camera is operated using only 5 DoF ($n_a = 5$), while the head camera only utilizes the yaw joint ($n_h = 1$).



(a) The TIAGo robot (b) The robot model

Fig. 1: The robotic system

First, we introduce four frames denoted as $F_0(O_0, x_0, y_0, z_0)$, $F_b(0_b, x_b, y_b, z_b)$, $F_{c_h}(O_{c_h}, x_{c_h}, y_{c_h}, z_{c_h})$, and $F_{c_{ee}}(O_{c_{ee}}, x_{c_{ee}}, y_{c_{ee}}, z_{c_{ee}})$, which respectively correspond to the world, mobile base, head camera, and end-effector camera frames (cf. Fig. 1b). In the sequel, the generic symbol c will be used to represent the relations for both cameras, the subscripts h or ee being indicated only when necessary. The mobile base pose and its control vector are defined as:

$$\chi_b = \left[X, Y, \theta\right]^T, u_b = \left[v, \omega\right]^T \tag{1}$$

where X, Y and θ are respectively the base coordinates in F_0 and the angle between F_b and F_0. v and ω are the linear and rotational velocities along x_b and around z_b. The arm configuration and its control vector are expressed as:

$$\chi_a = \left[q_1, q_2, q_3, q_4, q_5\right]^T, u_a = \left[\dot{q_1}, \dot{q_2}, \dot{q_3}, \dot{q_4}, \dot{q_5}\right]^T \tag{2}$$

where q_i is the i^{th} joint angle and $\dot{q_i}$ is the i^{th} joint velocity. The same reasoning holds for the head configuration and its control vector:

$$\chi_h = h_1, u_h = \dot{h}_1 \tag{3}$$

Thus, the mobile manipulator pose and its control vector are:

$$\chi_{mm} = \left[\chi_b^T, \chi_a^T, \chi_h\right]^T, u_{mm} = \left[u_b^T, u_a^T, u_h\right]^T \tag{4}$$

Now, it remains to express the end effector camera motion. Denoting by J_a and J_b the jacobian matrices of the arm and the mobile basis, the end effector camera kinematic screw $T_{C \in R_C/R}$ can be expressed as follows:

$$T_{C \in R_C/R} = \bar{J}_{b+a} \cdot \begin{bmatrix} u_b^T & u_a^T \end{bmatrix}^T \tag{5}$$

where $\bar{J}_{b+a} = \bar{J}_a + \bar{J}_b$ with:

$$\bar{J}_a = \begin{bmatrix} 0_{6\times 2} & J_a \end{bmatrix} \tag{6}$$

$$\bar{J}_b = {}^{c_{ee}}X_b \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & 0_{4\times 7} & & & \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

${}^{c_{ee}}X_b$ is the action matrix of the homogeneous transformation matrix ${}^{c_{ee}}H_b$ with:

$${}^{c_{ee}}H_b = \begin{pmatrix} {}^{c_{ee}}R_b & {}^{c_{ee}}t_b \\ 0 & 1 \end{pmatrix}, {}^{c_{ee}}X_b = \begin{pmatrix} {}^{c_{ee}}R_b & \hat{t}\ {}^{c_{ee}}R_b \\ 0 & {}^{c_{ee}}R_b \end{pmatrix} \tag{8}$$

${}^{c_{ee}}R_b$, ${}^{c_{ee}}t_b$ and $\hat{t}$ are respectively the rotation matrix between both frames, the position vector $O_{c_{ee}}O_b$, the skew-symmetric matrix deduced from ${}^{c_{ee}}t_b$.

B. Spherical projection method and visual features

Now, we focus on the choice of visual features allowing us to characterize the landmark. As classically done, we extract N interest points from the image provided by the camera. We can thus define a first visual feature vector S_{ip} made of the coordinates (x_i, y_i) of the N interest points of the landmark[1].

$$S_{ip} = \left[x_1, y_1, \ldots x_i, y_i, \ldots x_N, y_N\right]^T \tag{9}$$

If 2D points are often used in visual servoing, it has been shown that they induce a strong DoF coupling, which may be an issue with complex systems such as a mobile manipulator. In this context, some works have exhibited an interest in considering the spherical projection model and using 3D moments [18]. To consider this approach, it is necessary to determine the 3D point position (X_i, Y_i, Z_i) in the camera frame. As the robot is equipped with RGB-D cameras, Z_i is available. Considering a normalized focal distance, X_i and Y_i can be deduced using the perspective projection model:

$$\begin{bmatrix} X_i \\ Y_i \end{bmatrix} = \begin{bmatrix} Z_i & 0 \\ 0 & Z_i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{10}$$

The spherical projection consists in the projection of the 3D points $\boldsymbol{X}_i = \left[X_i, Y_i, Z_i\right]^T$ on the unit sphere centered in O_c:

$$\left[\tilde{x}_i, \tilde{y}_i, \tilde{z}_i\right]^T = \boldsymbol{X}_i / \|\boldsymbol{X}_i\| \tag{11}$$

If O is the observed object and O_{sp} its spherical projection, 3D discrete moments are defined by :

$$m_{l,j,k} = \sum_{O_{sp}} \tilde{x}_i^l \tilde{y}_i^j \tilde{z}_i^k \tag{12}$$

From them, we have built the following visual features vector allowing us to obtain a good DoF decoupling, thus making easier the tuning of certain parameters of the control law [16]:

$$S = \left[x_g, y_g, I_1, N_v \times z_c, z_g, \alpha_{sp}\right]^T \tag{13}$$

[1]Generally, it is necessary to consider at least 4 points to control the whole camera motion.

where (x_g, y_g, z_g) are the landmark gravity center coordinates, I_1 is a suitable combination of 3D moments, N_v the normal vector to the target plane and α_{sp} the orientation of the object projection around z_c. More details can be found in [16] and [18]. Using this projection method and the proposed visual features allows: (i) avoiding the inherent singularity around $Z_c = 0$ of the classical perspective projection and (ii) obtaining a nice decoupling DoF behavior [18], which will ease the tuning of the control law.

C. The re-projection model: the multi-camera solution

As previously mentioned, our robot is equipped with two cameras fixed on the end-effector and on the head. The task consists in positioning the first one with respect to a landmark. However, using only the visual features provided by this latter camera may lead to an undesired behavior: the arm will be stretched towards the landmark during the navigation, inducing vibrations and perturbations. To avoid this issue and allow motions with a tucked arm, we propose to project the visual information of the head camera in the end-effector camera frame. This re-projection is done using the homogeneous transformation matrix ${}^{c_{ee}}H_{c_h}$ which depends on χ_a and χ_h. Thus, when the end-effector camera cannot perceive the landmark, the head camera computes the visual features which are then projected on the end-effector image sphere to manage the classical perspective projection issue, *i.e.,* without projection singularities. The control law will then be fed using the visual features either directly provided by the wrist camera as mentioned above or recomputed from the data issued by the head vision system thanks to the re-projection model.

III. The multi-camera VPC strategy

Now, we focus on our VPC strategy. We first state the considered optimal control problem before detailing the required different elements and constraints.

A. The VPC control problem

As mentioned before, VPC is the result of coupling NMPC with IBVS. It thus shares characteristics from these two particular control techniques. As NMPC, it consists in finding an optimal control sequence $U^*(\cdot)$ that minimizes a cost function J_{N_p} over a N_p steps prediction horizon under a set of user-defined constraints $C(U(\cdot))$. The obtained optimal control sequence is a N_c-dimensional vector where N_c is called the control horizon. It means that the N_c first predictions of the N_p long prediction horizon are computed using independent control inputs, while all the remaining ones are obtained using a unique control input equal to the N_c^{th} element of $U(\cdot)$. Now, let us focus on J_{N_p}. It is made of two terms. The first one F_{vs}, similarly to IBVS, explicitly depends on the visual features S and is expressed as the weighted quadratic error between the predicted visual features vector $\hat{S}$ and the desired ones S^*. The weighting is done through a diagonal positive definite matrix denoted by Q_S which allows to prioritize specific DoF against others. This matrix can be easily tuned thanks to the nice decoupling properties of the considered visual features vector S. The second term, F_w, is intended to improve the manipulability of the arm and of the entire mobile manipulator. It is defined by weighting two dedicated indices w'_a and w'_{b+a} through a gain $\alpha_w > 0$. These indices, which tend to zero when the robot comes closer to singularities and joint limits, will be defined in the next section. Finally, the balance between F_w and F_{vs} is performed through a dedicated gain denoted by $K_w > 0$. This leads to the following optimal control problem:

$$U^*(\cdot) = \min_{U(\cdot)} \left(J_{N_p}(S(k), U(\cdot)) \right) \tag{14}$$

with

$$J_{N_p}(S(k), U(\cdot)) = \sum_{p=k+1}^{k+N_p} F(p) \tag{15}$$

and

$$F(p) = F_{vs}(p) + K_w F_w(p) \tag{16}$$

$$F_{vs}(p) = \left[\hat{S}(p) - S^*\right]^T Q_S \left[\hat{S}(p) - S^*\right] \tag{17}$$

$$F_w = \alpha_w / \hat{w}_a(p) + (1 - \alpha_w) / \hat{w}_{b+a}(p) \tag{18}$$

subject to

$$\hat{S}(k) = S(k) \tag{19a}$$

$$\hat{\chi}_a(k) = \chi_a(k) \tag{19b}$$

$$\hat{S}(p+1) = f(\hat{S}(p), U(p)) \tag{19c}$$

$$\hat{\chi}_a(p+1) = g(\hat{\chi}_a(p), U(p)) \tag{19d}$$

$$C(U^*(\cdot)) \leq 0 \tag{19e}$$

where $U^*(\cdot) = [u^*_{mm}(k), \ldots, u^*_{mm}(k+N_c-1)]$ is the computed optimal control and k represents instant $t_k = kT_s$, T_s being the prediction sampling period. f, g and $C(U^*(\cdot))$ respectively denote the prediction models and the inequality set of constraints (see next section). Once the problem is solved, only $u^*_{mm}(k)$ is applied to the robot, and the process is repeated. The previous optimization results are used to warm-start the solver.

B. The prediction models

Two prediction models f and g are needed. The first one, f, is obtained with the global and exact method used in [16]. It consists of first computing the homogeneous transformation matrix ${}^bH_c$ between F_b and F_c thanks to the forward kinematic model. Then, the exact integration of the mobile base kinematic model is used to determine matrix ${}^{b_k}H_{b_{k+1}}$ which connects two successive mobile robot poses. The prediction model for the points in camera frames is given by [16]:

$$\overline{\boldsymbol{X}}_i(k+1) = {}^{c_{k+1}}H_{b_{k+1}} {}^{b_{k+1}}H_{b_k} {}^{b_k}H_{c_k} \overline{\boldsymbol{X}}_i(k) = H(k) \overline{\boldsymbol{X}}_i(k) \tag{20}$$

where the bar indicates homogeneous coordinates. The second prediction model, g, corresponds to the integration of the robotic arm kinematic model.

C. The manipulability indices

1) Manipulator manipulability: To address the aforementioned issues of singularities and joint limits, a specific metric called w'_a [20] is proposed. This metric combines the envelope of a joint limits penalty function P with the classical manipulability index w_a, as shown by the following equation:

$$w'_a = P w_a^2 \tag{21}$$

where:

$$P = 1 - \exp\left(-k \prod_{i=0}^{5} \frac{(q_i - q_{imax})(q_{imin} - q_i)}{q_{imax} - q_{imin}}\right) \tag{22a}$$

$$w_a = \det\left(J_a^{red}(\chi_a)\, J_a^{red}(\chi_a)^T\right) \tag{22b}$$

q_{imax} and q_{imin} define the minimal and maximal joint limits and k is a positive constant. J_a^{red} is the jacobian J_a of the arm reduced to take into account only translation velocities. This reduction is needed because only 5 joints are controlled. Thus, w'_a tends to 0 when the robot comes closer to singularities or joint limits. This term has thus to be maximized.

2) Mobile manipulator manipulability: We now define a metric taking into account the mobile base, which affects the manipulability of the entire robot. From the previous reasoning, we propose the following measure:

$$w'_{b+a} = P w_{b+a}^2 = P \ \det\left(\bar{J}_{b+a}^{red}(\chi_a)\, \bar{J}_{b+a}^{red}(\chi_a)^T\right)^2 \tag{23}$$

where P is given by (22a), w_{b+a} being deduced from the entire system jacobian $\bar{J}_{b+a}^{red}$. As w'_a, this term tends to 0 when singularities and joint limits are close.

D. The visibility constraint

In the context of visual servoing, it is crucial to ensure that the target remains within the camera's field of view at all times. In the proposed framework, the visibility constraint is always set on the head image, allowing the arm to keep full freedom of movement. To achieve this, the following constraint is applied to the head image to guarantee that the visual cues do not exceed the image's limits:

$$\begin{bmatrix} S_{ip}^{c_h}(p) - S_u \\ S_l - S_{ip}^{c_h}(p) \end{bmatrix} \leq 0, \forall p \in [\![k+1, k+N_p]\!] \tag{24}$$

where S_l and S_u are respectively the lower and upper image boundaries of the head camera.

E. The joint limits constraints

Next, it is also essential that the arm joints never exceed their lower and upper boundaries χ_{al} and χ_{au} defined by the elements q_{imax} and q_{imin} which leads to the constraints:

$$\begin{bmatrix} \chi_a(p) - \chi_{au} \\ \chi_{al} - \chi_a(p) \end{bmatrix} \leq 0, \forall p \in [\![k+1, k+N_p]\!] \tag{25}$$

F. The positioning constraint set

The positioning constraint set is necessary for three reasons. First, it guarantees the closed-loop stability of the NMPC scheme. Second, it forces the realization of the positioning task to avoid a compromise with manipulability maximization. Third, it prevents the robot from being stuck in a local minimum that might appear when relying on local, and thus sub-optimal, solvers. In this section, we first present the three constraints comprised in this set, and we next detail their use.

1) The prediction-reference equality constraint: Inspired by the terminal constraint method [3], we propose a prediction-reference equality constraint. It imposes that given predicted visual features are equal to the reference ones. This is expressed in the following form:

$$\|\hat{S}(k + p_{TC}) - S^*\| = 0 \tag{26}$$

where the p_{TC} is the constrained prediction index.

2) The prediction-prediction decrease constraint: We now propose a second constraint imposing the transformation between two predicted poses to decrease. To do so, we first define $H_{p_{TC}}$ as the transformation matrix between the pose at the predicted instant $k + p_{TC} - 1$ and the one at $k + p_{TC}$. Next, we rely on the logarithmic map log_6 that allows transferring an element H of the Lie group $SE(3)$ to the corresponding element ν of its Lie algebra $se(3)$ [19]:

$$\nu = \log_6(H) \tag{27}$$

In this work, $H = H_{p_{TC}}$ is used in its homogeneous transformation matrix form and ν in its 6 dimensional motion vector form. Actually, ν corresponds to the velocity, linear and rotational, that should be applied during 1 second to obtain the transformation described by H. Thus, the constraint can be written as:

$$\|log_6(H_{p_{TC}})\| < min_{log} - \delta_{min} \tag{28}$$

where $H_{p_{TC}} = H(k + p_{TC} - 1)$, and min_{log} represents the smallest $\|\log_6(H_{p_{TC}})\|$ value observed up to the current instant. δ_{min} is introduced to force a minimum decrease, inspired by [21]. It must be large enough to speed up the convergence but small enough to let the solver focus on the tasks.

3) The velocity constraints: To provide the necessary large prediction horizon, the velocity constraints of the last inputs can be relaxed [8]. This approach leads to the following set of constraints for the mobile manipulator velocities:

$$\begin{bmatrix} u_{mm}(p) - u_{u|t} \\ u_{l|t} - u_{mm}(p) \end{bmatrix} \leq 0, \forall p \in [\![k, k+N_c-N_r-1]\!]$$
$$\begin{bmatrix} u_{mm}(p) - u_{u|r} \\ u_{l|r} - u_{mm}(p) \end{bmatrix} \leq 0, \forall p \in [\![k+N_c-N_r, k+N_c-1]\!] \tag{29}$$

N_r is the number of prediction steps with relaxed boundaries, $u_{l|t}$ and $u_{u|t}$ are respectively the lower and upper tight boundaries corresponding to the 'true' limits of the actuator, and $u_{l|r}$ and $u_{u|r}$ are respectively the lower and upper relaxed boundaries.

4) Using the constraints: At the beginning of the servoing, p_{TC} is set up to N_p. On the one hand, constraint (26) forces the last predicted features to be equal to the desired ones, guaranteeing that the task is feasible. It should be noted that this constraint can be respected thanks to constraint (29). On the other hand, constraint (28) imposes the transformation between the last two predicted poses to decrease, prioritizing the visual task and preventing the robot from being stuck in a local minimum.

p_{TC} is maintained to its current value until the logarithm becomes smaller than a threshold δ_{log}, meaning the poses predicted at instants $k+p_{TC}-1$ and $k+p_{TC}$ are close enough to be considered equal. At this moment, the current constraint does not have an impact on the optimization anymore, and the constraint configuration must be updated by applying $p_{TC} = p_{TC}-1$. This process is repeated until $p_{TC} = 1$ so that the command applied to the robot actually makes it reach the desired pose.

IV. Results

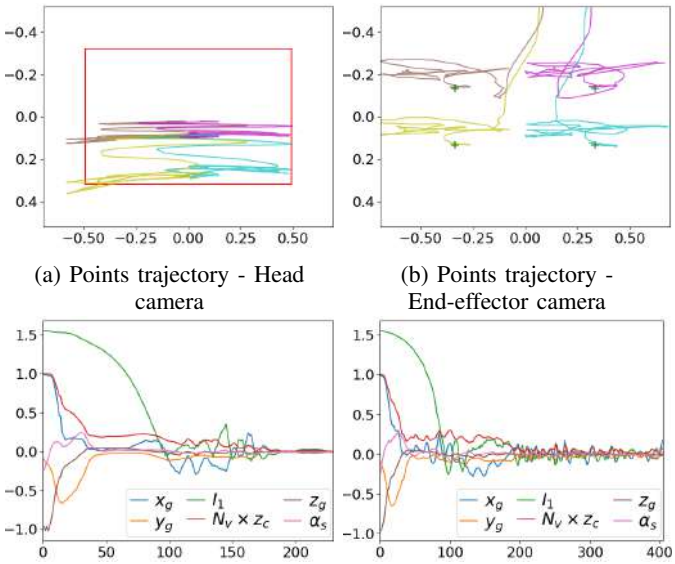
This section presents simulation results (cf. video) to evaluate the efficiency of the proposed approach. It is divided into three parts. In the first one, the presented VPC scheme is run and analyzed. Next, the scheme is tested with two different decrease constraints: one is based on the command, as in [17], and the other on the proposed logarithmic constraint to highlight its relevance. The last part analyses the influence of the manipulability measure.

All algorithms are implemented using the C++ language and the optimization problem is solved with the SLSQP solver from the NLopt package. All gradients are symbolically computed with the CasADi software [22] offline, and only evaluated online. Matrices ${}^{b}H_c$ and ${}^{b_k}H_{b_{k+1}}$ are obtained with Pinocchio [23], a rigid body dynamics library. All tests are performed on an Intel Core i7-10850H and the VPC runs at a frequency of 5Hz. The solver timeout is set to $0.15s$, N_p and N_c are fixed to 10 steps with a sampling time $T_s = 0.4s$. The target is a rectangle centered in $(3,0,1.08625)$. The camera and the mobile base have to travel about 2m to reach the target. The arm is initially tucked. The bounds on the mobile base linear and angular velocities are respectively equal to ± 0.1 m/s and ± 0.3 rad/s. The minimal and maximal joint limits are given by: $\chi_{au} = [2.68, 1.02, 1.50, 2.29, 2.07]$, $\chi_{al} = [0.07, -1.50, -3.46, -0.32, -2.07]$, $\chi_{hu} = 1.24$ and $\chi_{hl} = -1.24$. Matrix $Q_S(p)$ is the identity matrix, while $N_r = 1$. Time units of the plots are the control loop iterations.

A. Proposed scheme results

In this section, the approach is tested with the initial robot pose $(0,0,0)$. Additional results for other initial configurations are available in the supplementary video.

1) Visual task realization: Fig. 2b and 2c show that the visual task is correctly performed. Indeed, the controller successfully drives the camera to bring the interest points to their desired values (indicated by the green crosses), which is realized by vanishing the error between the image moments and their desired values. However, Fig. 2a shows that the visibility constraint may be sometimes violated, as the visual features may leave the head camera field of view. This problem is due to the optimization process which may terminate before satisfying all constraints because of the incorporated time-out. To deal with this issue, this constraint has been set up in a conservative way to avoid the loss of visual features. Finally, Figure 2d provides the results obtained without the use of the symbolic gradient computation with CasADi. This figure shows that an accurate positioning cannot be achieved.



(a) Points trajectory - Head camera

(b) Points trajectory - End-effector camera

(c) $S-S^*$ evolution

(d) $S-S^*$ without CasADi

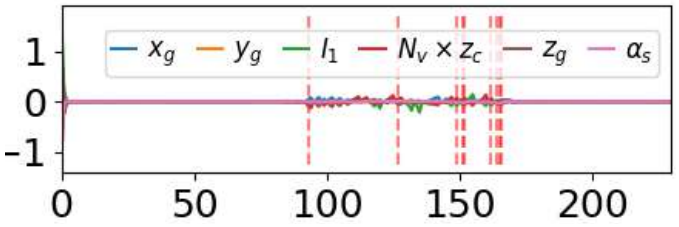
Fig. 2: Task realization results

2) Stability: Figure 3a displays the error between the $p_{TC}{}^{th}$ predicted image moments and their desired values, where a small error implies that the terminal constraint is satisfied. This figure clearly illustrates that the proposed scheme enables the satisfaction of the constraint despite the initial irrelevant arm configuration regarding the visual task and the large distance between the initial and desired poses. Red vertical lines represent the shifts realized to accelerate the convergence and are more detailed in section IV-B. Figure 3b presents again the obtained results without CasADi, and illustrates the difficulty to satisfy the terminal constraint when the arm is tucked. Indeed, it requires many iterations of the solver to calculate a solution respecting all the constraints, which cannot be done in a reasonable time ($< 200ms$) without CasADi.

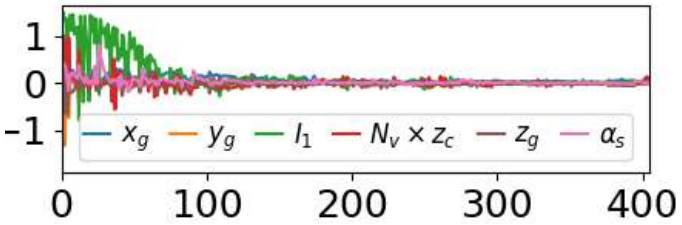
3) Joints and commands evolution: Finally, Fig. 4 shows the velocities and joint angles evolution. The values of the former remain within the given boundaries despite the use of a relaxed constraint. Concerning the joint angles, they stay away from their limits thanks to the manipulability measure.

B. Visual task convergence: Logarithmic vs command decrease constraint

To ensure the visual task convergence over the manipulability maximization, the positioning constraints set is needed.

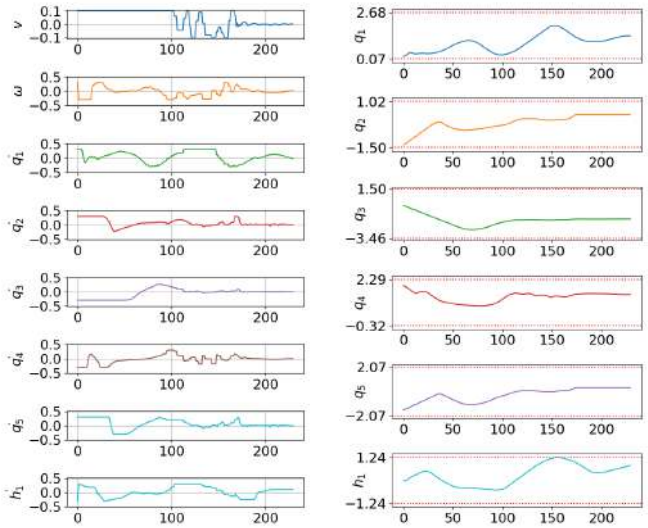


(a) Evolution of last $\hat{S}_m - \hat{S}^*_m$ with CasADi



(b) Evolution of last $\hat{S}_m - \hat{S}^*_m$ without CasADi

Fig. 3: Evolution last $\hat{S}_m - \hat{S}^*_m$

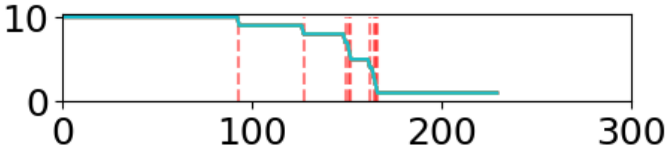


(a) Velocities evolution (b) Joint values evolution

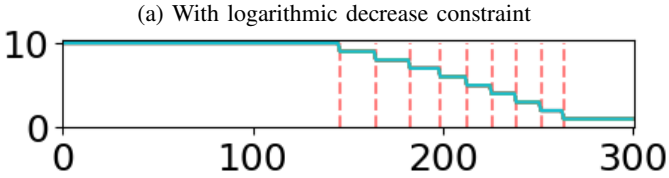
Fig. 4: Joints and commands evolution

In this section, we compare the logarithm-based method presented in this paper with the command-based one presented in [17]. To do so, we focus on the evolution of p_{TC} which quantifies the convergence rate (see Fig. 5). Indeed, the faster the value of p_{TC} is equal to zero, the faster the visual task will be completed. For both methods, the prediction-reference equality constraint is initially applied to the N_p^{th} prediction, which remains unchanged for a long time due to the input relaxation. Next, the prediction-reference equality is progressively shifted to the previous prediction until it is positioned on the first prediction, *i.e.,* $p_{TC} = 0$. However, the prediction-prediction decrease constraint being different, logarithm-based for Fig. 5a and command-based for Fig. 5b, we observe two different behaviors. In Fig. 5a, the switches are started earlier than in Fig. 5b, and the servoing is shorter. This highlights the efficiency of the logarithm-based constraint over the command-based one.

Remark: As can be seen in Fig. 3a, the prediction-reference equality constraint value is not exactly equal to zero. The chosen value to trigger the switch of the prediction-reference equality constraint has an impact on the shifting process. A smaller value smooths the transitions but increases the convergence time, while a larger one speeds up the convergence time but leads to rough changes in the optimization problem that might lead to the non-respect of the constraints.



(a) With logarithmic decrease constraint



(b) With command decrease constraint

Fig. 5: p_{TC} evolution

Figure 6 can be directly confronted to figure 2c. It shows that the visual task realization is much slower using the command decrease function.

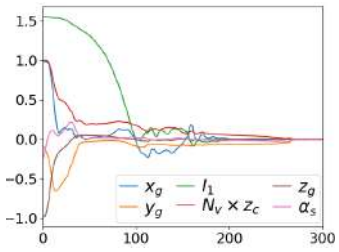


Fig. 6: $S - S^*$ evolution with command decrease constraint

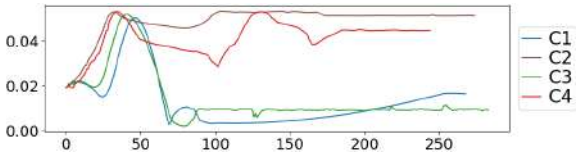
C. Manipulability measure analysis

This last section studies the influence of the manipulability measure choice. Four cases are considered:

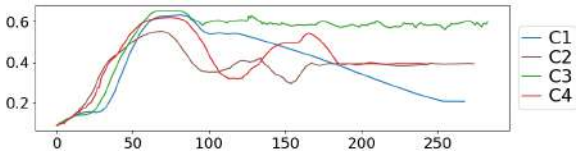
- C1: Without manipulability, i.e. $K_w = 0$
- C2: With w'_a only, i.e. $\alpha_w = 1$
- C3: With w'_{b+a} only, i.e. $\alpha_w = 0$
- C4: With w'_a and w'_{b+a}, with $\alpha_w = 0.2$

Figures 7a, 7b and 7c respectively presents the w_a, w_{b+a}, and P evolution obtained for each case. These figures clearly show that the C2 and C3 cases are indeed the scenarios where w'_a and w'_{b+a} are respectively maximized, as expected. They also demonstrate that, in the C4 case, the trade-off between both manipulabilities is reached, again as expected. It can also be noted that the evolution of w'_{b+a} in C3 and C4 leads to similar results in terms of maximization. However, Fig. 7c shows that P drops for the C3 case. It can be shown that this is due to the joint q_4 which is coming close to its limit, thus inducing convergence issues that can be observed in the video.

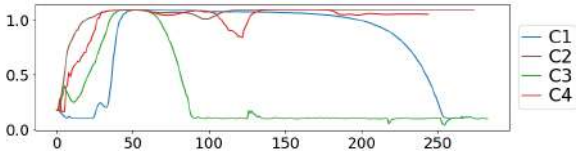
The same behavior for P can be seen for the C1 case except it occurs lately. These two results highlight the importance of keeping the term w'_a in F_w. Thus choosing $K_w = 0$ or $\alpha_w = 0$ does not appear to be the most relevant choice. Now, regarding C2 and C4 cases, it is more difficult to draw a conclusion. Indeed, performances are similar in the studied simulation scenario and a more thorough analysis should be conducted.



(a) w_a evolution



(b) w_{b+a} evolution



(c) P evolution

Fig. 7: Manipulability measures evolution

V. Conclusion

This work proposes a multi-camera VPC strategy to control a mobile manipulator. It benefits from two complementary vision systems to perform a task consisting in positioning the end-effector camera with respect to a given landmark. The proposed approach allows to deal with several challenges: (i) the large displacements which imply a large prediction horizon and question the stability, (ii) the large number of DoFs which induces a large search space when optimizing, and a high redundancy leading to possible non-suitable configurations and undesired behaviors, (iii) the processing time. It relies on a cost function depending on both the visual features and the manipulability coupled with several constraints. Among them, an original positioning constraint set allows prioritizing the vision-based task against the manipulability while avoiding local minima and guaranteeing closed-loop stability despite a large prediction horizon has been introduced. In addition, to deal with the processing time, we have also implemented the optimization problem using a symbolic representation. The strategy has been simulated using ROS and Gazebo and compared to our previous work, thus demonstrating its interest and efficiency. In the future, we plan to extend this new framework to handle the presence of obstacles.

References

[1] G. Allibert, E. Courtial, and F. Chaumette, "Predictive control for constrained image-based visual servoing," *IEEE Trans. on Robotics*, vol. 26, no. 5, pp. 933–939, October 2010.

[2] F. Chaumette and S. Hutchinson, "Visual servo control, part 1 : Basic approaches," *Robotics and Automation Mag.*, vol. 13, no. 4, 2006.

[3] L. Grüne and J. Pannek, "Nonlinear model predictive control," in *Nonlinear Model Predictive Control*. Springer, 2017, pp. 45–69.

[4] A. Paolillo, T. S. Lembono, and S. Calinon, "A memory of motion for visual predictive control tasks," in *International Conference on Robotics and Automation*, 2020.

[5] F. Fusco, O. Kermorgant, and P. Martinet, "Integrating features acceleration in visual predictive control," *IEEE Rob. and Autom. Letters*, 2020.

[6] I. Mohamed, G. Allibert, and P. Martinet, "Sampling-based mpc for constrained vision based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2021)*, 2021.

[7] K. Zhang, Y. Shi, and H. Sheng, "Robust nonlinear model predictive control based visual servoing of quadrotor uavs," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 2, pp. 700–708, 2021.

[8] A. Durand-Petiteville and V. Cadenat, "Advanced visual predictive control scheme for the navigation problem," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 2, pp. 1–21, 2022.

[9] S. Heshmati-alamdari, A. Eqtami, G. C. Karras, D. V. Dimarogonas, and K. J. Kyriakopoulos, "A self-triggered position based visual servoing model predictive control scheme for underwater robotic vehicles," *Machines*, vol. 8, no. 2, p. 33, 2020.

[10] J. Pankert and M. Hutter, "Perceptive model predictive control for continuous mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6177–6184, 2020.

[11] M. Giftthaler, F. Farshidian, T. Sandy, L. Stadelmann, and J. Buchli, "Efficient kinematic planning for mobile manipulators with non-holonomic constraints using optimal control," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3411–3417.

[12] G. B. Avanzini, A. M. Zanchettin, and P. Rocco, "Constrained model predictive control for mobile robotic manipulators," *Robotica*, vol. 36, no. 1, pp. 19–38, 2018.

[13] R. Colombo, F. Gennari, V. Annem, P. Rajendran, S. Thakar, L. Bascetta, and S. K. Gupta, "Parameterized model predictive control of a non-holonomic mobile manipulator: A terminal constraint-free approach," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2019, pp. 1437–1442.

[14] S. S. Martínez, J. G. Ortega, J. G. Garcia, A. S. Garcia, and J. de la Casa Cárdenas, "Visual predictive control of robot manipulators using a 3d tof camera," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2013, pp. 3657–3662.

[15] M. Logothetis, G. C. Karras, S. Heshmati-Alamdari, P. Vlantis, and K. J. Kyriakopoulos, "A model predictive control approach for vision-based object grasping via mobile manipulator," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018.

[16] H. Bildstein, A. Durand-Petiteville, and V. Cadenat, "Visual predictive control strategy for mobile manipulators," in *2022 European Control Conference (ECC)*, 2022, pp. 1672–1677.

[17] ——, "Multi-camera visual predictive control strategy for mobile manipulators," in *Accepted in 2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, July 2023. [Online]. Available: https://drive.google.com/file/d/1VgznusL-4HsKGYhRcrc3vmtc-aORqrBG/view?usp=share_link

[18] O. Tahri, F. Chaumette, and Y. Mezouar, "New decoupled visual servoing scheme based on invariants from projection onto a sphere," in *2008 IEEE International Conference on Robotics and Automation*, 2008.

[19] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.

[20] N. BJ and K. PK, "Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance," *The International Journal of Robotics Research*, vol. 14, pp. 255–269, 1995.

[21] P. Scokaert, D. Mayne, and J. Rawlings, "Suboptimal model predictive control (feasibility implies stability)," *IEEE Trans. Autom. Control*, p. 648–654, 1999.

[22] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[23] J. Carpentier, F. Valenza, N. Mansard *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems," https://stack-of-tasks.github.io/pinocchio, 2015–2021.

Motion Planning for Multi-legged Robots using Levenberg-Marquardt Optimization with Bézier Parametrization

David Valouch Jan Faigl

***Abstract*— This paper presents a novel formulation of motion planning for multi-legged walking robots. In the proposed method, a single-step motion is formulated as a nonlinear equation problem (NLE): including kinematic, stability, and collision constraints. For the given start and goal configurations, the robot's path is parametrized as Bézier curve in the configuration space. The resulting NLE is solved using Levenberg-Marquardt optimization implemented using a sparse matrix solver. We propose handling the trigonometric kinematic constraints with the polynomial path parametrization. A relaxation of the constraint is used while guaranteeing a desired tolerance along the planned path. Although the proposed method does not explicitly optimize any criterion, it produces high-quality paths. The method is deployed in transforming a sequence of discrete configurations produced by a step sequence planner into a valid path for a multi-legged walking robot in challenging planning scenarios where a regular locomotion gait cannot be used because of sparse footholds.**

I. Introduction

Motion planning for multi-legged robots [1], [2] can be considered challenging because of many controllable degrees of freedom. In addition to collision constraints, multi-legged robots are subject to kinematic and stability constraints [3], [4]. Moreover, the constraints change as the robot takes steps resulting in a multimodal constrained planning problem [5], [6]. Although popular approaches to motion planning are based on a random sampling of the configuration space or control space [7]–[9], optimization-based approaches have also been successfully deployed [2], [10], [11] that are reminiscent of the early motion planning using potential functions [12].

We focus on motion planning for multi-legged walking robots, such as the SCARAB II hexapod robot [13], depicted in Fig. 2a. In particular, we investigate scenarios where a correct sequence of steps is necessary for traversing challenging structured terrain with limited footholds as depicted in Fig. 3. In our previous work [14], [15], we address finding the sequence of steps to cross a gap with limited footholds. The sequence of steps is represented by discrete configurations that subsequently need to be connected with continuous motions that satisfy the required constraints. Since single-step motions do not require solving maze-like obstacle avoidance, we opt for a local optimization technique as motion planning.

The authors are with the Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Czechia. {valoudav, faiglj}@fel.cvut.cz

The presented work has been supported by the Czech Science Foundation (GAČR) under the research project No. 21-33041J.

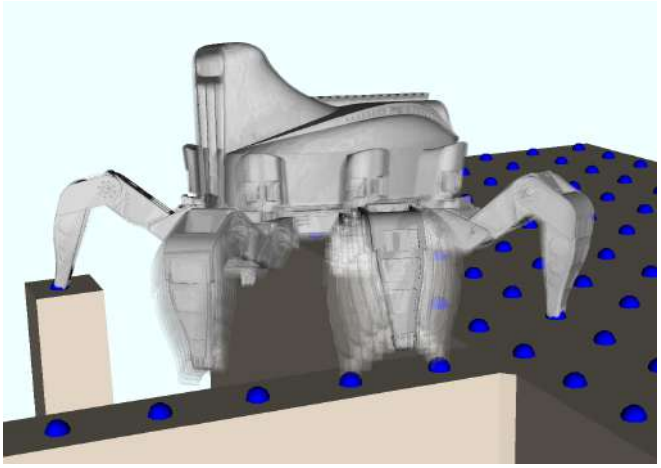


Fig. 1. Visualization of the planned motion of a legged robot.

The former work [14] is based on the Levenberg-Marquardt method to find valid configurations validating the feasibility of individual steps. In this work-in-progress report, we present an extension of the method to determine a valid path connecting two configurations while satisfying the motion constraints. Although we focused solely on constraint satisfaction, the developed motion planner yields relatively high-quality, smooth paths. Our results support a similar observation by [8] that a well-formulated simple method can produce near-optimal solutions without explicitly optimizing any criterion.

The rest of the paper is organized as follows. The studied problem is formulated in the following section. The proposed method is described in Section III. The validation results are reported in Section IV. The paper is concluded in Section V.

II. Problem Statement

Let $\mathcal{C}$ be the configuration space of the multi-legged robot. Specifically for a robot with only revolute joints, $\mathcal{C}$ can be defined as

$$\mathcal{C} = SE(3) \times SO(2)^{\text{CDOF}}\,, \tag{1a}$$

$$SE(3) \equiv \mathbb{R}^6\,,\ SO(2) \equiv \mathbb{R}\,, \tag{1b}$$

where CDOF is the number of controllable degrees of freedom. The six-legged walking robot SCARAB II has three revolute joints on each leg. Hence, with the pose of the base link, it has 24 degrees of freedom (DOF) in total, and its configuration is isomorphic with $\mathbb{R}^{24}$.

Now assume we have obtained a sequence of discrete configurations

$$q_1, q_2, \ldots, q_k \in \mathcal{C} \tag{2}$$

as a result of step-sequence planning, such as [4], [6], [14], [15]. In our case, the sequence consists of configurations in which the robot's leg switches between swinging and supporting state. Thus, the sequence represents the steps of the walking robot.

Then, for every pair of two consecutive configurations $q_i, q_{i+1} = q_{\text{start}}, q_{\text{end}}$, we aim to find a path π connecting them that can be defined as

$$\pi : [0,1] \to \mathcal{C}\,, \tag{3a}$$
$$\pi(0) = q_{\text{start}}\,,\ \pi(1) = q_{\text{end}}\,. \tag{3b}$$

The motion is subject to holonomic constraints that can be expressed as

$$f(\pi(t)) = \mathbf{0} \ :\ \forall t \in [0,1]\ , \tag{4a}$$
$$g(\pi(t)) \leq \mathbf{0} \ :\ \forall t \in [0,1]\ , \tag{4b}$$
$$g, f : \mathcal{C} \to \mathbb{R}^*\,. \tag{4c}$$

The equality constraint function f represents the kinematic constraints for the supporting legs that have to stay at the assigned footholds during the whole motion. The inequality constraint g enforces stability and collision-freeness. Note that we do not specifically define the collision-free subset $\mathcal{C}_{\text{free}}$, as collisions are covered by the inequality constraint (4b).

Further, we request the path π to be optimal under some quality criterion $\mathcal{Q} : \Pi \to \mathbb{R}$; here it is expressed using derivatives of π:

$$\mathcal{Q}(\pi) = \sum_{i=1}^{\infty} \int_{t=0}^{1} w_i \left\| \pi^{(i)}(t) \right\| dt\,, \tag{5}$$

where $w_i \geq 0$ are arbitrary weights, and $\pi^{(k)}$ represents the k-th derivative of π. The rationale behind the criterion is to minimize the length of the path: $\int \left\|\pi^{(1)}(t)\right\| dt$, and the complexity or 'roughness' of the path: contributing to the integral of the higher derivatives.

III. Planning Method

The proposed motion planning follows the problem formulation (3–5). We present the used parametrization of the path (3). Then, we formulate the utilized representation of the constraints (4). Finally, we describe the proposed algorithmic solution to the problem. Although the method is applied to a particular SCARAB II robot [13], which determines the used motion constraints, the method can be adapted to other planning scenarios with different robots.

A. Bézier Curve Parametrization

The proposed path π representation is based on Bézier curve [16] already used for robot motion parametrization [17], [18]. Bézier curve of the degree d is parametrized by $d+1$ control points $P = (p_0, p_1, p_2, \ldots . p_d)$. Each point on the curve is a linear combination of the control points

$$\pi_d(P,t) = \sum_{i=0}^{d} \mathrm{B}_i^d(t)\, p_i\,, \tag{6}$$

where B_i^d is the i-th Bernstein polynomial of the degree d. In our case, the control points are vector representations of the robot's configurations $P = (q_{\text{start}}, q_1, \ldots, q_{d-1}, q_{\text{end}})$.

Further, we use the property that for any Bézier curve, an equivalent Bézier curve of a higher degree can be constructed as

$$\pi_d(P_d,t) = \pi_{d+1}(P_{d+1},t)\ ;\ \forall t \in [0,1]\ , \tag{7a}$$
$$P_{d+1} = (p_0, \tilde{p_1}, \ldots, \tilde{p_d}, p_d)\ , \tag{7b}$$
$$\tilde{p_i} = \frac{i}{d+1} p_{i-1} + \left(1 - \frac{i}{d+1}\right) p_i\ ;\ \forall i \in 1 \ldots d\,. \tag{7c}$$

Here, it is worth noting that the degree of Bézier curve can be used to limit the path complexity, an important rationale behind the proposed planning algorithm.

B. Constraints

The constraints considered for planning the steps of the SCARAB II walking robot are as follows.

1) The kinematic constraint ensures that the legs in the support phase remain in their assigned footholds during the planned step. The constraints function is formulated using the forward kinematics of the supporting legs using standard methods [19].
2) Stability constraint ensures the stability of the robot that we constrain the robot's center of mass to be above the support polygon, defined as the convex hull of the footholds. The constraint is a simple affine inequality constraint.
3) Collisions are distinguished as self-collisions and collisions with the terrain. For the SCARAB II robot, the motion ranges of the joints avoid self-collisions. The collisions with the terrain are modeled using a signed distance field (SDF) [20]. In particular, using implementation [21]. A simplified collision model consisting of spherical primitives is used (see Fig. 2b) to efficiently check robot collisions using the SDF. A sphere is in a collision if and only if the signed distance of its center is less than its radius. The collision constraint is relaxed around the footholds as in [14], allowing the foot tips to reach the terrain.

(a) The SCARAB II robot (b) Collision model

Fig. 2. The SCARAB II robot used for the evaluation of the proposed method, and its collision model using sphere primitives.

Note that, in our implementation, the inequality constraint (4b) is replaced by an equality constraint

$$\tilde{g}(\pi(t)) = \mathbf{0}\ ;\ \forall t \in [0,1]\ , \tag{8a}$$
$$\tilde{g}(q) = \min(g(q), 0)\,. \tag{8b}$$

C. Path Constraint

The constraints (4a) and (8a) are used to form a path constraint

$$\mathcal{F}(\pi) = \int_{t=0}^{1} \|f(\pi(t))\|^2 + \|\tilde{g}(\pi(t))\|^2 \, dt = 0\,. \quad (9)$$

Computing (9) directly would be computationally intractable. However, in practice, the constraint functions are "well-behaved", with bounded derivatives. Therefore, the integral is approximated by a Riemann sum with a uniform partition of the size N:

$$\mathcal{F}(\pi) \approx \sum_{i=0}^{N} \|f(\pi(t_i))\|^2 + \|\tilde{g}(\pi(t_i))\|^2 \;,\; t_i = \frac{i}{N}\,. \quad (10)$$

It is clear that (10) is 0, if and only if $\|f\|$ and $\|\tilde{g}\|$ are both 0 in all sample points, which is exploited in the proposed planning algorithm.

The function f represents the kinematic constraint and thus contains trigonometric terms. Therefore, a curve represented by a polynomial spline, such as Bézier curve, cannot fully satisfy it. Hence, we allow a tolerance $\|f\| \leq \varepsilon$ for a small $\varepsilon > 0$ to ensure compatibility with Bézier curve parametrization. It is achieved by introducing slack vectors ξ_i such that

$$\tilde{f}(\pi(t_i), \xi_i) = f(\pi(t_i)) - \xi_i = \mathbf{0}\,, \quad (11a)$$

$$\|\xi_i\| \leq \varepsilon\,. \quad (11b)$$

The inequality (11b) is included in $\tilde{g}$ in the same way as the inequality constraint (4b) is included in (8b).

Now, we can formulate the constraint for the Bézier parametrized path with the slack ε and sampling N as

$$\tilde{\mathcal{F}}_{\varepsilon,N}(P, \boldsymbol{\xi}) = \begin{bmatrix} \tilde{f}(\pi(P, t_1), \xi_1) \\ \vdots \\ \tilde{f}(\pi(P, t_N), \xi_N) \\ \tilde{g}_\varepsilon(\pi(P, t_1), \xi_1) \\ \vdots \\ \tilde{g}_\varepsilon(\pi(P, t_N), \xi_N) \end{bmatrix} = \mathbf{0}\,. \quad (12)$$

D. Levenberg-Marquardt Method

Levenberg-Marquardt (LM) method, also called *damped least squares method* [22], is a more numerically stable version of the Newton-Raphson method for finding roots of nonlinear equations. It is used to solve nonlinear systems of equations in the form $h(\boldsymbol{\theta}) = \mathbf{0}$ by iteratively improving an initial guess for $\boldsymbol{\theta}$ by a step $\Delta\boldsymbol{\theta}$:

$$\Delta\boldsymbol{\theta} = (J_h(\boldsymbol{\theta})^T J_h(\boldsymbol{\theta}) + \lambda I)^{-1} J_h(\boldsymbol{\theta})^T h(\boldsymbol{\theta})\,, \quad (13)$$

where J_h is the Jacobian of h and $\lambda > 0$ is the damping parameter. The process is repeated until the solution error $\|h\|$ is reduced below numeric precision, $\|h\| < \epsilon_{\text{num}}$. Note that (13) minimizes the criterion

$$\|J_h(\boldsymbol{\theta})\Delta\boldsymbol{\theta} - h(\boldsymbol{\theta})\| + \sqrt{\lambda}\,\|\Delta\boldsymbol{\theta}\|\,. \quad (14)$$

Furthermore, for $\lambda \to 0$, the LM method becomes equivalent to the Newton-Raphson method using the pseudo-inverse of the Jacobian.

The choice of the damping parameter λ regulates the convergence speed and numeric stability of the method. A dynamic damping strategy [23] is used, which updates λ at each iteration for better performance. The damping is reduced and increased by multiplying it by constants $\lambda_{\text{drop}} \in (0, 1)$ and $\lambda_{\text{boost}} > 1$, respectively. The LM algorithm is described in the SolveLM subroutine of Algorithm 1.

E. Planning Algorithm

The proposed step planning is summarized in Algorithm 1. It finds a path parametrized by the Bézier control points P satisfying the constraint $\tilde{\mathcal{F}}_{\varepsilon,N}$ for the given ε and N. Note that the path quality criterion (5) is tackled indirectly as we aim for a similar effect as in [8], where a "cost-indifferent" sampling-based path-planning algorithm produced paths of competitive quality to cost-aware algorithms. The proposed planning algorithm works as follows.

Algorithm 1: Plan Step

PlanStep($q_{\text{start}}, q_{\text{end}}, \varepsilon, N, d_{\max}$)
$\mathcal{P} \leftarrow (q_{\text{start}}, q_{\text{end}})$
for $d \leftarrow \{2, \ldots, d_{max}\}$ **do**
  $\mathcal{P} \leftarrow P_d$ (7) s.t. $\pi_d(P_d, t) = \pi_{d-1}(P, t)\ ;\ \forall t \in [0, 1]$
  $\boldsymbol{\xi} \leftarrow \mathbf{0}$
  $\mathcal{P}, \boldsymbol{\xi} \leftarrow \text{SolveLM}(\tilde{\mathcal{F}}_{\varepsilon,N}, \mathcal{P}, \boldsymbol{\xi})$
  if $\left\|\tilde{\mathcal{F}}_{\varepsilon,N}(\mathcal{P}, \boldsymbol{\xi})\right\| \leq \epsilon_{num}$ **then**
    END SUCCESS – **return** $\mathcal{P}$
END FAILURE

SolveLM($\tilde{\mathcal{F}}, \mathcal{P}, \boldsymbol{\xi}$)
$\lambda \leftarrow \lambda_{\text{init}}$
$\mathcal{P}^\star, \boldsymbol{\xi}^\star \leftarrow \mathcal{P}, \boldsymbol{\xi}$
for $i \leftarrow \{1, \ldots, \text{MaxIt}\}$ **do**
  $\Delta\mathcal{P}, \Delta\boldsymbol{\xi} \leftarrow (J_{\tilde{\mathcal{F}}}^T J_{\tilde{\mathcal{F}}} + \lambda I)^{-1} J_{\tilde{\mathcal{F}}}^T \tilde{\mathcal{F}}(\mathcal{P}, \boldsymbol{\xi})$
  if $\left\|\tilde{\mathcal{F}}(\mathcal{P}^\star + \Delta\mathcal{P}\,,\, \boldsymbol{\xi}^\star + \Delta\boldsymbol{\xi})\right\| < \left\|\tilde{\mathcal{F}}(\mathcal{P}^\star, \boldsymbol{\xi}^\star)\right\|$ **then**
    $\lambda \leftarrow \lambda\lambda_{\text{drop}}$
    $\mathcal{P}^\star, \boldsymbol{\xi}^\star \leftarrow \mathcal{P} + \Delta\mathcal{P}\,,\, \boldsymbol{\xi} + \Delta\boldsymbol{\xi}$
  else
    $\lambda \leftarrow \lambda\lambda_{\text{boost}}$
  if $\left\|\tilde{\mathcal{F}}(\mathcal{P}^\star, \boldsymbol{\xi}^\star)\right\| < \epsilon_{num}$ **then**
    BREAK **for**
return $\mathcal{P}^\star, \boldsymbol{\xi}^\star$

The algorithm starts with a naive linear interpolation $P = (q_{\text{start}}, q_{\text{end}})$. We increase the degree d of Bézier curve at each step as in (7). The current P is then used as an initial guess for the solution to $\tilde{\mathcal{F}}_{\varepsilon,N}(P, \boldsymbol{\xi}) = \mathbf{0}$ in the LM method. If the constraint is satisfied, we find a valid path; otherwise, we continue with the increased d. By iterative increasing

d, the procedure ensures we minimize the degree of Bézier parametrization. Bézier curve of the degree d satisfies

$$\pi^{(i)} = \mathbf{0} \; ; \; \forall i \geq d \, . \tag{15}$$

Thus, we eliminate unnecessary terms from the path quality criterion (5).

Also, note that the LM algorithm only accepts steps that improve the objective. Even if the final iteration does not fully satisfy the constraint, we obtain a local optimum for the current d. The LM method minimizes the constraint function "*efficiently*," without unnecessary changes to the parameters. The pseudo-inverse method used in (13) produces the smallest step minimizing (14), and the term $\lambda \|\Delta\boldsymbol{\theta}\|$, in the criterion (14), further penalizes the change in the parameters. Our intuition is that, in the d-th iteration, Algorithm 1 makes a minimal necessary change mostly to the d-th derivative $\pi^{(d)}$ of the path. Therefore, it is indirectly optimizing (5). The resulting performance of the proposed method is reported in the following section.

IV. Results

The proposed planning method has been validated on a testing scenario with sequences of configurations generated by our step-sequence planner [14]. A sequence is planned for a challenging scenario, illustrated in Fig. 3, which is selected for demonstrating the ability of the planner to plan the individual steps. The robot is requested to cross a wide gap in the terrain using only a narrow beam and a single additional support. The modeled constraints are derived for the SCARAB II robot [13] depicted in Fig. 2a.

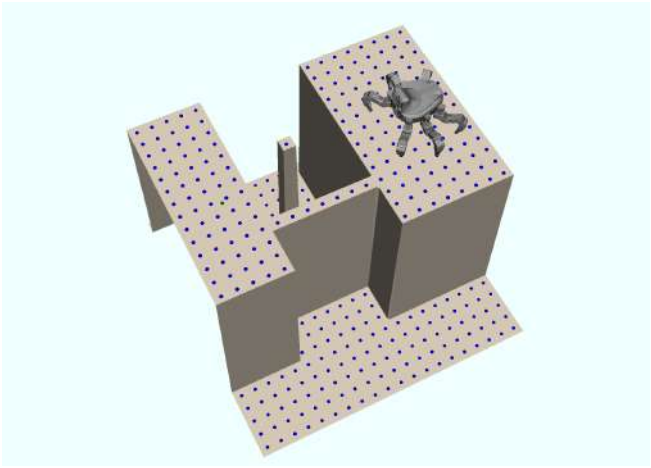


Fig. 3. A gap-crossing evaluation scenario, where only a singular pillar and a narrow beam can be used to cross the wide gap.

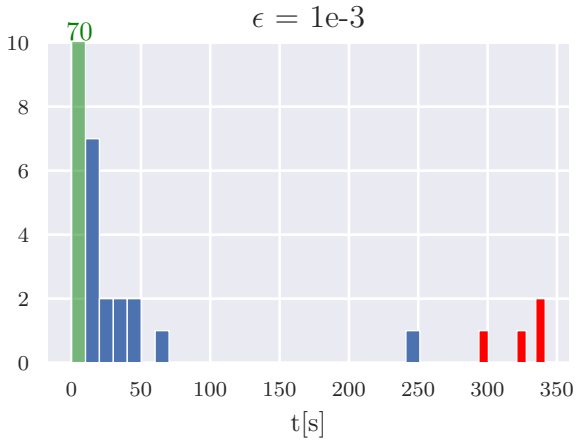
The proposed algorithm has been implemented in C++, compiled with GNU C++ compiler v9.4.0 with `-O3` level optimization. For computing the collision function, an implementation of the SDF provided with the *GridMap* library [21] is used. The Jacobian of the path constraint is constructed as a sparse matrix in the compressed column format using the *Eigen 3* linear algebra library [24]. The inversion of the matrix $J^T J + \lambda I$ in (13) is computed using LL^T Cholesky decomposition provided in *Intel® MKL Pardiso* sparse solver [25]. The method has been run on a computer with the Intel® i7-10700 processor running at 4.8 GHz and 64 GB RAM.

TABLE I

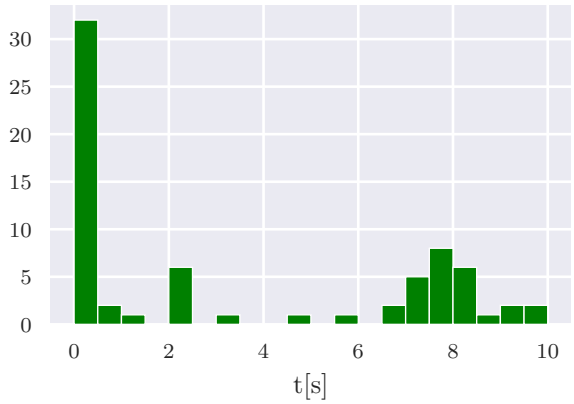
Parameters of the planning algorithm.

N	100
$d_{\max}$	10
`MaxIt`	100
λ_{init}	1
λ_{drop}	0.1
λ_{boost}	1.5
ε	1×10^{-3} m

The chosen values of the algorithm parameters are summarized in Table I. The selection of the values of the damping strategy parameters $\lambda_{\text{init}}, \lambda_{\text{drop}}, \lambda_{\text{boost}}$ is based on [23]. Values of the other parameters were chosen based on our practical experience.



(a) Histogram of the overall planning times; runs marked red failed to find a path satisfying the given tolerances. The green column represents runs under 10 s shown in detail in Fig. 4b.



(b) Detailed histogram of the planning times under 10 s

Fig. 4. Histograms of planning times for $\varepsilon = 10^{-3}$.

The results are summarized in Fig. 4 and Table II. In total, 89 steps have been planned. Most results have been obtained in less than 50 s, with most results in less than 10 s. The results found under 10 s are plotted in a separate histogram depicted in Fig. 4b to show the details of their distribution.

TABLE II
SUMMARY OF RESULTS

Steps planned	89
Steps succeeded	85
Worst constraint violation	$1.73 \times 10^{-3}\,\mathrm{m}$
Average planning time	$24.3\,\mathrm{s}$
Median planning time	$6.7\,\mathrm{s}$
Average, $t < 50\,\mathrm{s}$	$6.7\,\mathrm{s}$
Average, $1\,\mathrm{s} < t < 50\,\mathrm{s}$	$11.1\,\mathrm{s}$

The concentration of results near $0\,\mathrm{s}$ is caused by the steps that can be solved within the desired tolerance by a simple linear interpolation of $q_{\text{start}}, q_{\text{end}}$.

The average achieved planning time is $24.3\,\mathrm{s}$; however, the median planning time is only $6.7\,\mathrm{s}$, which corresponds to the average time if outliers above $50\,\mathrm{s}$ are excluded. Also, excluding the ultra-short planning times shorter than $1\,\mathrm{s}$, we obtain an average planning time $11.1\,\mathrm{s}$ for most non-trivial steps. It can be highlighted that the worst constraint violation across all configurations on a planned path is $1.73 \times 10^{-3}\,\mathrm{m}$, including the failed attempts. That is below the real-world accuracy of the robot's mechanical precision and the utilized localization using only the onboard sensors. We, therefore, believe that with further optimization, and relaxation of the tolerances, the method is viable for practical deployments.

An example of the generated motion is visualized in Fig. 1. Note that the path taken by the robot's leg is not produced by any motion primitive. It results from the numeric solver satisfying the collision constraint with a $2\,\mathrm{cm}$ collision margin locally relaxed around the footholds [14]. The motion results from satisfying the constraints by iteratively increasing the degree of the polynomial representation and alternating with a run of the LM solver.

V. CONCLUSION

We present our work-in-progress on optimization-based motion planning that targets single-step motions for multi-legged walking robots. The planner produces paths fitted to the submanifold defined by the kinematic constraint of the supporting legs up to a defined tolerance while satisfying stability and collision-freeness criteria. The method has been validated in a challenging planning scenario to connect a sequence of discrete preplanned configurations. Even with a relatively tight tolerance of $1\,\mathrm{mm}$, the method successfully finished in 85 out of 89 cases. The worst constraint violation in the remaining four cases remained under $2\,\mathrm{mm}$. Besides, the motions are planned with the median planning time of $6.7\,\mathrm{s}$. The method produces smooth natural looking motions without using any motion primitives. Thus, based on the reported results, the proposed optimization-based planning is a viable approach to motion planning for multi-legged walking robots.

REFERENCES

[1] D. Belter, P. Łabęcki, and P. Skrzypczyński, "Adaptive motion planning for autonomous rough terrain traversal with a walking robot," in *Journal of Field Robotics*, vol. 33, no. 3, 2016, pp. 337–370.

[2] A. Winkler, D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 1560–1567, 2018.

[3] Z. Kingston, M. Moll, and L. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual review of control, robotics, and autonomous systems*, vol. 1, pp. 159–185, 2018.

[4] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 897–915, 2010.

[5] Z. Kingston, A. Wells, M. Moll, and L. Kavraki, "Informing multi-modal planning with synergistic discrete leads," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[6] A. Escande, A. Kheddar, and S. Miossec, "Planning contact points for humanoid robots," *Robotics and Autonomous Systems*, vol. 61, no. 5, pp. 428–442, 2013.

[7] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[8] R. Luna, M. Moll, J. Badger, and L. E. Kavraki, "A scalable motion planner for high-dimensional kinematic systems," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 361–388, 2019.

[9] Z. Kingston, M. Moll, and L. Kavraki, "Decoupling constraints from sampling-based planners," in *Robotics Research*, 2020, pp. 913–928.

[10] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[11] B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. Lane, "Timed-Elastic Bands for Manipulation Motion Planning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 3513–3520, 2019.

[12] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[13] M. Forouhar, P. Čížek, and J. Faigl, "SCARAB II: A small versatile six-legged walking robot," in *5th Full-Day Workshop on Legged Robots at IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1–2.

[14] D. Valouch and J. Faigl, "Gait-free planning for hexapod walking robot," in *European Conference on Mobile Robots (ECMR)*, 2021, pp. 1–8.

[15] ——, "Caterpillar heuristic for gait-free planning with multi-legged robot," *IEEE Robotics and Automation Letters (RA-L)*, 2023. [Online]. Available: https://doi.org/10.1109/LRA.2023.3293749

[16] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1995.

[17] A. A. Saputra, N. N. W. Tay, Y. Toda, J. Botzheim, and N. Kubota, "Bézier curve model for efficient bio-inspired locomotion of low cost four legged robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4443–4448.

[18] S. Tonneau, "Convex strategies for trajectory optimisation: application to the polytope traversal problem," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 3335–3340.

[19] F. C. Park and K. M. Lynch, *Modern Robotics Mechanics, Planning and Control*. Cambridge University Press, 2017.

[20] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012.

[21] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, A. Koubaa, Ed., 2016, ch. 5.

[22] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[23] M. Lampton, "Damping–undamping strategies for the levenberg–marquardt nonlinear least-squares method," *Computers in Physics*, vol. 11, no. 1, pp. 110–115, 1997.

[24] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," https://eigen.tuxfamily.org, 2010, accessed: 2023-05-01.

[25] Intel®, "oneAPI Math Kernel Library," http://intel.com, 2022, accessed: 2023-05-01.

Social APF-RL: Safe Mapless Navigation in Unknown & Human-Populated Environments

S. Batuhan Vatan [1,2], Kemal Bektaş[3] and H. Işıl Bozma[1]

***Abstract*— Safe mapless navigation of mobile robots in unknown and human-populated areas is integral for increasing their usage in our daily lives. In this paper, we consider how such a behavior can be exhibited by a mobile robot and introduce Social APF-RL (Artificial Potential Functions with Reinforcement Learning). Social APF-RL extends our previously presented approach APF-RL in which the strengths of artificial potential functions (APF) with deep reinforcement learning are combined so that the robot learns how to adjust the input parameters of the APF controller. With Social APF-RL, the model is extended to accommodate the presence of humans and to respect their comfort zones while navigating. Our experimental results including both simulation and real-life scenarios demonstrate that differing from the classical navigation methods or social navigation methods, the robot can navigate successfully on its own even in complex scenarios with moving entities while maintaining social distance to humans encountered. Hence, it has better applicability in real-life scenarios. For future work, we plan to use the proposed approach in human following while adhering to social distance norms.**

I. Introduction

This paper is focused on social navigation. In social navigation, the goal is to make robots navigate like a human and not to intrude the personal or intimate space of people around unless direct interaction with them is intended [1], [2]. Thus, they navigate while taking human presence into account. Having such an ability has become important as mobile robots are increasingly being used in different sectors such as service or health in which they are required to operate in human-populated environments such as homes, cafeterias or hospitals and to share common workspaces with humans. While dynamic environments are already challenging for mobile robots, human presence further exacerbates the difficulty. In this case, merely finding one's way to the goal while avoiding collisions does not suffice. Rather, the presence of humans requires novel approaches that also consider the constraints of human comfort and social rules in addition to the environment structure [3].

In this paper, we propose Social APF-RL to address this problem. The robot is assumed to be endowed with a 2-D laser sensor and relies only on the sensed data. We consider a robot that can detect the humans in its surroundings and track them spatially. It is given a goal location and is assumed to know its relative pose to the target as given by a third-party localization module. Our proposed method builds upon the APF-RL (Artificial Potential Functions with Reinforcement Learning) method originally developed in [4] that combines the strengths of artificial potential functions (APF) with deep reinforcement learning (DRL) for safe and mapless navigation in unknown environments. Its motivation is that in contrast to previous work in which the parameters are typically manually tuned, in APF-RL, they are defined to be functions that are learned through experience. Social APF-RL extends this approach so that the robot additionally respects the comfort zones of the humans while navigating. This is achieved by introducing four major changes to APF-RL - namely internal representation of humans, DRL network input, learned parameters and reward function. We exploit the modeling of all obstacles as ellipses and use this representation to encode the social constraints regarding the humans detected. Our key insight is to modify the ellipse model of any human detected based on his/her detected velocity and feed this knowledge to the network. For deep reinforcement learning, we use soft actor-critic algorithm [5] since it is known both to be stable and to have a state-of-art performance in a range of continuous control benchmarks. Our experimental results demonstrate that Social APF-RL can be directly transferred from simulation to on-robot tasks in unknown real-world environments without any fine-tuning or data collection in the real world. While in deployment, the robot can then automatically determine the inputs to its APF controller depending on its sensory observations. To summarize, our key contributions are:

- A social navigation method that combines artificial potential functions and deep reinforcement learning to robustly maneuver the robot in unknown, possibly cluttered and human populated environments while maintaining distance to humans detected,
- Through simulations and experiments on a mobile robot, we demonstrate that our approach has high success rates in reaching the goal locations while respecting human distance in dynamic environments. A comparative study with related work shows the proposed approach to have higher success rates in complex static and dynamic environments.

The outline of this paper is as follows: Firstly, related literature is summarized in Section II. Following, the proposed Social Apf-RL method will be detailed in Section IV. The experimental study including both simulation and real robot results are discussed in Section V. Finally, Section VI concludes the paper by summarizing the paper along with a discussion of future work.

[1]Intelligent Systems Laboratory, Electrical & Electronics Engineering, Boğaziçi University, Istanbul, Turkey.
[2]Systems and Control Engineering, Boğaziçi University sbvatan@hotmail.com
[3] Formerly affiliated with Systems and Control Engineering, Boğaziçi University

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

II. Related Literature

Robots that work in human-populated areas and interact with people need to have socially compliant navigation. Proxemics define zones for interpersonal distances for human comfort [6]. These zones are classified depending on their proximity to the human: Intimate: (0 - 45 cm); personal: (45 - 120 cm), social: (1.2 - 3.6 m) and public: 3.6 - 7.6 m. While intimate or personal zones generally imply close interactions with the robot, typical robotic navigation is associated social or public zones [3].

Many works in this area mostly rely on Social Force Model [7]. In the social force model, the movement of a pedestrian is defined through the sum of attractive and repulsive forces. Attractive forces are based on the pedestrian's target position and speed, while repulsive forces are applied by obstacles and other humans around. This approach has been extended as to better represent person–person, object–person and robot–person interactions [8], [9]. As they are prone to get stuck in local minima, behavioral guarantees hold only simpler environmental settings (i.e. convex world).

Alternatively, deep learning-based approaches have been developed. These work differ with respect to the learning method and the inputs their method use. For learning, they commonly use either reinforcement learning (RL) or inverse reinforcement learning (IRL). The difference between RL and IRL methods lies on the design of the reward function. While the former uses a hand-crafted reward function, the latter is designed using expert demonstrations. A method that combines long short term memory (LSTM) with deep RL is proposed in [10]. An attention mechanism is used to increase performance of a deep RL approach [11]. The problem is divided into ego safety and social safety and is solved through using an end-to-end deep RL method [12]. A hybrid method that addresses both robot freezing and socially compliant navigation based on RL is presented in [13]. IRL with different feature sets has been used to achieve socially compliant navigation [14]. Bayesian IRL is used to learn socially normative robot navigation behaviors [15]. A multi-agent collision avoidance algorithm that exhibits socially compliant behaviors is proposed in [16]. An imitation learning-based method uses a pedestrian trajectory data set to obtain human-like movement [17].

Related works also differ regarding their inputs. Some methods require human position and velocities as input while others only require laser data. The former methods take advantage of the recent progress in human detection [16], [10], [11]. Some utilize classical image descriptors like histogram of gradients (HOG) while more recent work uses deep learning methods such as YOLO (You Only Look Once) [18]. YOLO and most of the other methods output only bounding boxes in image space, but there are also methods that directly output spatial masks directly [19], [13], [17], [12].

A third approach is based on combining classical navigation with deep learning in order to obtain the best of both worlds. In this aspect, our previously presented APF-RL method combines the strengths of artificial potential functions (APF) with deep reinforcement learning. However, the resulting navigation behavior is not socially-compliant since there is no special treatment of humans and the resulting behavior doesn't take human comfort and social rules into account.

III. APF-RL

Consider a differential drive robot with radius ρ, position function $c : \boldsymbol{R} \to \mathcal{C}$ and heading function $\alpha : \boldsymbol{R} \to S^1$ where $\mathcal{C} \subset \boldsymbol{R}^2$ defines the free workspace, $c = \begin{bmatrix} c_1 & c_2 \end{bmatrix}^T$ denotes the planar coordinates and $S^1 = [0, 2\pi]$. Let $c(0) \in \mathcal{C}$ be its initial location and $g^* \in \mathcal{C}$ be the goal. The robot is endowed with a 2-D laser rangefinder and RGB-D camera and senses its surroundings continuously using the incoming data. Its dynamics are defined as: $\begin{bmatrix} \dot{c} & \dot{\alpha} \end{bmatrix}^T = J(c)u$. Here, $J(c)$ is the velocity Jacobian matrix and $u : \boldsymbol{R} \to \boldsymbol{R}^2$ is the velocity control input consisting of linear velocity u_1 and angular velocity u_2. The velocity control u is obtained from the negative gradient of an artificial potential function $\varphi : \mathcal{C} \times \boldsymbol{R} \to [0, 1]$. Its formulation is based upon prior work [20]:$\varphi(c) \triangleq \sigma_d \circ \sigma \circ \hat{\varphi}(c)$ where $\hat{\varphi}$ is a time-varying function consisting of three terms:

$$\hat{\varphi}(c) \triangleq \frac{\gamma^k(c)}{\beta(c)} \tag{1}$$

and $\sigma : [0, \infty) \to [0, 1]$, $\sigma(x) = \frac{x}{1+x}$, and $\sigma_d : [0, 1] \to [0, 1]$, $\sigma_d(x) = x^{\frac{1}{k}}$ are used to make $\hat{\varphi}$ admissible. The numerator term encodes the distance to current goal $g \in \mathcal{C}$ as measured by $\gamma(c)$ and weighted by $k-$parameter. The goal distance function $\gamma : \mathcal{C} \to \boldsymbol{R}^{\geq 0}$ is defined as: $\gamma(c) = (c - g)^T(c - g)$. The denominator term $\beta(c) \geq 0$ encodes the time-varying set $\mathcal{O}(t)$ of obstacles and is defined so that as the robot approaches an obstacle $o \in \mathcal{O}(t)$, the potential $\hat{\varphi}$ approaches to infinity - namely $\beta(c) \to 0$. The obstacles depend on the environment. In mapless or dynamic environments, as it is not possible to know them a priori, they are detected based on the incoming sensory data. Each detected obstacle (static or dynamic) is represented by an ellipse. Navigation continues until the robot reaches its goal ($|c - g| < \tau_p$) or control input is zero ($u = 0$). The τ_p parameter is the proximity threshold. Thus, the construction of APF φ is associated with two parameters: $k-$parameter and current goal g. The $k-$parameter designates the weight of the goal distance $\gamma(c)$ with respect to the obstacle function $\beta(c)$. Thus, it determines the balance between goal attraction and obstacle avoidance. We also consider intermediate goal locations $g \in \mathcal{W}$ that lead to the ultimate goal position.

Deep reinforcement learning is used to learn k-parameter function $k(t)$ and the goal function $g(t)$. For this, we use soft actor-critic algorithm (SAC). SAC is a maximum entropy reinforcement learning framework in which two networks are trained as policy learning and value function evaluation is done separately by actor and critic networks respectively. The former decides which action to take while the latter tells the actor how good the action was and how it should adjust. Its

advantages include are: sample efficiency [21], [22], [23]; robust performance in many continuous control tasks [24] and lower computational requirements [25]. In this framework, the learning process is associated with a partially-observable Markov decision process (POMDP) $(\mathcal{S}, \Omega, \mathcal{A}, \mathcal{P}, r)$ where $\mathcal{S}$ is the state space that defines the world including the robot and Ω denotes the observation space with each observation $\omega \in \Omega$ consisting of current and previous range data $q(t)$ and $q(t-\delta t)$ and the respective goal positions - namely $\omega(t) = (q(t), q(t-\delta t), g(t), g(t-\delta t))$ where δt is the time-step of processing. The radius of the robot ρ is subtracted from the laser data, so the trained model can be transferred to other robots. Each action $a \in \mathcal{A}$ in the action space $\mathcal{A} \subset \boldsymbol{R}^3$ is defined by the 3-tuple $a = \begin{bmatrix} k & g^T \end{bmatrix}$: i) $k \in \mathcal{K}$ defines the k-parameter and $\mathcal{K} \subset \boldsymbol{R}^{>0}$ defines range for the k-parameter function; and ii) $g \in \mathcal{W}$ corresponds to an intermediate goal location in the region $\mathcal{W} \subset \mathcal{C}$ that denotes the admissible region of intermediate goals as defined by $\tau_x \times 2\tau_y$ region. It is set to be in front of the robot to encourage the learning agent to move forward. $\mathcal{P}$ is the state transition model or dynamics of the system and it is encoded in the physics simulator or implicit in the real world.

During training, the reward is the only feedback given to the learning robot. Let $r : \mathcal{A} \times \Omega \rightarrow [R_{min}, R_{max}]$ denote the reward function with the parameters R_{min}, R_{max} corresponding to the minimum and maximum reward respectively. It is defined based on the proximity of the robot to the goal location as well as the obstacles sensed based on the observation $\omega(t)$:

$$r\left(a(t); \omega(t)\right) = r_g\left(a(t), \omega(t)\right) + r_o\left(a(t), \omega(t)\right) \quad (2)$$

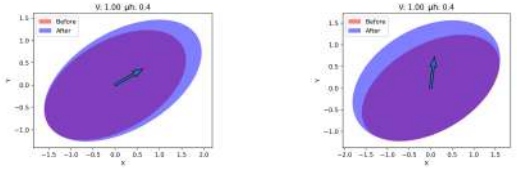
The first term r_g considers the goal:

$$r_g\left(a(t), \omega(t)\right) = \begin{cases} R_{g1} & \text{if } \delta(t) < \tau_p \\ R_{g2}\dot{\delta}(t) & \text{otherwise} \end{cases} \quad (3)$$

A big reward is given when the robot approaches within τ_p-neighborhood of goal and small incremental rewards are given with each time step for getting closer to the goal as measured by $\dot{\delta(t)}$ where $\delta(t) = |c(t) - g(t)|$. Incremental rewards are positive when the robot gets closer to the goal and negative if it moves away. Thus, the robot is encouraged to move toward the goal locations. In practice, we set $R_{g1} = 10$ and $R_{g2} = 2$. The second term r_o penalizes the robot for collisions and getting close to obstacles:

$$r_o\left(a(t), \omega(t)\right) = \begin{cases} R_{o1} & \exists o \in \mathcal{O}(t) \text{ s.t. } 0 < \beta(c, o) < \tau_o \\ R_{o2} & \text{if } \beta(c) = 0 \end{cases}$$

Closeness to each obstacle $o \in \mathcal{O}(t)$ is measured by the function $\beta(c, o) \geq 0$. A safety zone τ_o is defined around each obstacle. Regions around the obstacles are deemed risky as localization errors or a wrong move may result in collisions. The collision penalty is large since safe navigation is critical. Thus, the robot is encouraged to follow a safe path. In practice, we set $R_{o1} = -0.1$ and $R_{o2} = -10$.



(a) $v_h^T e_a = 1.0$ $\mu_h = 0.4$ (b) $v_h^T e_b = 0.8$; $\mu_h = 0.4$

Fig. 1: Human ellipse model depending on his/her velocity v_h.

IV. Socially Compliant Robot Navigation

In the Social APF-RL method, four major changes are introduced to APF-RL. First, the obstacle representations of humans are modified as to encode movement direction. Recall that each obstacle is represented by an ellipse. However, humans might be uncomfortable if a robot gets very close to them - differing from non-living obstacles. We propose an approach in which the ellipse representations of the humans are expanded depending on their motion. Each ellipse is defined by its major e_a and minor axes e_b with corresponding width a and height b as seen in Fig. 1. Then, the amount of enlargement is defined by the expansion of the ellipse along the major and minor axis directions.

$$a = a_o + \delta a \qquad b = b_o + \delta b$$

where a_o and b_o refer to the width and length along the respective axes e_a and e_b corresponding to the initial ellipse representation of the human. Suppose the human is detected at position h and is moving with planar velocity v_h - as obtained from the Kalman filter estimation of human movement. The expansion is done as to minimize the possibility of the robot intercepting with the pedestrian's path. Hence, rather than a symmetric increase in size, corresponding ellipse representations are enlarged in the direction of their movements. To make enlargement possible without increasing the size in the back side, the center is moved in the enlargement direction by half of the enlargement. The amount of expansion is defined by the product of human ellipse expansion parameter μ_h and the speed of the human along this direction:

$$\delta a = \xi_a \mu_h v_h^T e_a \qquad \delta b = \xi_b \mu_h v_h^T e_b$$

where

$$\xi_a = \begin{cases} 1 & \text{if } v_h^T e_a > v_h^T e_b \\ 0.5 & \text{otherwise} \end{cases} \quad \xi_b = \begin{cases} 1 & \text{if } v_h^T e_a < v_h^T e_b \\ 0.5 & \text{otherwise} \end{cases}$$

Two different cases are shown in Fig. 1. In each figure, the arrows represent the velocity vector.

Secondly, the input to the DRL network is changed as to incorporate this information. In addition to the inputs including range data $q(t)$, $q(t-\delta t)$ and the respective goal positions, the binary human labeling of the data $q_h(t)$ is newly input. Thus, the observation is as follows $\omega(t) = (q(t), q(t-\delta t), g(t), g(t-\delta t), q_h(t))$ where δt is the time-step of processing. The motivation for using human labels is to make agent take extra precautions for humans.

Thirdly, learning (k-parameter function and the waypoint function g) is expanded to include the μ_h parameter function. Each action $a \in \mathcal{A} \subset \boldsymbol{R}^4$ is now defined by the 4-tuple $a = \begin{bmatrix} k & g^T & \mu_h \end{bmatrix}$. Here $\mu_h \in \mathcal{H}$ corresponds human ellipse enlargement parameter where $\mathcal{H} \subset \boldsymbol{R}$ defines its range. SAC is again used as the deep learning method. In this case, networks have three hidden layers and each of them has 512 neurons. Hidden layer weights of the learned APF-RL model are used as initial values of this new model to jump-start the training. Actor learning is achieved via maximizing expected reward while also maximizing entropy. The selection of reward function has significant effect on the performance of the agent since the reward is the only feedback given to it. Let $r' : \mathcal{A} \times \Omega \rightarrow [R_{min}, R_{max}]$ denote the reward function with the parameters R_{min}, R_{max} corresponding to the minimum and maximum reward respectively. It is defined based on the proximity of the robot to the goal location as well as the obstacles sensed based on the observation $\omega(t)$. For each detected human position $h_i \in \boldsymbol{R}^2$, a penalty $r_{hi}\left(a(t), \omega(t)\right)$ for intruding the personal zone of the humans is added in order to obtain a socially compliant navigation:

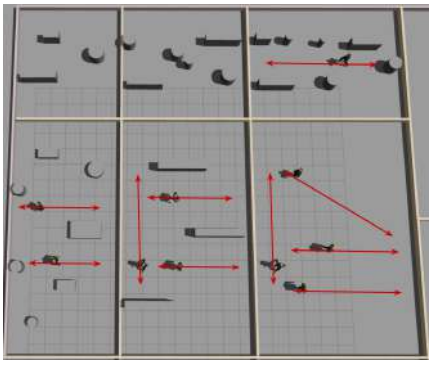
$$r'\left(a(t); \omega(t)\right) = r\left(a(t), \omega(t)\right) + r_h\left(a(t), \omega(t)\right)$$

with $r\left(a(t), \omega(t)\right)$ defined as in Section III. The definition of r_h is as follows:

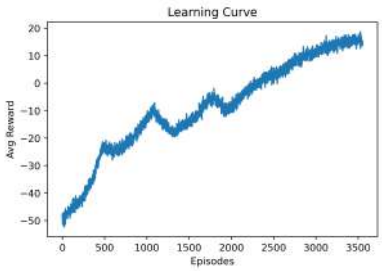
$$r_h\left(a(t), \omega(t)\right) = \sum_{i=1}^{N_h} r_{hi}\left(a(t), \omega(t)\right) \quad where$$

$$r_{hi}\left(a(t), \omega(t)\right) = \begin{cases} R_h(D_h - |c(t) - h_i(t)|) & \text{if } |c(t) - h_i(t)| < D_h \\ 0 & \text{else} \end{cases}$$

Here, D_h denotes human distance threshold for social penalty and the penalty increases proportionally as the robot gets closer to the pedestrian. R_h is the multiplier of the social penalty and it determines its importance against other rewards. In practice, $R_h = 0.6$ and $D_h = 1.4$ (sum of personal zone distance and robot radius). Training environment for the social training is shown in Fig. 2a. First two rooms have only static obstacles because the agent needs to learn to reach the target and obstacle avoidance first. Then, the pedestrians are added with increasing numbers to each room. The training starts at the first room and continues with the next one when it is learned. When all rooms are learned, a new room for training is selected at every 10 episodes and selection is done inversely proportional the respective success rate of the rooms. Training is continued until convergence of the average reward $\bar{r}'(t)$ is obtained. It is computed using last $N_T = 300$ episodes. The evolution of $\bar{r}'(t)$ is shown in Fig. 2b. Its convergence occurs around 3500 episodes.



(a) Training scenarios. Increasing static complexity (top to bottom) and increasing dynamic complexity (left to right)



(b) Learning curve.

Fig. 2: Social APF-RL training

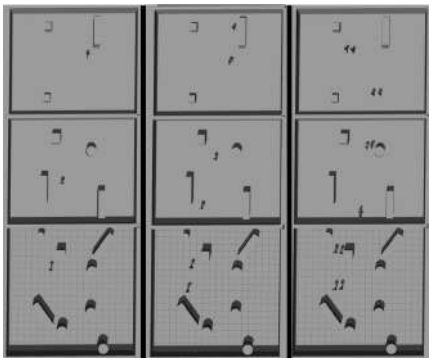


Fig. 3: Simulation test scenarios. Increasing static complexity (top to bottom) and increasing dynamic complexity (left to right)

V. Experimental Results

Social APF-RL is tested both in simulation and with a physical robot endowed with a RGB-D camera. Human detection is done using YOLO [26]. In order to alleviate the effect of noisy depth measurements, Kalman filtering is used in 3D position estimation. This also enables the estimation of human's velocity. Performance metrics are: success rate (safe arrival at the goal), travel distance to the goal and mean distance to humans. Path lengths are computed considering only successful runs and are meaningful only with high success rates. A comparative study is also conducted. The classical social navigation method social force model [7] is selected as a baseline. Also, classical APF [20] and APF-RL [4] are used as the other baselines. Note that the SFM parameters regarding relative weights of obstacles and human avoidance need to be initially manually tuned.

A. Simulation Results

Nine test scenarios are designed as shown in Fig. 3. In each row, the static complexity of the room increases as the number of obstacles increases and the arrangement

TABLE I: Comparative simulation results

Methods	Success Rate (%)	Travel Dist (m)	Human Dist (m)	Success Rate (%)	Travel Dist (m)	Human Dist (m)	Success Rate (%)	Travel Dist (m)	Human Dist (m)
	S1D1			**S2D1**			**S3D1**		
APF	90	11.3	4.7	75	13.8	4.5	65	14.9	4.1
APF-RL	100	10.1	4.5	100	12.2	4.3	95	13.5	3.8
SFM	90	11.8	5.3	70	13.1	5.0	60	14.0	4.7
Social APF-RL	100	11.1	5.2	100	12.9	4.9	95	13.9	4.5
	S1D2			**S2D2**			**S3D2**		
APF	85	12.9	4.2	70	15.2	3.9	60	15.3	3.7
APF-RL	90	10.6	4.0	85	12.4	3.6	80	13.9	3.3
SFM	85	12.3	4.7	65	13.9	4.5	50	14.5	4.1
Social APF-RL	100	11.9	4.7	90	13.4	4.3	85	14.9	3.9
	S1D3			**S2D3**			**S3D3**		
APF	75	13.7	3.1	60	13.7	3.1	50	15.7	3.0
APF-RL	80	11.4	2.9	75	11.4	2.9	60	14.5	2.8
SFM	75	13.0	3.9	60	13.0	3.9	40	14.9	3.7
Social APF-RL	90	12.5	3.6	80	12.5	3.6	70	15.6	3.2

of obstacles is more complex. In each column, dynamic complexity which is represented by the number of moving people increases. In the third row, humans walk in groups so it is additionally harder to avoid them. These are completely new in comparison to those of learning and totally unseen by the agent. All methods are tested in these rooms with the same initial and target locations. The evaluation is done based on these 20 location pairs. The rooms are categorized based on the static ('S') complexity levels and dynamic ('D') complexity levels. As the level increases, so does the complexity. Average performance results are given in Table I. It can be seen that the proposed method has the best success rate in all scenarios. Its travel distance is usually higher than APF-RL which can be related to moving away from people around. The mean human distance of Social APF-RL is higher than both APF and APF-RL, and it is comparable to SFM. However, SFM has a much lower success rate, especially in complex environments. It is observed that SFM struggles at avoiding humans and obstacles at the same time.

B. Real Robot Results

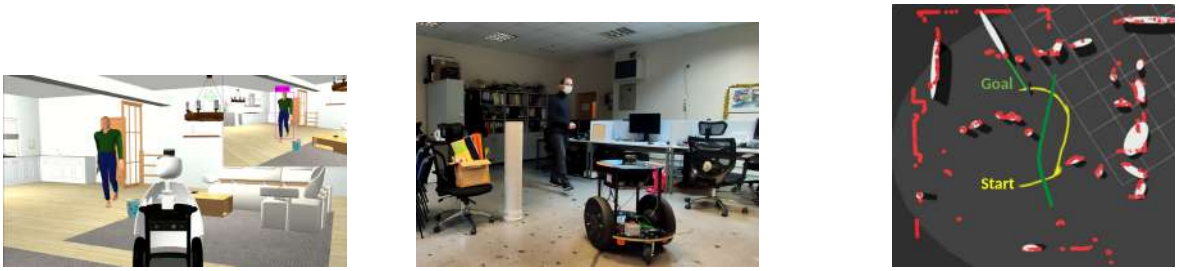
Real robot experiments are conducted with a mobile robot SempRob that has been developed within our lab. Human detection and tracking are done based on the visual data obtained from a ZED2 camera. First, tests are done in Gazebo in the small house world [27] with non-convex obstacles present. A human moves along a pre-defined path with 0.7m/s linear speed as seen in Fig. 4a. SempRob is made to navigate between random initial and goal positions with maximum 0.8 m/sec linear speed and $\frac{\pi}{4}$ rad/s angular speed. A run is successful iff the robot is able to reach $\tau_p = 0.5$ m of the goal. In the comparative statistical results from 20 simulations as given in Table II, all methods except SFM are observed to have similar success rates. However, Social APF-RL has the best performance with the lowest path length while maintaining the largest distance to human. Following, the proposed algorithm is tested on SempRob in a real setting. The humans follow different paths like crossing the robot's path and approaching from the opposite direction as seen in Fig. 4b. A sample path resulting from social APF-RL is shown in Fig. 4c. Here, the green curve represents the human's path while the yellow one represents that of robot. The robot successfully avoids obstacles and the human and reaches the goal location. Statistical comparative results are given in Table III. Four different scenarios varying in the number of obstacles, their relative positioning and number of humans (one or two) are considered and each scenario is repeated 15 times. Here, the ratio of travel distance wrt Euclidean distance btw the initial and goal positions is computed. Here, SFM again exhibits the worst performance. It fails to avoid obstacles when there is a human nearby and the robot either gets stuck or collides with the obstacles. Classical APF is observed to have lower success rate and human distance. This is attributed to head-to-head navigation trajectories in which the robot tends to falter. It is observed that Social APF-RL has the best performance in terms of success rate while also maintaining social distance to humans. Its path length performance is also on par with the best result by the APF-RL method.

TABLE II: Small House Simulation Results

Method	Success Rate (%)	Travel Dist. (m)	Human Dist. (m)
APF	85	8.2166	5.5
APF-RL	85	9.1828	5.3
SFM	50	8.0643	5.1
Social APF-RL	80	8.0551	5.7

VI. Conclusion

This paper presents Social APF-RL for robot navigation in human-populated environments. Our method extends APF-RL that combines artificial potential functions and deep reinforcement learning in order to obtain safe and efficient navigation. However, differing from it, Social APF-RL aims



(a) Small house world test scene. (b) Real robot experiment scene. (c) Human path (green) vs robot path (yellow).

Fig. 4: Real robot experiment: The robot successfully reaches its goal while also avoiding obstacles and respecting the comfort zone of the pedestrian.

TABLE III: Comparative real robot results

Method	Success Rate (%)	Travel Dist. (%)	Human Dist. (m)
APF	50	149.91	2.3491
APF-RL	93	145.03	2.5614
SFM	77	133.52	2.6080
Social APF-RL	97	125.59	2.7714

to stay away from the personal zones of humans around as much as possible. This is achieved through introducing four major changes - including obstacle representation of humans while taking their velocities into account, associating depth input to DRL network with 'human'-'no human' labels, learned parameters and reward function. As static or dynamic complexity of robot's current place increases, Social APF-RL performs considerably better than all previous methods with regard to reliable navigation to the goal while maintaining distance to encountered humans along the way. Future work will focus on social navigation in large-scale environments.

ACKNOWLEDGEMENTS

This work has been supported in part by TUBITAK grant EEEAG-118E857 and ROYAL CB SBB 2019K12-149250. We kindly acknowledge Serhat İşcan for his contributions in the experimental evaluation.

REFERENCES

[1] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics Auton. Syst.*, vol. 61, pp. 1726–1743, 2013.

[2] K. Dautenhahn, "Socially intelligent robots: dimensions of human-robot interaction." *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, vol. 362 1480, pp. 679–704, 2007.

[3] A. K. Pandey and R. Alami, "A framework towards a socially aware mobile robot motion in human-centered dynamic environment," *IEEE/RSJ IROS*, pp. 5855–5860, 2010.

[4] K. Bektaş and H. I. Bozma, "APF-RL: Safe mapless navigation in unknown environments," in *ICRA*, 2022, pp. 7299–7305.

[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *ArXiv*, vol. abs/1801.01290, 2018.

[6] E. T. Hall, "The hidden dimension," 1966.

[7] Helbing and Molnar, "Social force model for pedestrian dynamics." *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, vol. 51 5, pp. 4282–4286, 1995.

[8] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," *IEEE/RSJ IROS*, pp. 1688–1694, 2013.

[9] ——, "Social-aware robot navigation in urban environments," *2013 European Conference on Mobile Robots*, pp. 331–336, 2013.

[10] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," *IEEE/RSJ IROS*, pp. 3052–3059, 2018.

[11] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," *ICRA*, pp. 6015–6022, 2018.

[12] J. Jin, N. M. Nguyen, N. Sakib, D. E. Graves, H. Yao, and M. Jägersand, "Mapless navigation among dynamics with social-safety-awareness: a reinforcement learning approach from 2d laser scans," *ArXiv*, vol. abs/1911.03074, 2019.

[13] A. J. Sathyamoorthy, U. Patel, T. Guan, and D. Manocha, "Frozone: Freezing-free, pedestrian-friendly navigation in human crowds," *IEEE Robotics and Automation Letters*, vol. 5, pp. 4352–4359, 2020.

[14] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: An experimental comparison," *IEEE/RSJ IROS*, pp. 1341–1346, 2014.

[15] B. Okal and K. O. Arras, "Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning," *ICRA*, pp. 2889–2895, 2016.

[16] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," *IEEE/RSJ IROS*, pp. 1343–1350, 2017.

[17] M. Hamandi, M. D'Arcy, and P. Fazli, "Deepmotion: Learning to navigate like humans," *IEEE Int'l Conf. on Robot and Human Interactive Communication*, pp. 1–7, 2019.

[18] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *IEEE Conf. on CVPR*, pp. 779–788, 2016.

[19] L. Bertoni, S. Kreiss, and A. Alahi, "Monoloco: Monocular 3d pedestrian localization and uncertainty estimation," *Int'l Conf. on Computer Vision*, pp. 6860–6870, 2019.

[20] E. D. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robotics and Automation*, vol. 8, pp. 501–518, 1992.

[21] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, 2015.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016.

[24] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *ArXiv*, vol. abs/1812.05905, 2018.

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.

[26] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[27] A. Robotics, "Small house world," https://github.com/aws-robotics/aws-robomaker-small-house-world, 2019.

A new flex-sensor-based umbilical-length management system for underwater robots

Ornella Tortorici, Cédric Anthierens and Vincent Hugel

***Abstract*— This work focuses on the automatic control of the length of a tether that links an underwater vehicle to the surface, with the objective to prevent the tether from becoming taut or getting entangled due to too much length being deployed. The solution proposed here consists of equipping the tether with a balanced buoy-ballast system that gives the cable a V-shape in the vicinity of the vehicle. This system offers a passive compliance by smoothing the movements of the tether and damping external disturbances. The tether length is adjusted by an active feeder on the surface, whose control relies on the reading of a flex sensor embedded in the V-shape portion of the cable. The experiments conducted on a real ROV in a pool allowed validating this mechatronic compliant-actuated system, which can adapt to the movements of the underwater vehicle while it executes longitudinal and curved trajectories.**

I. Introduction

Underwater exploration is a promising and sensitive field which takes advantages of the manoeuvrability and reliability of remotely operated vehicles (ROV) [1], [2]. Missions like hulls or pontoons inspection for maintenance tasks do typically require such devices. Those systems are linked to a control station by a tether that can transmit data and supply power if required [3], [4]. However, this link may apply undesired forces on the ROV [5], [6] that imply a limitation of the ROV mobility, an increase of its power consumption and disturbances on its trajectory [7]–[10]. All these constraints are even more important for the small and less powerful ROV which are widely used in shallow waters. Furthermore, a passive slack tether increases the risk of entanglement, drag on the seabed thus early wear [11]–[13]. In order to take advantage of the cables linked to underwater robots, a variable cable length is required.

One of the main challenges in underwater robotics is to provide more autonomy to the robots, whereas the cable length is mostly managed manually. There exist three main solutions in the literature to manage tethers: tether customization/instrumentation, use of a surface winch, or use of an underwater tether management system (TMS) whose function can be carried out by a second robot.

Tethers are often customized by buoys and ballasts to change their buoyancy, shape or behaviour [6], [14], [15]. Those systems are passive and their positive impact on the cable management is limited if they are not associated with an active control. Less commonly, cables are instrumented with external or internal sensors to measure their behaviour and shape. The measurement is done either on several specific nodes along the cable by inertial or tension sensors [6], [16], or continuously all along the cable through embedded fiber optic solutions [17]–[19]. The first solution generates irregular shape, whereas the second one can be very expensive. Surface winches are used to deliver /retrieve cable, and their control is often manual or simply based on cable tension [20], [21]. They are commonly placed on the surface vessel, but they can also be embedded on the ROV itself [14]. Tether management systems (TMS) are widely used for deep water systems [1], [22]–[24]. They behave as an intermediate system between the surface and the ROV that manages the portion of the cable connected to the ROV. A second ROV can also play the role of a TMS [2]. However, this solution adds a potential risk of collision between the robots.

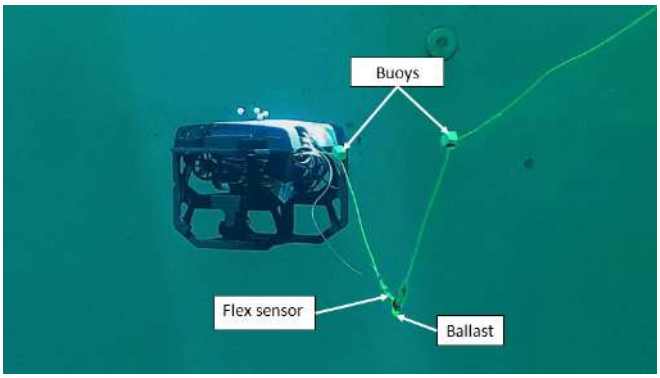


Fig. 1. Buoy-ballast compliant system mounted on neutrally buoyant cable.

This paper presents the design of an automatic cable management system to limit the undesired effects of the tether on the navigation of the ROV. The contributions of this work include the design of a mechatronic compliant-actuated system for a tether that is linked to an underwater robot, the associated length control management to maintain a semi-stretched shape of the tether, and experimental validations with the whole system connected to a compact underwater vehicle.

The paper is organized as follows. Section II details the proposed solution, including the mechatronic adaptation brought to the tether and the control scheme. Section III presents the experimental setup used for evaluating the cable management system. Section IV reports and discusses the experimental results. Finally, Section V draws the conclusions of the work.

Ornella Tortorici,Cédric Anthierens and Vincent Hugel are with COSMER laboratory EA7398, University of Toulon, avenue de l'Universite, CS60584, 83041 Toulon Cedex 9, France. Corresponding author: Cédric Anthierens cedric.anthierens@univ-tln.fr
979-8-3503-0704-7/23/$ 31.00

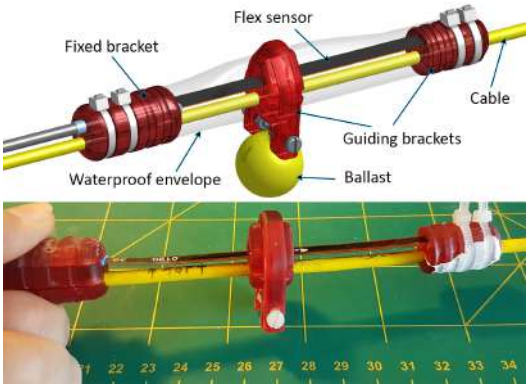


Fig. 2. Integration of the flex sensor on the tip of the V-shape buoy-ballast compliant system

II. Method

A. Mechatronics

The mechatronics of the system includes customization of the tether near the ROV vehicle to achieve passive compliance, and the design of an active feeder on the surface to automatically control the tether length.

The passive compliance is based on the local deformation of the cable created by two buoys and a ballast fixed on the cable, that give it a V-shape as shown in Fig. 1. This V-shape portion of the cable is symmetrically designed with the ballast in the middle of the two buoys to have a neutrally buoyant system. A flex sensor is mounted on the cable at the ballast place using a fixed bracket and guides along the cable (Fig.2). The sensor has a negligible bending stiffness. It is isolated from the water by a thin plastic envelope. The buoy-ballast system is placed in proximity to the ROV. The heavier the ballast, the stiffer the system, but the higher the drag force. The reactivity also depends on the stiffness. The deformed part of the cable must keep a V shape and not a droplet shape, so that the flex sensor provides a monotonous response with the deformation.

The buoy-ballast system is designed to smooth the movements of the tether and to damp external disturbances. Table I summarizes the specifications and the characteristics of the system that have been determined by simulation for a Fathom Slim tether from Blue Robotics that is neutrally buoyant in freshwater and has a low stiffness. Figure 3 shows the data acquisition chain of the flex sensor.

buoyancy	neutral
desired compliance	1 m
distance between buoys	min: 20 cm, max.: 1.2 m
flex sensor model	FS-L-0095-103-ST (Spectra Symbol)
flex sensor size	11 cm long, 0.5 mm thick
buoy size	2.9 x 2.9 x 4.7 cm^3
buoy foam density	288 kg/m^3
ballast mass	76 g

TABLE I

Specifications and characteristics of the buoy-ballast V shape system adapted for the Fathom Slim tether.

The specifications for the tether feeder are the following,

- it must be able to pay out cable at the same speed as the ROV movements.
- in case the feeder system becomes inactive, it should not stop the movements of the ROV
- if the control of the ROV is lost, the feeder must be capable of trailing the ROV back to the surface vessel

Therefore, the feeder must be able to bear more than the ROV's dead weight in water, but less than the maximum ROV thrust. Here, the ROV is a BlueRov 2 from BlueRobotics that has a maximum forward speed of 1.5 m/s and a maximum forward thrust of 100 N. A strength of 11 N is necessary to drag it in water at 0.5 m/s. Furthermore, the feeder must have a smooth behaviour and be controllable at low speed. To ensure control, the length measurement must remain accurate and avoid any slippage, even with a wet tether.

The feeder structure is depicted on Fig. 4. The cable is fed in and out by transmission between two gears of the same diameter. One of the two gears is actively driven by a motor and the other one is passively driven by friction from the cable. This assembly is composed of machined parts, rapid prototyping parts (ABS) and off-the-shelf components. An incremental encoder on the passive gear measures the length of unwound tether. This part comprises a flange, a toothed gear and an encoder shaft guided in rotation by a bearing box. The toothed gear guarantees a good grip on the cable without the need to tighten it too much between the two gears. The drive gear is actuated by a DC motor with an integrated gearbox.

B. Control scheme

Modeling of the flex sensor. Since the resistor of the sensor is directly linked to its bending, a 3rd order polynomial fit between the output voltage and the distance between buoys has been drawn and identified (Fig. 5). A cubic regression was chosen to take into account a possible inflection point. The error between the experimental points and the fit is also plotted. The average error is 3 cm, and the maximum error is 7 cm. The experimental carried out to determine the fit showed a very fast response (1.26 s as time constant), a monotonic behavior and no significant hysteresis or phase shift.

The estimation of the distance by the flex sensor model is sent by the ROV to the feeder on the surface, which is actuated to keep an average distance between buoys. Because of the drag force, the gap between the buoys increases with the ROV speed. So the normal distance between buoys is set relative to the actual ROV speed. During the ROV motion, the feeder speed is controlled to regulate the desired average gap between both buoys.

Figure 6 describes the control block diagram of the feeder. The control of the feeder consists of a length control loop (proportional controller) that encloses the speed control loop (proportional-integral controller with anti-windup). The cable length is computed in the speed control loop so that it is accessible even when the length control is disabled. The

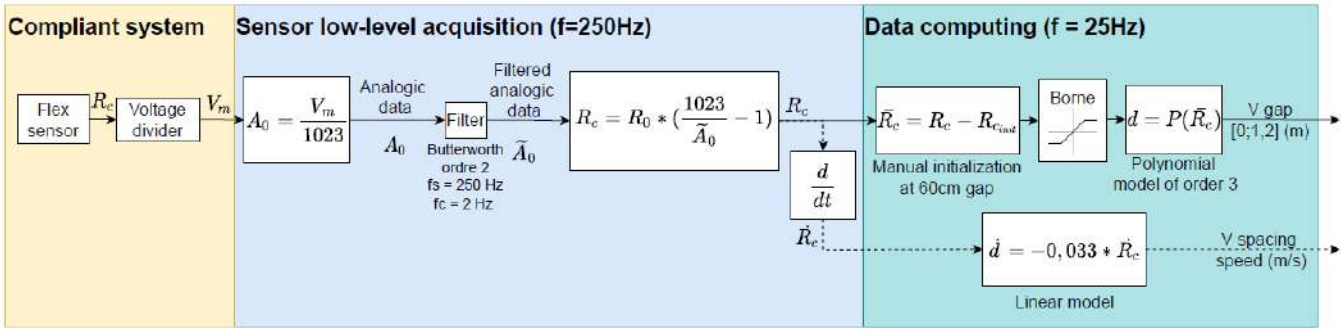


Fig. 3. Block diagram of the flex sensor data acquisition

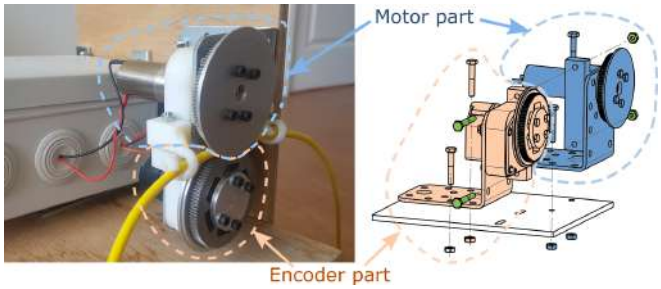


Fig. 4. Overview of the tether feeder, composed of a driving part and an encoder part.

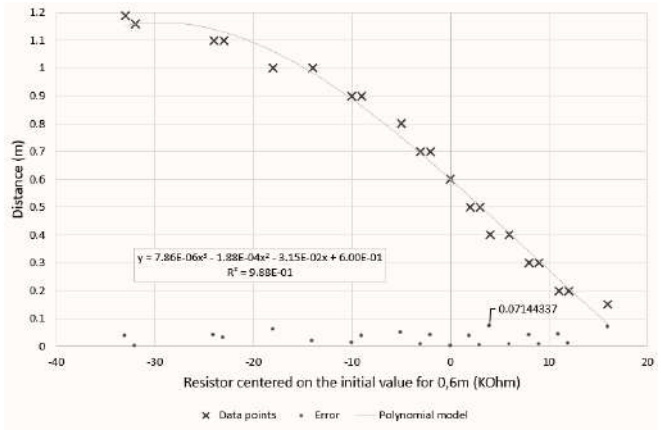


Fig. 5. Model of the distance between the buoys as a function of the centered resistance of the flex sensor with experimental data points.

desired speed, whether it comes directly as a controller input or from the length control loop, is first bounded to avoid overshooting the motor limits. It outputs a raw PWM signal, which is then bounded and smoothed by a low pass filter before it is transmitted to the motor control board.

The feeder can be operated in two ways to release the cable on request. Either it works sequentially, i.e. the length controller releases 2 m of cable when the V-system is getting taut (2 m of cable is reeled back after 10 s period when the V-system is loosed), or it works continuously and so the speed control loop releases the cable when the V-system lengthens, or reels it in when the gap between the buoys is less than the average.

III. Experiments

For the experiments, the feeder is fixed on the edge of an experimental water tank (16 x 8 *m* pool with a maximum depth of 5 *m*) and is connected to the ROV through the 25 m long umbilical equipped with the V-shape buoy-ballast system. Figure 7 represents the global implementation scheme of the system.

The behaviour of the system is tracked by an underwater motion-tracking system, namely *Qualisys*, in addition to internal sensors. These data are used in post-processing to obtain the configuration of the cable, as well as the position and orientation of the ROV in the global frame.

In order to compare the behaviour of the system in different modes of the cable, two trajectories of the ROV were defined:

- a linear trajectory where the ROV goes forwards and then backwards
- a curvilinear trajectory.

They illustrate two configurations where the cable control should play an important role in relation to a high risk of cable snagging or entanglement. All these trajectories are associated with a 1.5 *m* depth control of the ROV. The ROV is controlled in open-loop to track these trajectories by setting the thrust level for each degree of freedom. Therefore, the observed trajectories are expected to be different depending on the mode of the cable, namely taut, slack or controlled. The controlled mode of the cable is tuned to keep an average distance of 0.65 *m* between the buoys with a tolerance of ± 0.05 *m*.

IV. Results and Discussion

a) Linear trajectory: The cable control for this trajectory is illustrated on Fig. 8 for the three modes. The feeder appears to be quite reactive and smooth to wind/unwind the cable depending on the ROV motion and the distance between both buoys.

Figure 9 depicts the paths of the ROV projected onto the horizontal plane measured by the *Qualisys* system for the three cable modes. If there were no external disturbances at all, the path would be rectilinear. The ROV deflects slightly to the left when the cable is slack. This deviation is slightly larger in control mode and is observed both during its forward and backward motion. The deviation

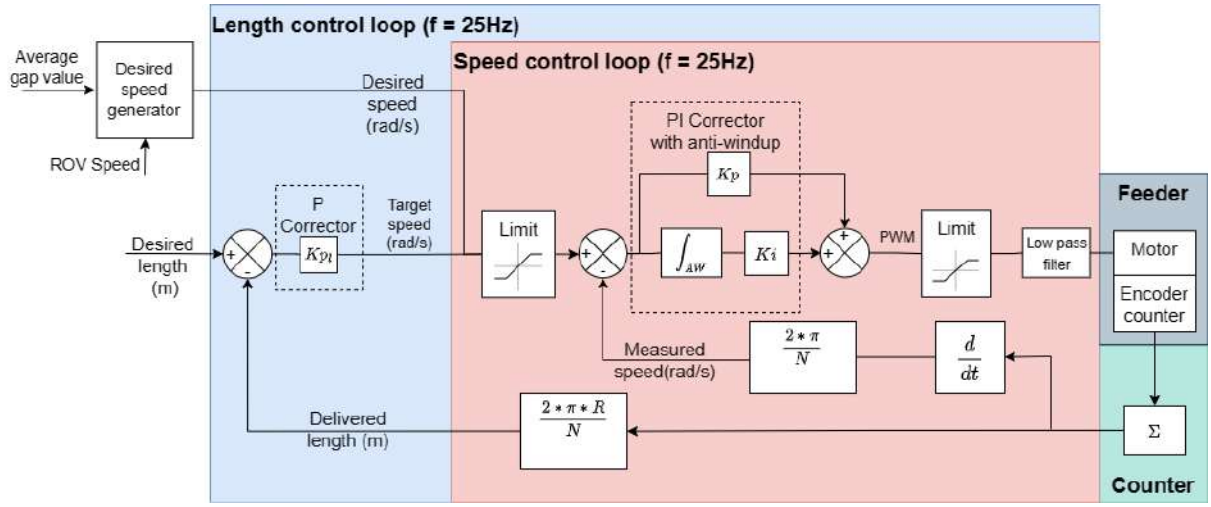


Fig. 6. Control block diagram of the feeder. The input is either a cable length value or a speed value.

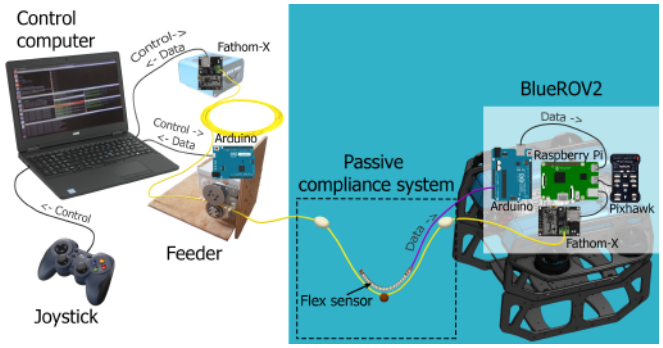


Fig. 7. Global implementation diagram of the system

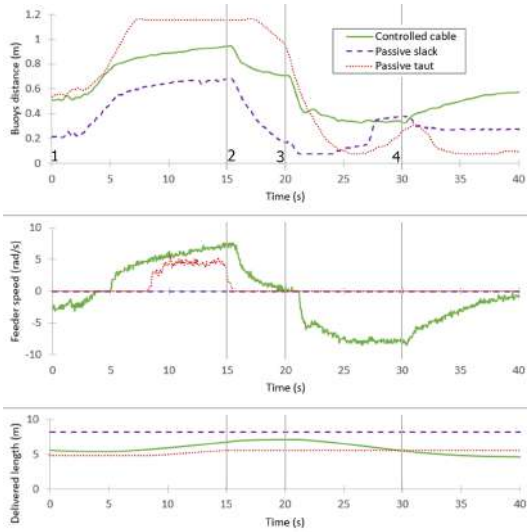


Fig. 8. Distance between buoys (flex sensor), feeder speed and unwound length of cable in control mode, passive slack mode and passive taut mode for forward-backward trajectory. (https://youtu.be/owekUkN_UtM for control mode, https://youtu.be/1RTT23-USDY for passive taut mode, https://youtu.be/FG5iyNfjzck for passive slack mode.)

is significantly greater in passive taut mode, approaching 90deg. Furthermore, the distance covered by the ROV is significantly shorter with the taut cable.

Figure 10 shows the evolution of the depth control of the ROV for the three modes. Only the control mode efficiently helps to regulate the ROV depth. The vertical thrusters work significantly more when the cable is not controlled than when it is controlled.

b) Curvilinear trajectory: To achieve a slalom shape, a forward thrust level of 25% is sent for 15*s* (between points 1 and 2), then a yaw command of 11.25% is sent in addition with a forward thrust level of 40% for 6.5*s* to turn to the right (between points 2 and 3), followed by a left turn with the same levels for 6*s* (between points 3 and 4), and finally a forward thrust of 25% for 4*s* (between points 4 and 5).

The compliant system keeps its V-shape and a reasonable distance between the buoys even when the ROV turns (Fig. 11). Figure 12 presents the behavior of the compliant system in the three modes. The feeder is also reactive and smooth to reel back the cable and manage the desired gap between both buoys.

For a flawless system without any disturbance, the yaw angle (Fig. 13) should be constant during the ROV straight line commands (before point 2 and after point 5). It should also be linear during rotating commands (between points 2 and 3, then 3 and 4). There is a small deviation of about 20° to the right during the first straight line (2.6 *m*) command of the ROV for the passive slack cable. This deviation is oriented to the left and its absolute value is doubled with the controlled cable and doubled again with the passive taut cable. In fact, the cable is fixed on the left side of the back of the ROV, which induces a slight deviation to the left for the controlled cable both in straight line and during a right rotation. This deviation is much larger for the passive taut cable. The left rotation of the ROV appears to be less affected in the control mode and in the passive taut mode.

The impact of these deviations is observed on Fig. 14, which presents the ROV paths, projected onto the horizontal plane, for the three modes. The distances traveled are quite

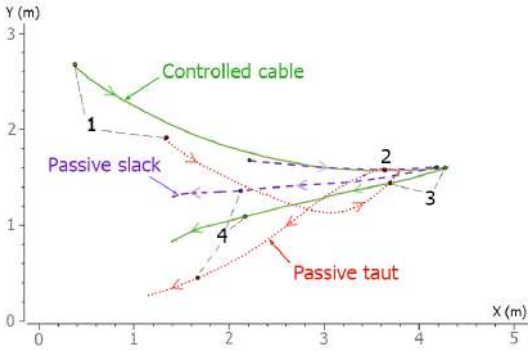


Fig. 9. Actual forward-backward trajectories of the ROV projected in the horizontal plane with cable control and with slack or taut passive cable. These paths are superimposed on point 2 for easier comparison.

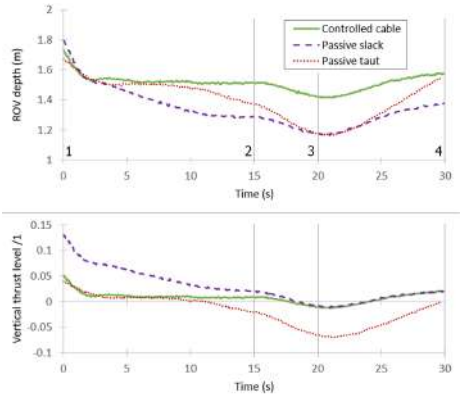


Fig. 10. ROV depth and vertical depth-control thrust in the three modes.

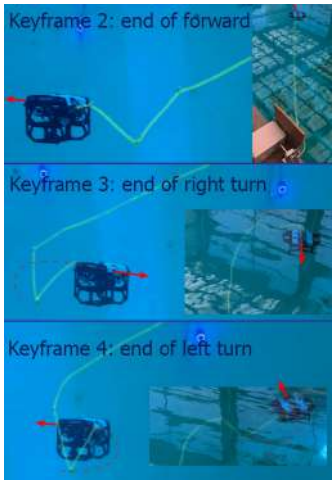


Fig. 11. Views of the overall system at different points along the curvilinear trajectory when the cable is controlled. The red arrows indicate the direction of motion of the ROV. (Video at https://youtu.be/fy-JTc8PvIY)

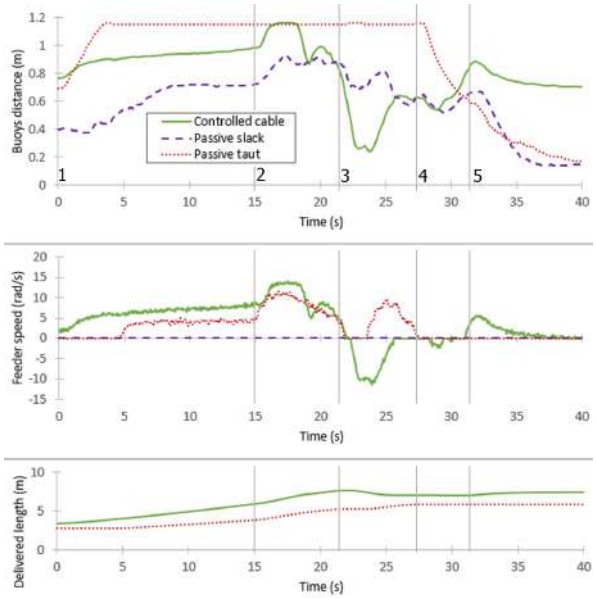


Fig. 12. Distance between buoys (flex sensor), feeder speed and unwound length of cable in control mode, passive slack mode and passive taut mode for the curvilinear trajectory. (Videos at https://youtu.be/LR4BKefRSnM for passive slack mode and at https://youtu.be/74p5Bzee9kY for passive taut mode.)

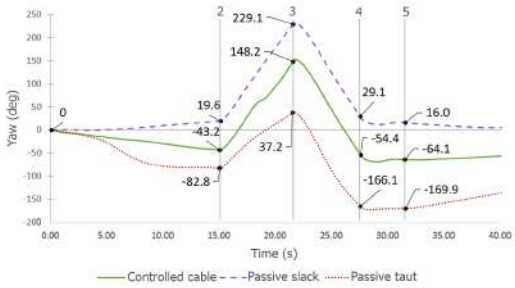


Fig. 13. Comparison of the yaw angle of the ROV (measured with its embedded compass) along the curvilinear trajectory when the cable is controlled or not. The angle was initialized to 0^o at the beginning of the trajectories to facilitate their comparison. An increase of the angle represents a rotation to the right of the ROV.

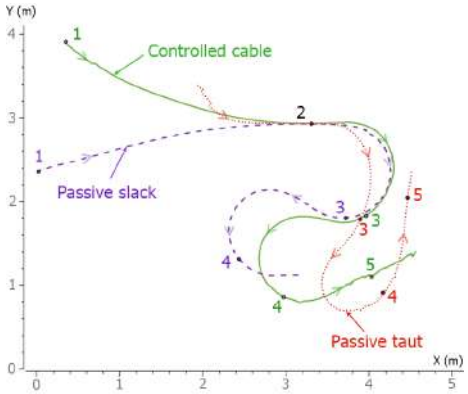


Fig. 14. Comparison of the actual curvilinear path of the ROV projected onto the horizontal plane, measured by the *Qualisys* system, with or without cable control. These paths are superimposed on point 2 for easier comparison.

close for the controlled cable and the passive slack cable. The turns are slightly different between these two modes. The path of the ROV with the passive taut cable is totally distorted. The first rotation to the right (between points 2 and 3) is very confined and the rotation to the left (between points 3 and 4) is quite irregular (much wider curvature in the middle than at the beginning and the end).

The experiments also showed that the control of the cable with passive compliance is effective in keeping the cable in a semi-stretched configuration, the delivered length being properly managed, and preventing the creation of cable loops and reducing the risks of snagging and tangles. The control mode generates a slight tension in the cable, which is transmitted to the ROV and results in a minor deviation in the trajectories of the system. This deviation could be avoided by fixing the cable closer to the center of gravity of the ROV. In addition, the passive compliance system appears to improve the stability of the depth control of the ROV.

V. Conclusion

This paper describes a mechatronic solution to automatically and actively manage the cable length of a ROV. The cable is equipped with a balanced buoy-ballast system, which creates a V-shape in the cable near the ROV and provides a passive compliance to it. The buoy-ballast system has been made for a specific tether, namely the Fathom Slim from BlueRobotics, but the design methodology can be used to equip other types of cables with different physical properties. The experiments show that the feeder is responsive enough with respect to the command speeds of the ROV, with no error or drift observed on the controlled cable length. Even wet, the cable does not slip through the feeder. Longitudinal forward-backward and curvilinear trajectories have been tested to validate the capability of the entire system to keep the cable in a semi-stretched configuration.

Future developments will focus on the implementation of the cable feeder on a surface vehicle (USV) and the monitoring of the semi-stretched configuration to use the cable as a means of proprioception for the estimation of the relative position between the vehicles. The measurement of the consummed current will help to estimate the cable strain on the USV side. Synchronized navigation strategies between the ROV and the USV are also under prospect to have an optimized displacement of the vehicles in terms of energy consumption and seabed coverage in shallow waters.

Acknowledgements

We gratefully thank the Subsea Tech company for their technical support and CIRS Girona (Spain) for providing us with their water tank to run our experiments. This work was supported by the French Provence-Alpes-Cote d'Azur region.

References

[1] R. Christ and R. Wernli, *The ROV Manual A User Guide for Remotely Operated Vehicles - Second Edition*. Elsevier, 2014.

[2] O. Khatib, X. Yeh, G. Brantner, B. Soe, B. Kim, S. Ganguly, H. Stuart, S. Wang, M. Cutkosky, A. Edsinger, P. Mullins, M. Barham, C. Voolstra, K. Salama, M. L'Hour, and V. Creuze, "Ocean one: A robotic avatar for oceanic discovery," *IEEE Robotics & Automation Magazine*, vol. 23, no. 4, pp. 20–29, 2016.

[3] R. Capocci, G. Dooly, E. Omerdić, J. Coleman, T. Newe, and D. Toal, "Inspection-Class Remotely Operated Vehicles—A Review," *Journal of Marine Science and Engineering*, vol. 5, no. 1, p. 13, 2017.

[4] S. A. Ajwad and J. Iqbal, "Recent Advances and applications of tethered robotic systems," in *Science International*, vol. 26, 2014.

[5] O. Tortorici, C. Anthierens, V. Hugel, and H. Barthelemy, "Towards active self-management of umbilical linking ROV and USV for safer submarine missions," *IFAC-PapersOnLine*, vol. 52, no. 21, pp. 265–270, 2019.

[6] T. W. McLain and S. M. Rock, "Experimental Measurement of ROV Tether Tension," in *Proceedings of ROV '92*. San Diego, California: IEEE, 1992, p. 6.

[7] L. Bevilacqua, W. Kleczka, and E. Kreuzer, "On the Mathematical Modeling of ROV'S," *IFAC Proceedings Volumes*, vol. 24, no. 9, pp. 51–54, 1991.

[8] S. Soylu, B. J. Buckham, and R. P. Podhorodeski, "Dynamics and control of tethered underwater-manipulator systems," in *OCEANS 2010 MTS/IEEE SEATTLE*, 2010, pp. 1–8.

[9] M.-C. Fang, C.-S. Hou, and J.-H. Luo, "On the motions of the underwater remotely operated vehicle with the umbilical cable effect," *Ocean Engineering*, vol. 34, no. 8, pp. 1275–1289, 2007.

[10] Z. Feng and R. Allen, "Evaluation of the effects of the communication cable on the dynamics of an underwater flight vehicle," *Ocean Engineering*, vol. 31, no. 8, pp. 1019–1035, 2004.

[11] A. Gay Neto and C. de Arruda Martins, "Structural stability of flexible lines in catenary configuration under torsion," *Marine Structures*, vol. 34, pp. 16–40, 2013.

[12] J. Coyne, "Analysis of the formation and elimination of loops in twisted cable," *IEEE Journal of Oceanic Engineering*, vol. 15, no. 2, pp. 72–83, 1990.

[13] G. Drumond, I. Pasqualino, B. Pinheiro, and S. Estefen, "Pipelines, risers and umbilicals failures: A literature review," *Ocean Engineering*, vol. 148, pp. 412–425, 2018.

[14] L. Brignone, E. Raugel, J. Opderbecke, V. Rigaud, R. Piasco, and S. Ragot, "First sea trials of HROV the new hybrid vehicle developed by IFREMER," in *OCEANS 2015 Genova*, 2015, pp. 1–7.

[15] C. Viel, "Self-management of the umbilical of a ROV for underwater exploration," *Ocean Engineering*, vol. 248, p. 110695, 2022.

[16] J. E. Frank, R. Geiger, D. R. Kraige, and A. Murali, "Smart tether system for underwater navigation and cable shape measurement," Patent US8 437 979B2, 2013.

[17] R. G. Duncan, M. E. Froggatt, S. T. Kreger, R. J. Seeley, D. K. Gifford, A. K. Sang, and M. S. Wolfe, "High-accuracy fiber-optic shape sensing," K. J. Peters, Ed., San Diego, California, 2007, p. 65301S.

[18] C. Xu, J. Chen, D. Yan, and J. Ji, "Review of Underwater Cable Shape Detection," *Journal of Atmospheric and Oceanic Technology*, vol. 33, no. 3, pp. 597–606, 2016.

[19] C. Xu, K. Wan, J. Chen, C. Yao, D. Yan, J. Ji, and C. Wang, "Underwater cable shape detection using ShapeTape," in *OCEANS 2016 MTS/IEEE Monterey*, 2016, pp. 1–4.

[20] A. K. Banerjee and V. N. Do, "Deployment control of a cable connecting a ship to an underwater vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 6, pp. 1327–1332, 1994.

[21] C. Zhao, P. R. Thies, and L. Johanning, "Investigating the winch performance in an ASV/ROV autonomous inspection system," *Applied Ocean Research*, vol. 115, p. 102827, 2021.

[22] E. Raugel, J. Opderbecke, M. Fabri, L. Brignone, and V. Rigaud, "Operational and scientific capabilities of Ariane, Ifremer's hybrid ROV," *OCEANS 2019 - Marseille*, 2019.

[23] H. Zhou, J. Cao, B. Yao, and L. Lian, "Hierarchical NMPC–ISMC of active heave motion compensation system for TMS–ROV recovery," *Ocean Engineering*, vol. 239, p. 109834, 2021.

[24] M. B. Lubis, M. Kimiaei, and M. Efthymiou, "Alternative configurations to optimize tension in the umbilical of a work class ROV performing ultra-deep-water operation," *Ocean Engineering*, vol. 225, p. 108786, 2021.

Multi-Formation Planning and Coordination for Object Transportation

Weijian Zhang, Charlie Street, Masoumeh Mansouri

***Abstract*— Multi-robot formations have numerous applications, such as cooperative object transportation in smart warehouses. Here, robots must deliver objects in formation while avoiding intra- and inter-formation collisions. This requires solutions to multi-robot task assignment, formation generation, rigid formation maintenance, and route planning. In this paper, we present a cooperative multi-formation object transportation system which explicitly handles inter-formation collisions. For formation generation, we propose a distributed motion planning approach which combines artificial potential field methods and leader-follower based control. For formation planning, we present a heuristic search-based algorithm which uses convex segmentation techniques, and extend the minimum snap method to synthesise smooth trajectories while maintaining the formation. We also propose a variant of the dynamic window approach to avoid collisions between formations. We demonstrate the efficacy of our approach in simulation.**

I. Introduction

A common approach for large object transportation in robotics is to use multiple robots to collaboratively push, cage, or grasp the objects [1]. This paper addresses cooperative transportation problems where multiple formations carry objects on top of them (see Fig. 1), which we refer to as *multi-formation planning and coordination (MFPC)*. Here, the formations are *rigid*, i.e. the formation remains unchanged during transportation, which is necessary to balance heavy loads. To solve MFPC, we must generate and maintain effective formations until a goal location is reached while avoiding collisions with other formations and obstacles.

In this paper, we decompose MFPC into several subproblems (see Fig. 2). For formation generation, we assign robots to formations using the Hungarian algorithm [2]. We then use an artificial potential field (APF) method for navigation towards each robot's formation location. APF methods are a common distributed approach for formation control [3], but can become trapped in local minima, which prevents robots from reaching their target positions. To address this, we combine a wall-following strategy with consensus-based APF for robust formation generation.

After formation generation, each formation should move toward its destination while maintaining form and avoiding collisions with other formations and obstacles; this is the planning component of MFPC. For this, we employ a consensus-based method [4], where robots follow a virtual leader in the formation centre, and reach a consensus on velocity to maintain form [5]. The virtual leader should maintain the formation by guaranteeing a kinematically feasible path for each follower. To achieve this, we present a heuristic-based global path planner which computes optimal plans under the formation and kinematic constraints. We then synthesise smooth trajectories from the global path using an extended minimum snap approach [6]. To avoid inter-formation collisions, we propose a variant of the dynamic window approach (DWA) [7]; this is the coordination component of MFPC. Finally, we combine first-order consensus control [8] and pure pursuit control [9] to track robot trajectories and complete object transportation.

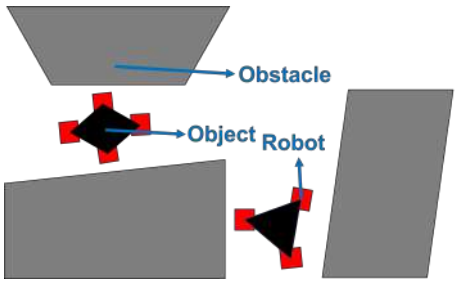


Fig. 1. Object transportation for multiple rigid formations.

The main contribution of this work is a comprehensive MFPC framework that integrates formation generation, planning, and coordination techniques. We demonstrate the performance of our MFPC system and the individual components empirically in simulation.

II. Related Work

Formation generation requires a team of robots to organise into a predefined shape [10]. Centralised approaches to formation generation can attain high performance, but scale exponentially in the number of robots, making them intractable for realistic problems [11].

In [12], an APF-based consensus control method is presented to formulate coordination and control strategies between robots without considering avoiding obstacles. An improved APF is proposed in [13] for path planning of a multi-robot formation which efficiently avoids getting trapped in local minima caused by obstacles but fails to address deadlocks among robots reaching their goal positions. An alternative decentralised approach is proposed in [14], which requires only the relative positions of robots and obstacles from each robot. This approach, however, does not guarantee

Weijian Zhang, Charlie Street and Masoumeh Mansouri are with the School of Computer Science at the University of Birmingham, wxz163@student.bham.ac.uk and {c.l.street, m.mansouri}@bham.ac.uk

Charlie Street and Masoumeh Mansouri are UK participants in Horizon Europe Project CONVINCE, and supported by UKRI grant number 10042096.

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

solution convergence. A template-based technique is used in [15] where no experiments are provided regarding the scalability issue. In our work, we ensure achieving formation convergence, a deadlock-free and scalable solution by using the Hungarian algorithm [2] for optimal task assignment and the first-order consensus method for driving robots to their goal positions [8].

Formation planning has leveraged the wealth of classical AI search methods. For instance, in [16], a relaxed A^* planner is employed to generate an optimal collision-free global path over a map where obstacles are inflated by a circle that covers the entire footprint of the formation. However, such conservative approximations can result in a loss of precision and, at times, failure to find a solution. In this paper, we define the obstacle-free space without making any approximations of the obstacles. Another example includes constrained optimisation for non-rigid formation planning [17] in a dynamic environment, an approach which cannot be applied directly to the type of rigid MFPC considered in this paper. An alternative approach is to use sampling-based methods such as rapidly-exploring random trees [18] or probabilistic roadmaps [19] that consider the geometric constraints of the formation. However, these approaches rely on sampling a large number of configurations to find a path which often suffers from abrupt variations in direction. To alleviate this problem, we use heuristic search to reduce the number of sampling configurations and abrupt changes in direction. Further, we apply a modified minimum snap method [6] to produce a feasible smooth trajectory.

For rigid object transportation, formations should be maintained during locomotion. For instance, a leader-follower strategy combined with APF is used in [20]. The leader robot determines its navigation path through APF, and the other robots in the group follow the leader to maintain the formation using distance-angle $(l-\phi)$ control. However, dynamic obstacles are not considered. In [21], the authors demonstrate a hierarchical quadratic programming approach, such that the *a priori* unknown obstacles can be detected and avoided at runtime. However, due to the lack of a global planner to generate a set of waypoints, the control may run the risk of falling into local minima. In [22], an improved A* algorithm is adopted to generate optimal global paths and a multiple sub-target APF is proposed for local path planning of the formation. In this approach, when the formation avoids dynamic obstacles, the generated trajectory toward the local subgoal may not be executable for the robot. In our work, we benefit from the smooth and executable global trajectories generated by the formation planner. Further, we extend the dynamic window approach (DWA) [7] by introducing formation prioritization to achieve collision avoidance between formations while ensuring that the robot motion constraints are not violated.

III. Preliminaries

We begin by defining our assumptions over the workspace, robots, and formations.

Workspace. Let $W \subset \mathbb{R}^2$ be a 2D workspace which contains a set of static obstacles $O \subset W$, and let $F = W \setminus O$ denote the obstacle-free workspace.

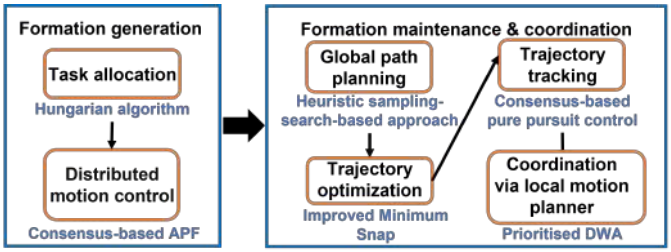


Fig. 2. The proposed problem decomposition for MFPC.

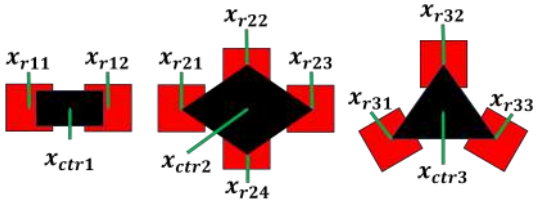


Fig. 3. Linear, rectangular, and triangular formations consisting of several robots (in red) and an object (in black). Each robot i is placed with its state $\boldsymbol{x}_{ril}$ relative to the jth formation's centroid state $\boldsymbol{x}_{ctrj}$.

Robots. Consider a team of homogeneous rectangular robots $R = \{1,...,|R|\}$, where $|R|$ is the cardinality of R (see Fig. 3). Let $\boldsymbol{x_i}(t) = \langle x_i(t), y_i(t), \theta_i(t) \rangle$ be the state of robot $i \in R$ at time t, where $x_i(t), y_i(t) \in \mathbb{R}$ denote robot i's position at time t, and $\theta_i(t) \in [-\pi, \pi)$ denotes its orientation. We assume all robots are holonomic, and use a unicycle kinematic model:

$$\dot{\boldsymbol{x_i}}(t) = \frac{d}{dt}\begin{bmatrix} x_i(t) \\ y_i(t) \\ \theta_i(t) \end{bmatrix} = \begin{bmatrix} v_i(t) \cdot cos\theta_i(t) \\ v_i(t) \cdot sin\theta_i(t) \\ \omega_i(t) \end{bmatrix} \quad i \in R, t \in [0, t_{f_i}], \tag{1}$$

where $v_i(t)$ and $\omega_i(t)$ denote the velocity and angular velocity applied at time t respectively, and t_{f_i} is the finite horizon for robot i. At each time step, we assume robots can observe the state of all other robots and obstacles within sensing range $d_0 \in \mathbb{R}_{\geq 0}$.

Formations. In this paper, we consider formations in straight lines, rectangles, and triangles (see Fig. 3). We write $z_j = < \boldsymbol{x}_{ctrj}, \boldsymbol{x}_{rj1}, ...\boldsymbol{x}_{rjk} >$ to denote the configuration of the jth formation, formed of k robots, where $\boldsymbol{x}_{ctrj}$ denotes the formation's centroid state, and $\boldsymbol{x}_{rjl}$ is the relative state of the lth robot with respect to $\boldsymbol{x}_{ctrj}$.

IV. Formation Generation

In this section, we present a distributed approach for formation generation which combines APFs with consensus-based leader-follower control and a wall-following strategy to avoid local minima and unreachable targets. The first sub-problem for formation generation is to allocate each robot i to a formation location $g_i \in G$, where g_i corresponds to a corner of a formation (see Fig. 3). For this, we use the optimal Hungarian algorithm [2] with a Euclidian distance cost function.

To drive robot i towards g_i while avoiding obstacles, we employ a modified APF approach [13]. Let $F_g(l_i(t))$ be the gravitational force towards the goal, where $l_i(t) =< x_i(t), y_i(t) >$ denotes the position of robot i at time t, and let $F_r(l_i(t))$ be the resultant repulsive force from the obstacles and other robots within robot i's sensing range. Under the APF, robot i's control input is given by $u_i(t) = F_g(l_i(t)) + F_r(l_i(t))$.

In cluttered environments, robots may become trapped in local minima and oscillate around their current location indefinitely. Local minima may be detected by observing a robot's state or the forces applied to a robot. Formally, a local minima occurs if either of the following hold at time t:

$$|F_g(l_i(t)) + F_r(l_i(t))| < \varepsilon \text{ or } |l_i(t-\tau) - l_i(t)| < \rho s, \quad (2)$$

where $\varepsilon \in \mathbb{R}_{>0}$ and $\rho \in [0,1]$ are small thresholds, and s is the distance travelled by robot i in the past τ timesteps. The first inequality holds if the gravitational and repulsive forces are approximately zero. The second inequality holds if the robot's position has changed less than the distance it has travelled. When a local minima is detected, we remove the gravitational and repuslive forces and drive the robot along the wall of the nearest obstacle towards the formation location. The gravitational and repulsive forces are reapplied once the inequalities in (2) are violated.

Local minima may also occur due to robot deadlocks. For example, a robot may be blocked from its formation location by the repulsive forces of robots who have already reached theirs. For this, we combine first-order consensus [8] and leader-follower control [23]. Here, the blocked robot becomes the leader, and the other robots become followers. First-order consensus control ensures the robots satisfy the relative position relationships in the formation and converge to velocity consistency, thus driving them toward the target formation. Let $a_{ij} = 1$ if robot i can communicate with robot j, and 0 otherwise. Then, let $N_i = \{j \mid a_{ij} = 1\}$, i.e. the robots that robot i can directly communicate with. Under first-order consensus control [8], the control input of the leader robot i is set to be:

$$u_i(t) = F_g(l_i(t)) + F_r(l_i(t)) + \sum_{f \in N_i} w_{if}(t)(\boldsymbol{x}_{if}(t) - r_{if}), \quad (3)$$

where r_{if} is the required relative distance between the leader i and follower f, and $\boldsymbol{x}_{if}$ is the state of the follower f with respect to the leader i. The weight $w_{if}(t)$ is given by $w_{if}(t) = 2 - e^{-(x_{if}(t) - r_{if})^2}$. Similarly, the control input for follower f is as follows, where $u_i(t)$ is the control input of the leader:

$$u_f(t) = u_i(t) + F_r(l_f(t)) + \sum_{f' \in N_j} w_{ff'}(x_{ff'}(t) - r_{ff'}). \quad (4)$$

V. Formation Planning

In this section, we synthesise a discrete global path T for the centre of each formation which minimises the distance from its initial configuration z_s to the goal configuration z_g. For this, we incrementally construct a graph $G = (V, E)$ over the workspace, where each node $v \in V$ lies inside a convex polygon P, and edges $(v, v') \in E$ connect nodes. We begin with a heuristic approach for partitioning the workspace into convex polygons (Subsection V-A), which we then use for global path planning (Subsection V-B).

A. Heuristic Workspace Decomposition

The obstacle-free workspace F can be partitioned into a set of convex polygons P which maintain the formation's geometric constraints, defined as:

$$A_n v_m \leq b_n, \forall n = 1, ..., N, m = 1, ..., M, \quad (5)$$

where A_n and b_m are the parameters of the separated hyperplanes, N is the number of sides of the polygon, and v_m represents the outer vertices of the formation's convex hull. In Alg. 1, we show how to generate the next convex polygon P in a partition, given the existing polygons $\mathbb{P} = \{P_1, ..., P_K\}$. For this, we use IRIS [24], which given an initial seed point alternately solves two convex optimisations: (1) finding a set of hyperplanes that separate an ellipse e from the obstacles O via quadratic optimization, and (2) finding the largest ellipse within the polygon P via a semi-definite program. Each convex polygon P has a corresponding ellipse e; we denote the set of existing ellipses as $\mathbb{E} = \{e_1, ..., e_K\}$.

For efficient global planning (see Subsection V-B), polygon construction should be guided towards the goal. Therefore, in Alg. 1, we introduce a heuristic strategy for sampling new IRIS seed points. First, we discretise the workspace into cells, and assign a cost to each cell in a matrix B (lines 2-6). Cells in obstacles or existing ellipses have cost $-\infty$, and all other cells have cost:

$$Cost(c) = \frac{1}{K} \sum_{k=1}^{K} ||c - e_k||_2 - \gamma^{(1-\zeta)} ||c - g||_2. \quad (6)$$

Here, $\gamma \in \mathbb{R}_{>0}$ is a weight parameter, $\zeta \in [0,1]$ is the proportion of the map covered by obstacles and ellipses, and g is the cell containing goal configuration z_g. Intuitively, (6) assigns high cost to cells which are further from existing polygons and closer to the goal. However, the second term in (6) decays as the explored area increases, which admits backtracking to explore regions further from the goal. We select the cell c^* with maximum cost in B as the new seed point for IRIS (lines 9-10), pushing the polygons towards the goal. After running IRIS, we test whether the new polygon P_{new} and ellipse e_{new} should be accepted into the partition (lines 11-16). Polygon P_{new} is rejected if the seed point c^* is less than distance α from a previous seed point, or if e_{new} is less than distance β from an existing ellipse $e_k \in \mathbb{E}$. If P_{new} is rejected, Alg. 1 must be re-run. Thresholds α and β decay as the workspace is explored to relax the conditions for polygon acceptance, as the distance to the nearest cell or ellipse will decrease as more polygons are added.

B. Global Path Planning

Using Alg. 1, we incrementally construct a graph $G = (V, E)$ by adding convex polygons to the workspace until a path exists between z_s and z_g, where nodes are added for

Algorithm 1: Heuristic IRIS

Input: Obstacle set O, existing polygons $\mathbb{P}$ and ellipses $\mathbb{E}$, previously selected cells $\mathbb{C}$
Output: P_{new}

$P_{new} \leftarrow \varnothing$
foreach $c \in W$ **do**
  if $c \in O$ *or* $c \in \mathbb{E}$ **then**
    $B\{c\} \leftarrow -\infty$
  else
    $B\{c\} \leftarrow \text{Cost}(c)$

while $P_{new} = \varnothing$ **do**
  $c^* \leftarrow \arg\max_{c \in W} B$
  $\{P_{new}, e_{new}\} \leftarrow \text{IRIS}(c^*)$
  foreach $c \in \mathbb{C}$, $e_k \in \mathbb{E}$ **do**
    if $\|c^* - c\| < \alpha$ *or* $\|e_{new} - e_k\| < \beta$ **then**
      foreach $c \in e_{new}$ **do**
        $B\{c\} \leftarrow -\infty$
      $\{\alpha, \beta\} \leftarrow \text{UpdateThreshold}(\alpha, \beta, B)$
      $P_{new} \leftarrow \varnothing$

$\mathbb{P} \leftarrow \mathbb{P} \cup P_{new}$, $\mathbb{E} \leftarrow \mathbb{E} \cup e_{new}$, $\mathbb{C} \leftarrow \mathbb{C} \cup c^*$
return P_{new}

each polygon and polygon intersection. We then synthesise a path T over graph G. In detail, we do the following:

1) **Initialise Graph:** Graph G is initialised with nodes at z_s and z_g. Convex polygons are generated from each of these points using IRIS [24]. If these polygons intersect, z_s, z_g, and an intersection point are connected as in step 4.
2) **Test for Path:** If a path exists from z_s to z_g in G go to step 5, else go to step 3. If the total area covered by polygons exceeds a threshold, terminate without a plan.
3) **Add New Polygon:** Generate a new polygon P_{new} using Alg. 1. Then, add a node z_{new} to G at the location within P_{new} closest to the goal z_g. Formally:

$$z_{new} = \arg\min_{z \in P_{new}} (z - z_g)^2. \tag{7}$$

4) **Add Intersection Nodes and Edges:** Find all existing polygons $P_k \in \mathbb{P}$ who intersect with P_{new}. Given the nodes z_{new} and z_k computed from P_{new} and P_k respectively using (7), find the configuration z_{inter} which minimises the perpendicular distance from the straight line between z_{new} and z_k. Formally:

$$z_{inter} = \arg\min_{z \in P_{new} \cap P_k} dis(z, line(z_{new}, z_k))^2. \tag{8}$$

Next, add the node z_{inter} and edges (z_{new}, z_{inter}) and (z_{inter}, z_k) to G. By optimising z_{inter} using (8), we minimise edge length. Following this, return to step 2.
5) **Find Shortest Path:** Synthesise the shortest path from z_s to z_g on graph G using A* search [25].

We demonstrate our global planner in Fig. 4. In Fig. 4(a), a triangular formation must reach the top right of the map. Fig. 4(b) shows the convex polygons generated from the initial and goal configurations. These regions do not intersect, and so we use Alg. 1 to add a new polygon (see Fig. 4(c)). The intersections between these three polygons admit a graph which connects the initial configuration to the goal configuration, and so a global path can be found (see Fig. 4(d)).

VI. Trajectory Generation and Formation Control

The global path T computed in Subsection V-B contains sharp turns which cannot be executed smoothly by a robot. To mitigate this, we now present techniques for trajectory optimisation, trajectory tracking, and local motion planning.

A. Trajectory Optimization

To synthesise continuous collision-free trajectories from the global path T which respect robot kinematic constraints, we consider an extended minimum snap approach [6]. Minimum snap cannot be applied directly to formations, as the geometric constraints of the formation are ignored, which may cause collisions (see Fig. 4(e)). To avoid collisions, we apply the inequality constraints in (5), and minimise the deviation from global path T.

In minimum snap, trajectories are represented in S segments. The rth trajectory segment is represented as an order q polynomial over the current time t:

$$p_r(t) = [1, t, t^2, ..., t^q] \cdot \boldsymbol{p},\ t \in [0, t_f], \tag{9}$$

where $\boldsymbol{p}$ is the coefficient matrix of the trajectory polynomial. Here, $p_r(t)$ and $\dot{p}_r(t)$ represent the position and orientation of the formation's centroid, respectively. The time information for each segment is computed assuming a trapezoidal velocity-time profile.

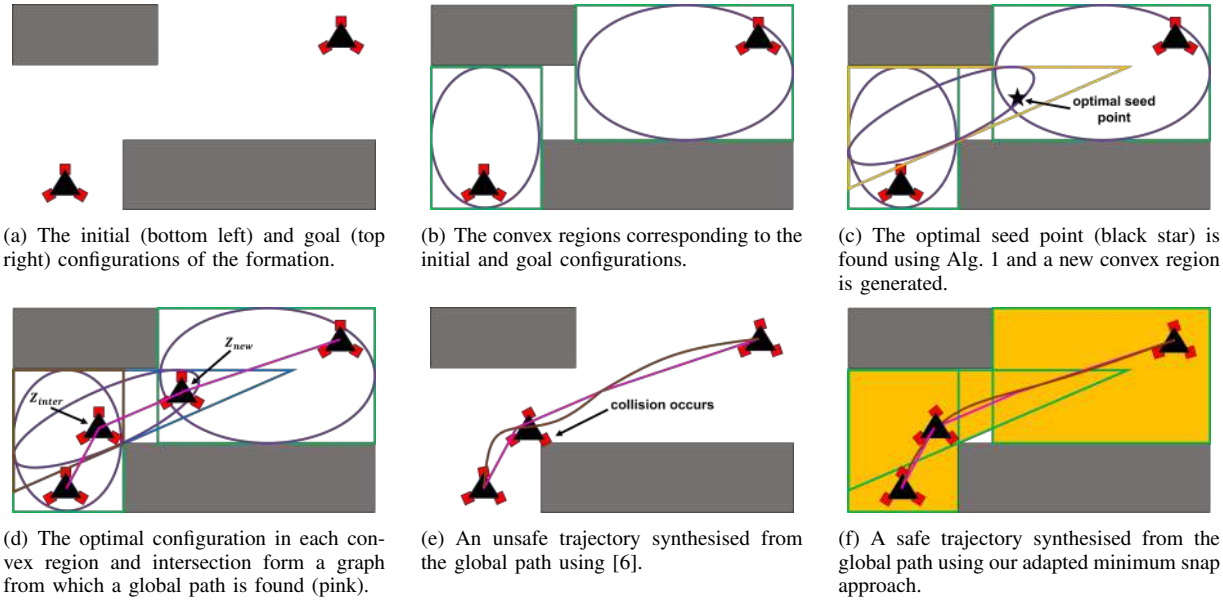
To adapt minimum snap to formations, we first include the inequality constraints in (5) during optimisation. This ensures the formation remains inside a convex polygon generated by IRIS [24], which guarantees collision avoidance. Second, we introduce an error term between the optimised trajectory and the trajectory corresponding to global path T, where T can be segmented based on the intermediate nodes in the path. The rth segment of the global path trajectory p_{o_r}, composed of endpoints $p_{t_{r-1}}$ and p_{t_r} can be represented as:

$$p_{o_r} = [p_{t_r} - \frac{p_{t_r} - p_{t_{r-1}}}{t_r - t_{r-1}} t_r, \frac{p_{t_r} - p_{t_{r-1}}}{t_r - t_{r-1}}, 0, 0, ..., 0]. \tag{10}$$

With this, the quadratic program for our extended minimum snap approach can be written as:

$$\min_{p} \sum_{r=1}^{S} \int_{t_{r-1}}^{t_r} (p_r^{(4)}(t))^2 + \lambda (p_r(t) - p_{o_r}(t))^2 \, dt \ \ s.t\ (5), \tag{11}$$

where λ is a weighting term. An example trajectory optimised with our approach is shown in Fig. 4(f).



(a) The initial (bottom left) and goal (top right) configurations of the formation.

(b) The convex regions corresponding to the initial and goal configurations.

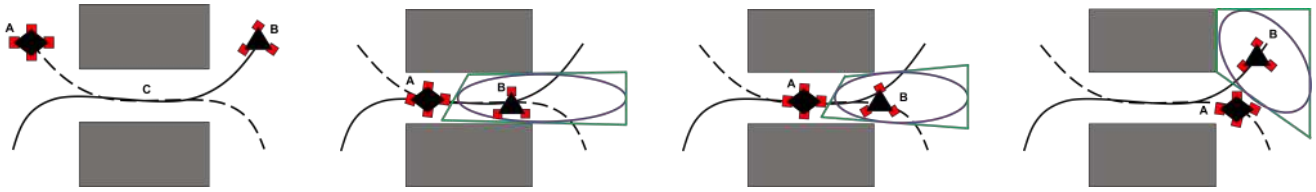
(c) The optimal seed point (black star) is found using Alg. 1 and a new convex region is generated.

(d) The optimal configuration in each convex region and intersection form a graph from which a global path is found (pink).

(e) An unsafe trajectory synthesised from the global path using [6].

(f) A safe trajectory synthesised from the global path using our adapted minimum snap approach.

Fig. 4. An illustrative example of global path planning and trajectory optimization for a triangular formation.



(a) Formations A and B are on track to collide at point C.

(b) Formation B treats formation A as an obstacle and computes its convex region.

(c) Formation B backs off to prevent a collision and computes the updated convex region.

(d) Formation A is eventually clear of B, and B returns to trajectory tracking.

Fig. 5. An illustrative example of our prioritised DWA for local motion planning and coordination. Solid and dashed lines represent robot trajectories.

B. Consensus-based Trajectory Tracking

To track the global trajectory, we combine pure pursuit control [9] with a consensus-based approach [4], where the formation centre is the virtual leader. For each robot, we use first-order consensus control to maintain the formation. Similar to (4), the control input for the ith follower of the jth formation at time t is given by:

$$u_i(t) = u(x_{ctrj}) + F_r(l_i(t)) + \sum_{f\in\{1,\dots,k\}} w_{if}(x_{if}(t) - r_{if}), \quad (12)$$

where x_{ctrj} is the state of the virtual leader, $u(x_{ctrj})$ is the virtual leader's control input, and there are k robots in the formation.

C. Local Motion Planning Using DWA

When one formation senses another, it should deviate from its trajectory to prevent collisions. For this, we employ a prioritised DWA, where formations with longer trajectories have higher priority [26]. This is an ad-hoc solution for the coordination component of MFPC (see Fig. 2). Formation j begins collision avoidance behaviours upon sensing a higher priority formation. First, formation j computes the convex region around its virtual leader given the other formations and surrounding obstacles. We then sample a set of motion primitives m using DWA [7], i.e. velocities and angular velocities, which respect kinematic constraints and keep the formation within its convex region. The highest value motion primitive according to a function E is then executed, where E is defined as in [7]:

$$E(m) = w_1 \cdot Dir + w_2 \cdot Dis + w_3 \cdot Vel. \quad (13)$$

Here, Dir is the absolute directional change between the current velocity and the velocity in m, Dis is the average distance between formation j and any higher-priority formations after executing m, and Vel is the magnitude of the velocity in m. Formations execute collision avoidance primitives until higher priority formations are out of range, and then switch back to trajectory tracking. This ad-hoc coordination approach does not guarantee collision avoidance, as formations may be occluded, but collisions are reduced, as shown in Sec VII. We demonstrate our prioritised DWA in Fig. 5. In Fig. 5(a), formations A and B are on track to collide, where A has higher priority. Formation B computes its convex region (see Fig. 5(b)), and backs off to prevent a collision (see Fig. 5(c)). This is repeated until formation B is clear of A, and switches back to trajectory tracking (see Fig. 5(d)).

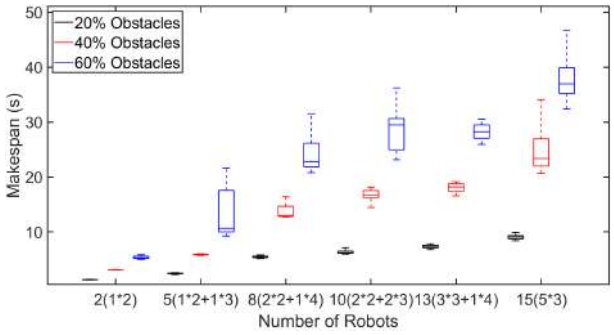


Fig. 6. Formation generation performance as the number of robots increase.

VII. Experiments

In this section, we demonstrate the efficacy of our approach in simulation. All experiments are implemented in MATLAB on an Intel i5 processor at 2.3GHz with 16GB of RAM. We consider 3 10×10m environments with randomly generated quadrilateral obstacles which cover 20%, 40%, and 60% of the environment respectively. All robots are 0.3m squares, where the velocity, angular velocity, acceleration, and angular acceleration are constrained within $\pm 0.3\,$m/s, $\pm 0.35\,$rad/s, $\pm 0.2\,$m/s^2, and $\pm 0.8\,$rad/s^2 respectively. Formations may be linear, triangular, or rectangular (see Fig. 3).

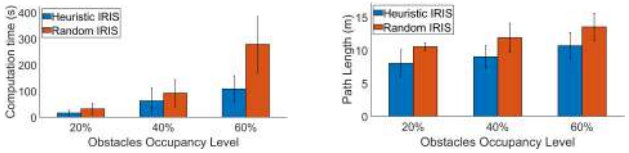
A. Formation Generation Performance

First, we evaluate formation generation performance as the number of robots increase. For each environment and number of robots, we generate 30 random problem instances, i.e. start and goal locations. We evaluate the makespan, i.e. the time for the last robot to reach its formation location. We present our results in Fig. 6, which also shows the formation configuration for each number of robots, e.g. $8(2*2+1*4)$ represents 8 robots forming 2 straight lines and 1 rectangle.

In Fig. 6, the makespan increases as the environment becomes more occluded, as robots must take longer routes, and because robot coordination is harder in tighter spaces. However, the makespan increases slowly with the number of robots, demonstrating how our approach effectively coordinates robots towards their formation locations. In all experimental runs, no robots became trapped in local minima.

B. Formation Planning Performance

Next, we evaluate our global formation planner against a variant which uses IRIS [24] with random seed points. In each environment, we randomly generate 30 formation start and goal locations for each of the three formation shapes, where start and goal locations are at least 5m apart. We combine the results for all formation shapes and present them in Fig. 7, where the area threshold in step 2 in Subsection V-B is set to 80%. Our planner consistently synthesises paths quicker than the random IRIS planner, where the gap increases with the number of obstacles. Our approach also synthesises shorter paths using Alg. 1 and the optimal configurations within each polygon and intersection.



(a) Computation time results. (b) Path length results.

Fig. 7. Global path planning performance as occupancy increases.

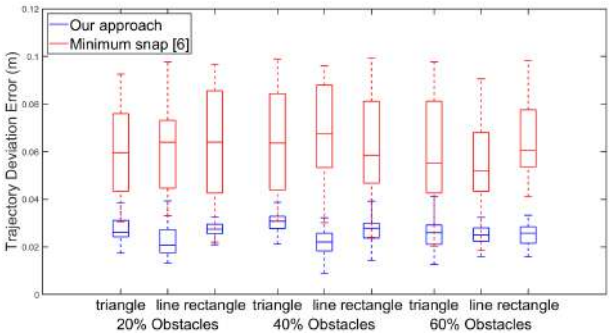


Fig. 8. The average trajectory deviation error of our approach against minimum snap [6].

We also demonstrate the efficacy of our extended minimum snap approach in Section VI-A. Using the global paths synthesised in the previous experiment, we run our trajectory optimisation method and the original minimum snap algorithm in [6]. For each run, we record the average trajectory deviation error, and present the results in Fig. 8. Our approach synthesises trajectories which are consistently easier for robots to track. We also recorded the total collisions in each trajectory: our approach produced collision-free trajectories across all runs, whereas minimum snap trajectories intersected with obstacles at least 4.6 times on average. This is because our approach is guaranteed to satisfy the formation's geometric constraints.

C. Formation Coordination Performance

We now evaluate our prioritised DWA for local motion planning and coordination. We generate 30 random 2-5 formation problems in the 60% occluded environment, and evaluate the number of inter-formation collisions with and without our approach. For each problem instance, the initial and goal formation positions are on either side of a circle of radius 5m centred on the map. We present our results in Table I. Our prioritised DWA resolves all conflicts for 2-3 formations, but not for 4-5 formations. This is because there may not exist feasible collision-free controls for lower priority robots in narrow regions of the environment. Despite this, our approach still reduces the total number of collisions.

VIII. Conclusion

In this paper, we proposed a framework for MFPC that outperforms state-of-the-art techniques. For formation generation, we extended APF approaches to avoid local minima. For formation planning, we applied a heuristic sampling strategy to improve path quality and computation

TABLE I

NUMBER OF INTER-FORMATION COLLISIONS WITH AND WITHOUT OUR PRIORITISED DWA.

Number of Formations	No Prioritised DWA	With Prioritised DWA
2 $(1*3+1*4)$	0.62 ± 0.48	$\mathbf{0}$
3 $(2*3+1*4)$	1.56 ± 0.93	$\mathbf{0}$
4 $(2*3+2*4)$	1.94 ± 0.89	$\mathbf{0.61 \pm 0.44}$
5 $(1*2+2*3+2*4)$	2.47 ± 1.16	$\mathbf{0.92 \pm 0.69}$

time. Finally, for formation coordination, we introduced a prioritised DWA to reduce inter-formation collisions. In future work, we will demonstrate the efficacy of our framework on real robots, explore coordination methods that guarantee inter-formation collision avoidance, and capture the effects of uncertainty on robot execution. Further, we will consider formations of real nonholonomic robots in transport environments shared with humans.

REFERENCES

[1] E. Tuci, M. H. Alkilabi, and O. Akanyeti, "Cooperative object transport in multi-robot systems: A review of the state-of-the-art," *Frontiers in Robotics and AI*, vol. 5, p. 59, 2018.

[2] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quaterly*, vol. 2, pp. 83–97, 1955.

[3] J. Wang, X. Wu, and Z. Xu, "Potential-based obstacle avoidance in formation control," *Journal of Control Theory and Applications*, vol. 6, pp. 311–316, 2008.

[4] Z. Peng, G. Wen, A. Rahmani, and Y. Yu, "Distributed consensus-based formation control for multiple nonholonomic mobile robots with a specified reference trajectory," *International Journal of Systems Science*, vol. 46, no. 8, pp. 1447–1457, 2015.

[5] R. Falconi, L. Sabattini, C. Secchi, C. Fantuzzi, and C. Melchiorri, "Edge-weighted consensus-based formation control strategy with collision avoidance," *Robotica*, vol. 33, no. 2, pp. 332–347, 2015.

[6] Z. Ma, H. Qiu, H. Wang, L. Yang, L. Huang, and R. Qiu, "A* algorithm path planning and minimum snap trajectory generation for mobile robot," in *Proceedings of the International Conference on Robotics, Control and Automation Engineering (RCAE)*, pp. 284–288, IEEE, 2021.

[7] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[8] K. Moore and D. Lucarelli, "Forced and constrained consensus among cooperating agents," in *Proceedings of IEEE Networking, Sensing and Control*, pp. 449–454, 2005.

[9] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," tech. rep., Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[10] Y. Mohan and S. Ponnambalam, "An extensive review of research in swarm robotics," in *Proceedings of the World Congress on Nature & Biologically Inspired Computing (NABIC)*, pp. 140–145, IEEE, 2009.

[11] E. Bahceci, O. Soysal, and E. Sahin, "A review: Pattern formation and adaptation in multi-robot systems," tech. rep., Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-03-43, 2003.

[12] M. Machida and M. Ichien, "Consensus-based artificial potential field approach for swarm," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pp. 6671–6677, IEEE, 2021.

[13] Z. Wu, W. Su, and J. Li, "Multi-robot path planning based on improved artificial potential field and b-spline curve optimization," in *Proceedings of the Chinese Control Conference (CCC)*, pp. 4691–4696, IEEE, 2019.

[14] G. Lee and D. Chwa, "Decentralized behavior-based formation control of multiple robots considering obstacle avoidance," *Intelligent Service Robotics*, vol. 11, pp. 127–138, 2018.

[15] J. C. Barca and A. Sekercioglu, "Generating formations with a template based multi-robot system," in *Proceedings of the Australasian Conference on Robotics and Automation*, 2011.

[16] T. Recker, H. Lurz, and A. Raatz, "Smooth spline-based trajectory planning for semi-rigid multi-robot formations," in *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1417–1422, IEEE, 2022.

[17] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[18] S. K. A. De Sousa, R. C. S. Freire, E. Á. N. Carvalho, L. Molina, P. C. Santos, and E. O. Freire, "Two-layers workspace: A new approach to cooperative object transportation with obstacle avoidance for multi-robot system," *IEEE Access*, vol. 10, pp. 6929–6939, 2022.

[19] M. U. Farooq, Z. Ziyang, and M. Ejaz, "Quadrotor uavs flying formation reconfiguration with collision avoidance using probabilistic roadmap algorithm," in *Proceedings of the International Conference on Computer Systems, Electronics and Control (ICCSEC)*, pp. 866–870, IEEE, 2017.

[20] K. H. Kowdiki, R. K. Barai, and S. Bhattacharya, "Leader-follower formation control using artificial potential functions: A kinematic approach," in *Proceedings of the IEEE International Conference On Advances In Engineering, Science And Management (ICAESM)*, pp. 500–505, IEEE, 2012.

[21] D. Koung, O. Kermorgant, I. Fantoni, and L. Belouaer, "Cooperative multi-robot object transportation system based on hierarchical quadratic programming," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6466–6472, 2021.

[22] H. Sang, Y. You, X. Sun, Y. Zhou, and F. Liu, "The hybrid path planning algorithm based on improved a* and artificial potential field for unmanned surface vehicle formations," *Ocean Engineering*, vol. 223, p. 108709, 2021.

[23] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[24] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Proceedings of Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, pp. 109–124, Springer, 2015.

[25] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[26] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 430–435, IEEE, 2005.

Impact of UAV Propellers on Gas Plume Tracking

Rui Baptista,[1] Hugo Magalhães[1] and Lino Marques[1]

***Abstract*—Unmanned Aerial Vehicles (UAVs) are highly manoeuvrable platforms that are becoming popular to map and localise the source of chemical substances in the environment. However, their propellers disturb the environment and may seriously impact the measurements and the expected results in such applications. Some works have already addressed this problem, but only for specific cases and proposing different solutions about the best way to mitigate this problem. This work studies the impact of propellers for multiple sensor positions and 3 UAVs of different sizes, in an Odour Source Localisation (OSL) problem under field conditions and compares the results with a non-disturbing setup. The experiments show a significant impact on the measured signals from asymmetrical frames and from the motion of the agent, with the best results achieved from sensors acquiring data on intake air regions.**

I. INTRODUCTION

When a harmful gas leak occurs, the volatile substances introduce in the environment are carried by the wind and dispersed over a potentially vast area, endangering humans and other living organisms. In these situations it is very important detecting the dangerous gas and locating its source in a shortest time in order to mitigate the risks. This operation is usually carried-out with specialised human teams but it can also be pursued by mobile robots, equipped with adequate gas sensors and control algorithms. This approach has the advantages of moving human operators from dangerous areas and , eventually, reducing the requirements for human resources, but the task is not trivial since the fluctuating and unpredictable nature of the dispersion phenomena breaks the odour into small patches making them harder to detect as the distance to the source increases.

Marques et al. [1] contributed to pioneering work on Mobile Robot Olfaction (MRO) endowing wheeled platforms with the ability to sense, distinguish and locate odour sources. Recently, Unmanned Aerial Vehicles (UAVs), also known as drones, have been employed for such tasks due to their increased manoeuvrability. However, choosing the best sensor position onboard the UAV can be a challenging endeavour due to the strong airflow caused by the propulsion system to move the platform. This distortion can corrupt the measured signal affecting the quality of the observations or reduce the number of odour encounters by pushing the chemicals away from the sensor. One of the approaches to determine an optimal sensor position relies on Computational Fluid Dynamics (CFD) simulations to evaluate the motion of the fluid in the vicinity of the drone [2]. Other experiments consider dispersion and rotor wake models to calculate the distortion of the gas plume inducted by the propellers of the UAV [3], [4]. One limitation of these approaches is that they do not consider the chemical signal, focusing only on the regions with the least flow distortion or high number of odour encounters, however, increasing the number of positive chemical contacts does not necessarily lead to quality measurements. An alternative process consists on laboratory experiments with Particle Image Velocimetry that combine smoke emitters and vision-based systems to highlight the path of the smoke [5]. A ventilator generates a strong airflow pushing the smoke plume towards the UAV frame which is fixed at a vertical pole or hovering close to the source of emission. By analysing the motion of the smoke, multiple conclusions can be drawn such as the plume being pulled by the propellers if the UAV is below the smoke, or being pushed towards the ground with the UAV above the plume, reducing the chance of contact with the target gas. While insightful, the drawback is that, typically, the propulsion system is programmed to operate at hover speed, with the propellers equally rotating. In order to move, the drone needs to change the velocity of the propellers which can originate complex flow interactions [6] and potentially lead to different results to the ones taken while stationary.

Fig. 1. Testing environment. The image shows the odour source in the left-top corner, a Hexa-Copter UAV in the middle and an ultrasonic anemometer in the right-bottom corner.

Independently of the approach, each author proposes a different sensor position in order to avoid a negative impact on the measurement process, with positions varying from under the UAV [7] to the front of the frame [8]. According to the literature, there are both advantages and disadvantages of placing the sensors on top or bottom of the drone [9], which may depend on the application and the type of sensor to be used. Particulate Matter sensors can make use of the increased flow below the propellers to capture chemicals towards the

[1]All authors are with the Institute of Systems and Robotics, Department of Electrical and Computer Engineering, University of Coimbra, 3030-290 Coimbra, Portugal {rui.baptista, hugo.magalhaes, lino}@isr.uc.pt

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

measurement chamber while Metal oxide gas (MOX) sensors can be exposed to additional noise in this region due to the stronger flow that can affect both the heating element and metal oxide layer corrupting the signal. Hence, evaluating more than one sensor position may provide broader insights that can also be used to validate some of the conclusions from the flow analysis. Here, only a small number of works [5], [10], [11] compare the performance of different sensor locations during the same experiments. Furthermore, a single UAV frame is employed during most of the studies, which is usually a pocket drone [7]. The smaller frame dimensions require a less complex infrastructure for the testing scenario, however, these frames cannot operate in field environments due to limited autonomy and payload capacity. Larger and heavier platforms require additional flow displacement to move while different frame configurations can produce distinct airflow patterns which require further validation in order to avoid measurement disturbances [12].

This leads to the first question: How do different UAV frames perform during similar OSL missions? The second question rise on the fact of the experiments being performed within small indoor environments under specific environmental conditions, with only a limited number of field experiments [13], [14]. How well does the location of the sensor, deemed as good or optimal within laboratory studies, perform during OSL missions under real field conditions?

This work aims to tackle these questions by performing multiple OSL missions under realistic field conditions (Figure 1) with the goal of evaluating: (1) the impact of different UAV frames and (2) the impact of different sensor positions during the search process. The results are compared with a non-disturbing reference platform operated by a human agent moving to the positions provided by the searching algorithm.

II. Methods and Materials

Consider an aerial mobile agent moving in known locations $\mathbf{p_a} = (x, y, z) \in \mathbb{R}^3$ with the capability to sense for chemical measurements $c(\mathbf{p_a}, t)$ at time step $t \geq 0$ s. The agent is characterised by their frame geometry Φ, number of rotors $\mathbf{n_r}$, propeller diameter $\mathbf{d_p}$, length between opposing rotors $\mathbf{l_a}$, mass $\mathbf{m_a}$, maximum payload $\mathbf{m_{py}}$ and flytime $\mathbf{t_{fly}}$. The location of the chemical sensor related to the agent body is $\mathbf{p_s} = (x_a, y_a, z_a) \in \mathbb{R}^3$.

The goal is to evaluate the impact of UAV propellers in OSL. To achieve this goal, a set of OSL experiments is performed with different drones in similar conditions. The results are evaluated and compared with experiments accomplished by a non-intrusive platform guided by a human operator.

A. Odour Source Localisation

A localisation process relies on observed data to pinpoint the source position. This raises the question of where should the robot move to in order to collect valuable information. This mission is usually accomplished by two tasks: (1) plume searching and (2) plume tracking. Plume searching consists of finding the active region of the plume, and in this work, a series of zig-zag motion patterns are adopted in the upflow direction. After making contact with the plume, the tracking procedure tries to follow the odour towards the source of emission and declare its position. Cognitive strategies are reliable approaches to operate in turbulent environments, where a belief of the source parameters is estimated from assimilated data and a movement decision guiding the agent towards informative locations quantified by a cost function. This work adopts the same framework as in [15] where the belief is approximated by a Particle Filter, with a likelihood function modulated by a Normal distribution and a Gaussian Plume dispersion model. The decision process quantifies the expected information gain through an Entropy cost function.

Fig. 2. A field experiment with the human operator. On the left, a human operator guides the Hexa-Copter UAV attached to a vertical pole. On the right, a controllable odour source emits acetone at 3m from the ground.

B. Odour Source

The odour plume emerges from a continuous release of acetone vapour, an organic compound with the formula $(CH_3)_2CO$. The liquid acetone, in a reservoir, is converted to a gaseous state through a controllable piezoelectric transducer and an adjustable 120 mm fan pushes the acetone vapour over a polyvinyl chloride chimney releasing the chemical at approximately 3 meters from the ground (Figure 2). The emission rate was set to 784 g/h, approximately 7 times less when compared with other studies (e.g. Hutchinson et al. [16] defined a release rate of 5.4 kg/h).

C. Mobile Platforms

TABLE I
Characteristics of multi-rotor UAVs.

UAV	Φ	$\mathbf{n_r}$	$\mathbf{d_p}$	$\mathbf{l_a}$	$\mathbf{m_a}$	$\mathbf{m_{py}}$	$\mathbf{t_{fly}}$
Quad450	X	4	279mm	450mm	1.2kg	1.5kg	10min
Quad860	V	4	406mm	860mm	1.8kg	3kg	20min
Hex980	X	6	406mm	980mm	3.5kg	7kg	25min

Three different multi-rotor UAVs based on some of the most popular platforms from the literature are used for this

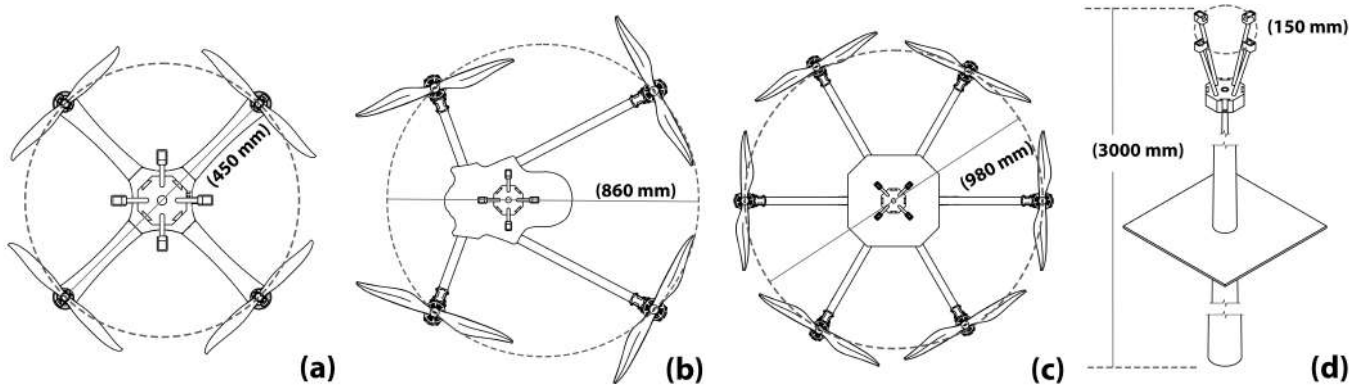


Fig. 3. Tested platforms with the measurement system attached to the top. (a) Quad450 UAV with 450mm of diameter. (b) Quad860 platform measuring 860mm. (c) Hexa-Copter Hex980. (d) Non-disturbing platform.

study, however, smaller UAVs such as nano-copters are not considered due to their inability to fly outdoors [17]. Table I summarises the physical characteristics of three UAVs where the Small quad-copter (Figure 3a), Medium-size quad-copter (Figure 3b) and Hexa-copter (Figure 3c) are renamed as Quad450, Quad860 and Hex980, respectively.

The X frame consists on a structure with all rotors located at the same distance from the center of the frame resulting in a symmetrical shape. On the V frame, the structure geometry is asymmetric, meaning that the rotors at the front are not at the same distance from the centre of the frame as the rotors from the back.

All platforms are equipped with a Pixhawk autopilot and a small board computer (SBC). A fourth non-disturbing platform consisting of a vertical pole is used as reference (Figure 3d). The measurement system is attached to the top at the same height as the source and shares the same hardware as the UAVs, however it is manned by a human operator moving to the goals generated by the localisation algorithm.

D. Measurement System

Metal oxide gas sensors (MOX) are commonly used to detect chemicals. It is a small and low-cost solution, composed of a resistive transducer that decreases its resistance in atmospheres that contain oxidising vapours. This sensor can detect few parts per million of vapour and has a rise time of about 10 ms and a fall time of 1-2 s [18]. The measurement system is composed of a total of four MiCS-5524 MOX sensors from SGX Sensortech Ltd, whose signals are acquired at 100 Hz by a 12-bit Analog-to-Digital converter. The measurement system is evaluated in four different $\mathbf{p_s}$ locations (Figure 4), where two positions are above the propellers and the remaining two below the propellers:

1) Top: Sensors are installed 30 cm above the centre of the frame, and spaced 15 cm apart (Figure 4a). This position aims to understand the impact of placing the sensors outside the region of influence of the airflow.
2) Upper middle: Sensors installed above the propellers (Figure 4b) and fixed at the centre base of the frame spaced 30 cm apart. The goal is to analyse the influence of the airflow intake effect.
3) Lower middle: Sensors installed directly below the propellers (Figure 4c), 50 cm apart. The air pulled from the propellers recirculates and mixes with the gas due to the vortexes generated in this region. This re-circulation can improve the detection rate by repeatedly pushing the gas towards the sensors.
4) Bottom: Sensors installed bellow the propellers (Figure 4d), 30 cm from the centre of the frame and with horizontal configuration identical to the upper middle position. It is a region with minimal disturbance due to the structure of the platform, which also has less airflow than the lower middle position.

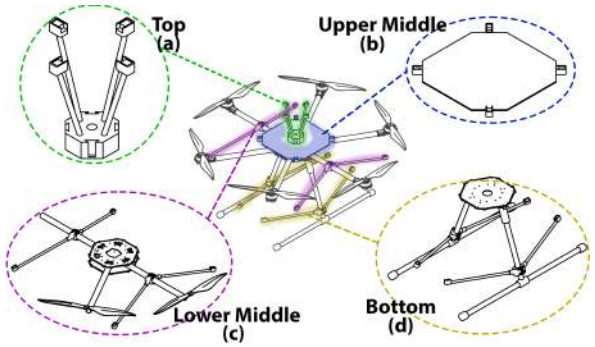


Fig. 4. Measurement system scheme composed of four sensors. (a) Top scheme. (b) Upper middle scheme. (c) Lower middle scheme. (d) Bottom scheme.

E. Evaluation Metrics

To evaluate the impact of each configuration on the OSL performance, four metrics are employed:

- Success Rate: A trial is considered a success when the estimated source position is less than 5 meters of distance to the true source location. This metric shows the overall performance of the OSL but does not provide enough details about its efficiency.
- Detection Rate: The ratio between the number of positive odour detections and the total sampled data. A low de-

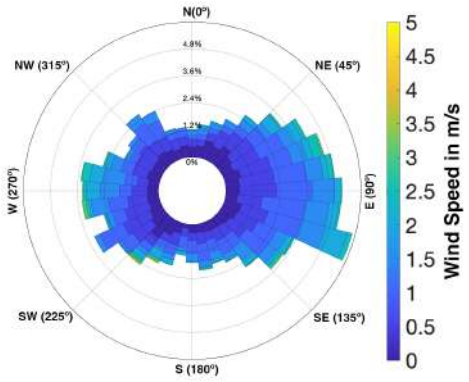


Fig. 5. Speed and direction characterisation of the wind at the testing environment.

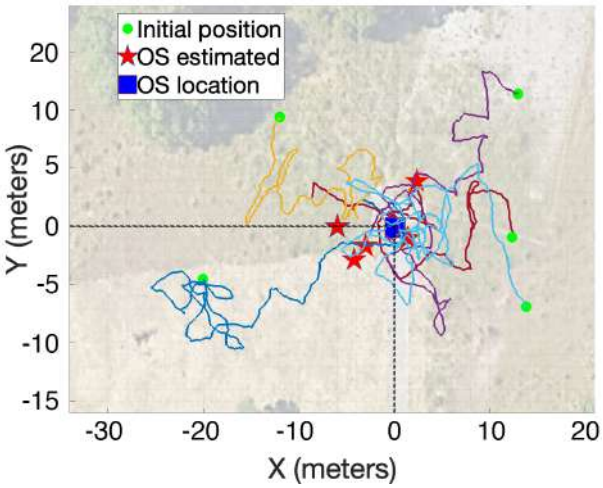


Fig. 6. Five OSL experiments. The initial position of the search (circle) starts opposite the measured wind direction. Also is shown the estimated position of the odour source (star) and the real position of the source (square).

tection rate may indicate a high measurement disturbance guiding the agent to less informative locations.

- Accumulated Distance: The total travelled path required to find the source. It is the overall efficiency metric that shows the energy required to perform each experiment.
- Distance Rate: The percentage of the accumulated distance allocated to the search and tracking tasks. Long distances during the search stage may indicate external interference's and a long tracking stage can suggest significant measurement disturbances.

Additionally, other performance indicators can be directly observed, such as high-frequency components in the acquired signal, resulting from very turbulent air mixing near the gas sensors.

F. Evaluation Methodology

The methodology adopted in this study is divided into two phases:

1) UAV Platform: To evaluate the impact of the UAV platform, first, OSL experiments are performed with the non-disturbing platform (Figure 3d). It acts as a reference for the remaining experiments since there are no disturbances generated by the propellers. Second, each UAV platform is coupled to the same vertical pole with the human operator moving the structure. The rotors are programmed to rotate at the minimum required velocity for the platform to hover. Since each platform has different weights and propeller placements, the required velocities are also different which enables quantifying the respective impact on OSL efficiency. During these trials, the measurement system is placed on the top position (Figure 4a) restricting the analysis to the impact of the airflow and the structural characteristics.

2) Sensor Positioning: The least disturbing platform identified during the previous experiments is consequently used to evaluate the impact of the sensor positioning. Here, each position scheme from Figure 4 is performed under fully realistic conditions with the UAV performing in a fully autonomous mode detached from the vertical pole, where all the motion dynamics are taken into account.

III. Field Experiments

The experiments are performed in a natural field region of approximately 70x50 m^2. It is composed of trees and small hills that produce different dispersion phenomena from the ones of indoor environments. The wind is monitored with two WindSonic ultrasonic anemometers from Gill Instruments installed in the testing area. A wind characterisation over an extensive time period (Figure 5) showed an unstable behaviour with variations of 360º under low wind speeds. The average wind speed is approximately 2.5 m/s in most of the experiments, flowing mostly from the east direction with variations of 90º. The testing area is covered by a wireless network and a Real-time Kinematic base station responsible for sending position corrections to the mobile platforms. The searching platforms execute onboard the same OSL algorithm as in [15], running Robot Operating System framework on SBC Intel NUC-i5 with 16Gb of RAM. The OSL algorithm framework uses all observed values between movement goals in the inference process. As the platforms are equipped with four chemical sensors, the chemical measurement at each time step t results from the maximum concentration of the four sensors. Also, the framework was modified to use the wind speed and direction as observations allowing a more accurate convergence and a considerably less number of possible plume combinations. Due to the random nature of the wind direction, the platforms are initialised near one of the edges of the searching space, opposite to the measured direction at the actual instance. The maximum initial distance from the odour source is 25 meters. The agent then starts searching for the plume with a sequence of zig-zaging motion patterns, switching to the tracking strategy after a positive odour contact. The linear speed of the platform was set to 0.5 m/s and the measurement data was collected at a fixed height of 3 meters over the ground. The work is evaluated over 20 experiments per platform and sensor scheme.

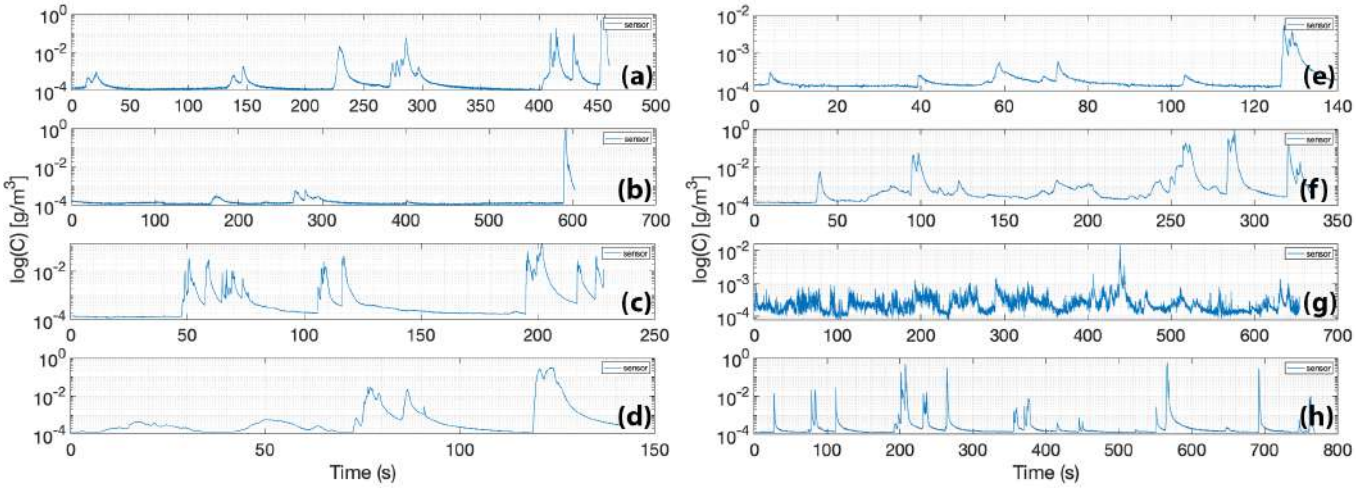


Fig. 7. Data sampled of one experiment for each platform and sensor scheme evaluated (Concentration in logarithmic scale). (a) Data of Reference platform. (b) Data of Quad450 UAV. (c) Data of Quad860 platform. (d) Data of Hex980 UAV. (e) Data of Top scheme positioning. (f) Data of Upper Middle scheme positioning. (g) Data of Lower Middle scheme positioning. (h) Data of Bottom scheme positioning.

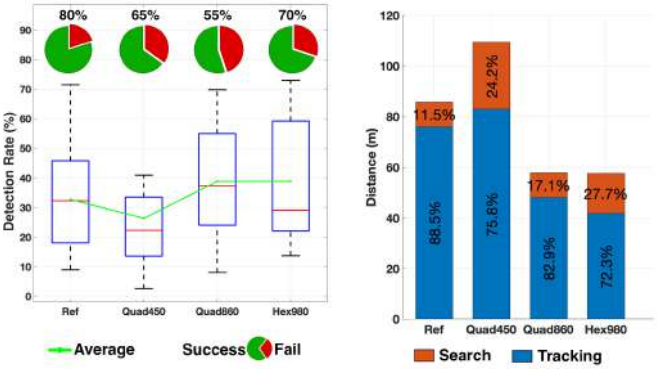


Fig. 8. Statistics of field experiments in human operator mode. Left graphic show the success rate and detection rate statistic. The right plot shows the accumulated distance and the percentage allocated to the search and tracking tasks.

IV. Results

Figure 6 shows five OSL experiments where it is visible the initial location of the drone, the trajectory and the real and estimated odour source localisation. Due to the nature of the wind, the drone is initialised in different positions, opposite the measured wind direction.

From Figure 7a, the sample data from the reference platform shows low fluctuations and an increasing mean concentration as the platform moves closer to the odour source. This platform achieved a success rate of 80% and a detection rate of approximately 32%. On average it required approximately 86 meters of travelled distance to declare the source position and spent 11.5% of the distance searching for the first odour contact (Figure 8 Ref).

A. UAV Platform

The Quad450 UAV had a success rate of 65% and a detection rate 26% lower than the reference (Figure 8 Quad450). The reduction of the detection rate can be explained by the lower energy efficiency of the platform. The weight of the platform is close to its maximum payload, requiring a higher rotor speed to hover which in term pushes the chemicals away from the sensors. The travelled distance also increased by 24 meters relative to the reference prompted by an inconsistent mean concentration (Figure 7b). The same Figure suggests the presence of external interference on measured data since the concentration magnitude decreased far away from the odour source, and close to the odour source has values similar to the reference. Analysing the wind, it was observed a high wind direction fluctuation, flowing mostly from East to West with variations of 360º and a wind speed between 1 and 2 m/s. These variations lead to a higher search distance and caused a negative impact on the estimation process. The highly dynamic nature of the environment can result in deviations on the belief map due to the mismatch between predicted and observed data. The second UAV (Figure 8 Quad860) beat the reference and the Quad450 in both the detection rate (39%) and travelled distance (58 meters), although the success rate is 25% lower than the reference and 10% lower than the previous UAV platform. These results show that an increase in the detection rate does not necessarily lead to a better localisation process again, due to significant deviations between the instantaneous observed data and the average value from the dispersion model. The mismatch is noticeable in Figure 7c where it is seen that odour comes into contact with the sensors but the overall concentration is lower and the mean concentration does not increase over time. The results suggest that this platform has a high impact on the measurement system which may be explained by the asymmetrical V shape. This type of frame (Figure 3b) creates an unbalanced flow in the location of the measuring system since the distance of each rotor to the centre of the sensors is not equal. Lastly, the Hex980 had the best success rate of the three drones with a value of 70% and a 7% increase of the detection rate relative to the reference. On average it travelled 57 meters to find the

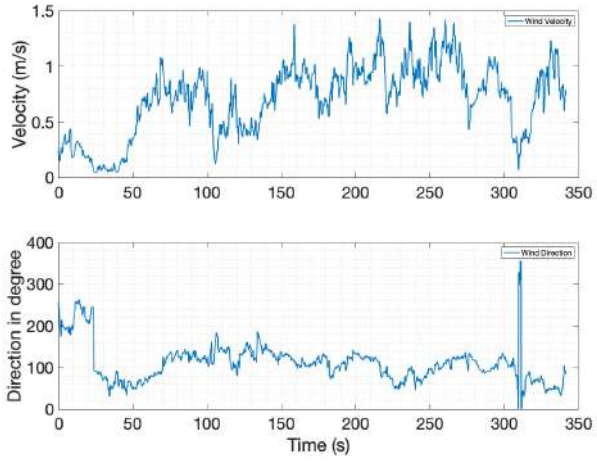


Fig. 9. The wind velocity and direction of an experiment. The velocity and direction change rapidly at instants 30 and 310 seconds due to the highly dynamic nature of the environment.

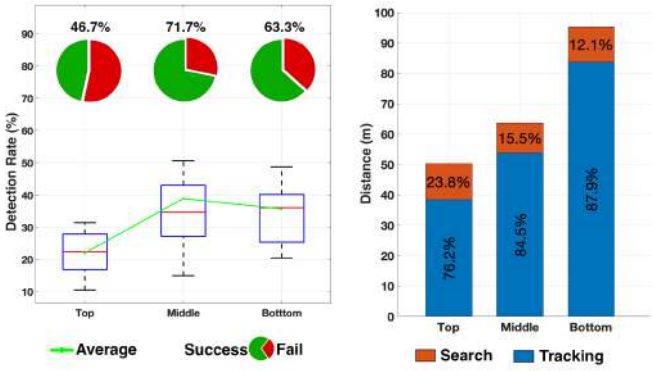


Fig. 10. Statistics of field experiments with Hexacopter. Left graphic show the success rate and detection rate statistic. The right plot shows the accumulated distance and the percentage allocated to the search and tracking tasks.

source with 27% of that value spent trying to find the plume. Since the performance is similar to the Quad860, the results (Figure 8 Hex980) indicate a higher measurement quality, that can be related to the uniform airflow of the 6 rotors around the sensors. The same quality can also be observed in Figure 7d where the amplitude of the signal is similar to the reference i.e., no disturbances from the motion of the propellers are observed.

The increase of the search distance is related to the wind condition (Figure 9 at the beginning of the experiments because the UAV was not placed opposite to the wind direction. It was observed that, in the first stages, the wind was approximately 0.3 m/s from South-West but it rapidly increased to 1.2 m/s changing the direction to South-East. Under the studied conditions, a higher detection and success rate combined with a lower travelled distance makes the Hex980 the most efficient platform for OSL. It outperformed the reference on both the detection rate and travelled distance, however, the success rate was 10% lower than the same reference.

B. Sensor Positioning

Figure 10 shows the results from the studied position schemes with the Hex980 UAV in fully autonomous mode. While four positions were analysed, OSL experiments were only possible with three configurations, because the measured signal was highly corrupted by the disturbance induced by the downflow of the propellers. This will be discussed during the course of this section.

The top scheme position (Figure 4a) which is the same as the previous experiments, had a success rate of 46.7% and a detection rate of 22.5%. Both these values were worst than the ones performed with the same frame manned by the human operator, except for the travelled distance which was similar. Comparing the Figure 7d and 7e, the UAV in fully autonomous mode reduces the magnitude of the measured concentration, impacting the overall performance. A huge improvement was observed in the upper middle scheme (Figure 4b) achieving the best success rate of 71.7%, excluding the reference. The high success rate and detection rate indicate an improvement in the quality of the measured values although it was observed an unpredictable increase in the travelled distance. From Figure 7f, the magnitude values are similar to the reference (Figure 7a) but an unexpected behaviour is observed at the instant 325s where the sensor's recovery does not reach the baseline. This behaviour may justify a higher travelled distance due to an increase of false detection's, resulting from changes in the sensor baseline, which in term increases the mismatch between the predicted and the observed data. The third position (Figure 4c) did not allow to perform the same number of trials as can be seen from Figure 7g, where high-frequency components in the acquired signal appeared due to the turbulence level generated by the motion of the propellers. A possible explanation is related to the technology of the sensor. The gas-sensing layer of the MOX sensor is warmed by a platinum heater responsible for targeting the sensitivity to specific gases. When subject to the strong airflow produced by the propellers, the heat dissipation of the platinum heater changes, which generates an unstable reaction when the chemical contacts the crystal, corrupting the measured signal. This observation may also prove the unexpected baseline behaviour in the upper middle position. The bottom region of the UAV (Figure 4d) is less vulnerable to heat dissipation. The distance of the sensors with respect to the frame attenuates the direct impact of the airflow, maintaining the sensor baseline over time (Figure 7h). These conditions result in a detection rate of 36%. The success rate outperforms the top scheme with 63.3% but less 7.8% than the upper middle position. The travelled distance increased by more than 48% indicating a slower convergence of the odour source estimation. In fact, the re-circulation pushes the gas repeatedly towards the sensors, increasing the detection rate of the target chemical. This can have a positive factor on localisation strategies that rely on binary detection's, since a higher number of observations improves the convergence of the belief, hence why a certain number of works refer to bottom positions as the most advantageous [19]. On the other

hand, re-circulation increases the fluctuation of the instantaneous measurements which for average concentration models such as the Gaussian Plume leads to a higher discrepancy between the predicted and the observed data.

To summarise, the overall success and detection rate values were lower than those of the UAV platform experiments. This leads to the conclusion that the motion dynamics of the platform have a negative impact on the measurement process, reducing the quality of the observations. The upper middle position outperforms the remaining schemes with a slightly higher travelled distance than the top position. The results suggest that placing the sensors above the propellers leads to higher quality measurements due to a stable airflow from the intake region, while, below the propellers, the detection rate is higher due to air re-circulation. The negative point is a higher noise on the measured signal that can severely disrupt the detection of the target chemicals.

V. CONCLUSIONS

This work studied the impact of the UAV platforms and sensor positioning on OSL performance in natural conditions. The searching mission was guided by a cognitive strategy, and by a non-disturbing vertical pole carried by a human operator, which served as a reference. The experiments seem to indicate that small-size platforms are more vulnerable to external conditions decreasing the performance of the localisation, whereas an asymmetric structure poses a negative impact due to the non-uniform airflow. Overall, it was concluded that the sensor locations and the motion dynamics impact negatively in the search. The middle position, above the rotors, lead to the best results in general and the bottom scheme showed a high disturbance on the observed data. The position immediately below the propellers produced the worst results due to high level of turbulence. It was observed that the airflow interfered with the correct operation of the sensor due to temperature dissipation on the heater. For future work, sensors with temperature compensation are going to be tested under the propellers in order to evaluate if the quality of the measured signals can be improved. The goal is to take advantage of the higher detection rate observed in these regions, which may be crucial for the success of the search when the number of detections are scarce.

ACKNOWLEDGMENT

This work has been supported by PRR Project “Agenda Mobilizadora Sines Nexus” (ref. No. 7113), and by the Portuguese Foundation for Science and Technology (FCT) Ph.D. studentships SFRH/BD/149527/2019 and SFRH/BD/147988/2019, co-founded by the European Social Fund and by the State Budget of the Portuguese Ministry of Education and Science.

REFERENCES

[1] Lino Marques and Anibal T De Almeida. Electronic nose-based odour source localization. In *6th International Workshop on Advanced Motion Control. Proceedings (Cat. No. 00TH8494)*, pages 36–40. IEEE, 2000.

[2] Kok Seng Eu, Kian Meng Yap, and Tiam Hee Tee. An airflow analysis study of quadrotor based flying sniffer robot. In *Applied Mechanics and Materials*, volume 627, pages 246–250. Trans Tech Publ, 2014.

[3] Bing Luo, Qing-Hao Meng, Jia-Ying Wang, and Shu-Gen Ma. Simulate the aerodynamic olfactory effects of gas-sensitive uavs: A numerical model and its parallel implementation. *Advances in Engineering Software*, 102:123–133, 2016.

[4] Zhang-Qi Kang, Qing-Hao Meng, Bing Luo, Jia-Ying Wang, Xu-Yang Dai, and Shu-Gen Ma. Experimental verification of an aerodynamic olfactory effect model for the simulation of gas-sensitive rotorcrafts. In *2018 13th World Congress on Intelligent Control and Automation (WCICA)*, pages 1652–1657. IEEE, 2018.

[5] Sangwon Do, Myeongjae Lee, and Jong-Seon Kim. The effect of a flow field on chemical detection performance of quadrotor drone. *Sensors*, 20(11):3262, 2020.

[6] Shunsuke Shigaki, Muhamad Rausyan Fikri, and Daisuke Kurabayashi. Design and experimental evaluation of an odor sensing method for a pocket-sized quadcopter. *Sensors*, 18(11):3720, 2018.

[7] Bing Luo, Qing-Hao Meng, Jia-Ying Wang, and Ming Zeng. A flying odor compass to autonomously locate the gas source. *IEEE Transactions on Instrumentation and Measurement*, 67(1):137–149, 2017.

[8] Godall Rohi, Godswill Ofualagba, et al. Autonomous monitoring, analysis, and countering of air pollution using environmental drones. *Heliyon*, 6(1):e03252, 2020.

[9] Karena A McKinney, Daniel Wang, Jianhuai Ye, Jean-Baptiste de Fouchier, Patricia C Guimarães, Carla E Batista, Rodrigo AF Souza, Eliane G Alves, Dasa Gu, Alex B Guenther, et al. A sampler for atmospheric volatile organic compounds by copter unmanned aerial vehicles. *Atmospheric Measurement Techniques*, 12(6):3123–3135, 2019.

[10] Endrowednes Kuantama, Radu Tarca, Simona Dzitac, Ioan Dzitac, Tiberiu Vesselenyi, and Ioan Tarca. The design and experimental development of air scanning using a sniffer quadcopter. *Sensors*, 19(18):3849, 2019.

[11] Hayden A Hedworth, Tofigh Sayahi, Kerry E Kelly, and Tony Saad. The effectiveness of drones in measuring particulate matter. *Journal of Aerosol Science*, 152:105702, 2021.

[12] G Suchanek and R Filipek. Cfd analysis of a multi-rotor flying robot for air pollution inspection. In *Journal of Physics: Conference Series*, volume 2367, page 012010. IOP Publishing, 2022.

[13] Miguel Alvarado, Felipe Gonzalez, Peter Erskine, David Cliff, and Darlene Heuff. A methodology to monitor airborne pm10 dust particles using a small unmanned aerial vehicle. *Sensors*, 17(2):343, 2017.

[14] Abdul Samad, Diego Alvarez Florez, Ioannis Chourdakis, and Ulrich Vogt. Concept of using an unmanned aerial vehicle (uav) for 3d investigation of air quality in the atmosphere—example of measurements near a roadside. *Atmosphere*, 13(5):663, 2022.

[15] Hugo Magalhães, Rui Baptista, and Lino Marques. Evaluating cognitive odour source localisation strategies in natural water streams. In *5th Iberian Robotics conference*, pages 154–165. Springer, 2023.

[16] Michael Hutchinson, Cunjia Liu, Paul Thomas, and Wen-Hua Chen. Unmanned aerial vehicle-based hazardous materials response: Information-theoretic hazardous source search and reconstruction. *IEEE Robotics & Automation Magazine*, 27(3):108–119, 2019.

[17] Javier Burgués, Victor Hernández, Achim J Lilienthal, and Santiago Marco. Smelling nano aerial vehicle for gas source localization and mapping. *Sensors*, 19(3):478, 2019.

[18] Shunsuke Shigaki, Kei Okajima, Kazushi Sanada, and Daisuke Kurabayashi. Experimental analysis of the influence of olfactory property on chemical plume tracing performance. *IEEE Robotics and Automation Letters*, 4(3):2847–2853, 2019.

[19] Chiara Ercolani and Alcherio Martinoli. 3d odor source localization using a micro aerial vehicle: System design and performance evaluation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6194–6200. IEEE, 2020.

Finite-Time Standoff Target Tracking in The Presence of Wind

Pallov Anand[1], Pranav Niturkar[2], A. Pedro Aguiar[1], Rajat Agarwal[2], Manav Mishra[2], and P.B. Sujit[2]

[1]SYSTEC-ARISE and Department of Electrical and Computer Engineering
Universidade do Porto, Porto, Portugal
Email: {pallov, pedro.aguiar}@fe.up.pt
[2]Department of Electrical Engineering and Computer Science
Indian Insitute of Science Education and Research Bhopal, Bhopal, India
Email: {pranavn,arajat,manav20,sujit}@iiserb.ac.in

***Abstract*— In this paper, we propose a novel vector-field based guidance law for an unmanned aircraft (UA) to track a stationary and a moving target while maintaining a desired standoff distance under constant vehicle speed conditions in the absence and presence of wind. The guidance law achieves the convergence of the path of UA to the desired standoff distance with its heading angle to the desired heading angle in finite-time. We analyze the theoretical properties of the proposed guidance law using Lyapunov theory and evaluate the performance of the proposed guidance law through simulations and hardware experiments.**

***Index Terms*— finite-time, guidance vector field, target tracking, Lyapunov stability**

I. Introduction

Unmanned Aircraft (UA) have various advantages over manned aircraft namely safety, cost-effectiveness, precision, flexibility, accessibility etc. Among the various applications, safety is one of the most critical aspects as an UA can operate in dangerous and hazardous environments without risking human lives. This is mainly useful in military operations, search and rescue missions, target observation or tracking and monitoring natural disasters. Given the above, the investigation of target observation and tracking by UAs has been very active.

The work done in [1] presents a framework for UA to track a stationary as well as a moving target with and without wind by using Lyapunov guidance vector field approach. In [2], the authors have proposed a non-singular fast terminal sliding mode guidance law to achieve the objective of tracking a target by an UA. Tracking a target by multiple UA has an advantage in terms of accuracy which has been done in [3]. Several other works have been done in recent years to achieve the same objective [4], [5], [6], [7].

Realistically, in many situations, in addition to tracking, there is a necessity of doing the same in finite-time or under some bound which is known beforehand. Designing a finite-time controller is challenging as compared to designing an asymptotically stabilizing control because of the lack of effective analysis tools. Convergence rate also plays an important role in examining the performance of a controller. Some works in this direction can be found in [8], [9] where the objective is to increase the rate of convergence by increasing coupling strength, optimizing the system gain or designing a better communication topology. But, all these methods guarantee asymptotic convergence only. In practical applications, achieving finite-time convergence is desirable because the finite-time controllers lead to the improvement in the behavior of systems such as: high-speed, disturbance rejection and control accuracy. Finite-time target tracking is an important problem in the field of UA. This problem is challenging because of the uncertainty in the target's motion, environmental disturbances, and limited sensing capabilities of the UA. However, the ability to track a target in a finite time horizon has numerous applications, such as surveillance, search and rescue, and monitoring of critical infrastructure. One of the advantages of finite time target tracking by an UA is its ability to cover large areas quickly and efficiently. An UA can move faster and cover more ground than a human on foot or in a vehicle, which makes it an ideal choice for tracking a target over a wide area. Additionally, an UA can fly at different altitudes and angles, providing a unique perspective of the target and the surrounding environment. This perspective can help in identifying and tracking the target more accurately, even in challenging conditions such as low light or bad weather. Another advantage of target tracking by an UA is its potential to reduce the risk to human life. In situations such as search and rescue operations or monitoring of critical infrastructure, sending a human to track a target can be dangerous and even life-threatening. By using an UA for tracking, the risk to human life can be minimized, and the task can be completed more efficiently. Moreover, an UA can operate in areas that are difficult or impossible for humans to reach, such as mountainous terrain or hazardous environments.

Considering the advantages of an UA tracking a target mentioned above, the main contributions of this paper include:

This work was supported in part by project RELIABLE (PTDC/EEI-AUT/3522/2020), the ARISE Associated Laboratory (LA/P/0112/2020) and RD Unit SYSTEC-Base (UIDB/00147/2020) and Programmatic (UIDP/00147/2020) funded by national funds through the FCT/MCTES (PIDDAC). It is also partially funded by SERB CRG Grant-CRG/2021/007916. The first author was supported by a Ph.D. Scholarship, grant 2022.11199.BD from Fundação para a Ciência e a Tecnologia (FCT), Portugal.

979-8-3503-0704-7/23/$31.00 ©2023 IEEE

- The development of a vector field based guidance law for an UA to track a stationary and a moving target in the presence of wind achieving convergence in finite-time.
- The convergence analysis of the proposed vector field guidance law using Lyapunov based tools.
- The illustration of the behavior and efficacy of the developed guidance law through various simulation results as well as with some hardware experiments.

The remainder of the paper is outlined as follows: Section II presents the basic concepts about finite-time criteria. Section III is devoted to the problem formulation. Section IV elaborates the main results for UA to achieve target tracking in finite-time for various scenarios. Simulation results and hardware implementation are discussed in section V which verifies the efficacy of the proposed methodology while section VI concludes this paper along with the scope of work to be in done in future.

II. Basic Preliminaries

Let $\Xi \subseteq \mathbb{R}^n$ be an open connected set which includes the origin. Consider a nonlinear system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{1}$$

where $\boldsymbol{x}$ is a state vector signal of n-dimension and $\boldsymbol{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a vector function with $\boldsymbol{f}(0) = 0$ which is assumed to be sufficiently smooth. The origin is Lyapunov stable if for any $\boldsymbol{x}(t_0) \in \Xi$, the solution $\boldsymbol{x}(t; x_0)$ exists $\forall\, t \geq 0$, and for any $\epsilon > 0$ there is a $\delta > 0$ such that for any $\boldsymbol{x}_0 \in \Xi$, if $||\boldsymbol{x}_0|| \leq \delta$ then $||\boldsymbol{x}(t, x_0)|| \leq \epsilon\ \forall\ t \geq 0$. Further, in addition to the origin being Lyapunov stable, if the trajectory converges in finite-time i.e. for any $\boldsymbol{x}_0 \in \Xi$ there exists $0 \leq T < +\infty$ such that $\boldsymbol{x}(t, x_0) = 0\ \forall\ t \geq T$ then the system's origin is finite-time stable in a pre-determined time and the function

$$T(x_0) = \inf\{T \geq 0 : \boldsymbol{x}(t, \boldsymbol{x}_0) = 0\ \forall\ t \geq T\}$$

is known as settling-time function for (1).

Proposition 1: A positive definite function $V(t)$, which satisfies

$$\dot{V}(t) \leq -cV^{\alpha}(t),\ \forall\ t \geq t_0;\ V(t_0) \geq 0 \tag{2}$$

for any given time $t_0 \geq 0$, where c is a positive constant and $0 < \alpha < 1$. Then there exists $T \geq t_0$ such that for $t_0 \leq T, V(t)$ will satisfy

$$\begin{aligned} V^{1-\alpha}(t) &\leq V^{1-\alpha}(t_0) - c(1-\alpha)(t - t_0) \\ V(t) &\equiv 0,\ \forall\ t \geq T \text{ with } T = t_0 + \frac{V^{1-\alpha}(t_0)}{c(1-\alpha)} \end{aligned} \tag{3}$$

Refer to [10] for the proof.

III. Problem Formulation

We consider an UA that is equipped with a low-level flight control system that provides roll, pitch and yaw stability of the aircraft as well as velocity tracking and altitude-hold functions. For aircraft guidance, the flight control system accepts explicit speed, and turn commands. Using this command structure, the system model presented to the guidance layer is a kinematic model given as follows:

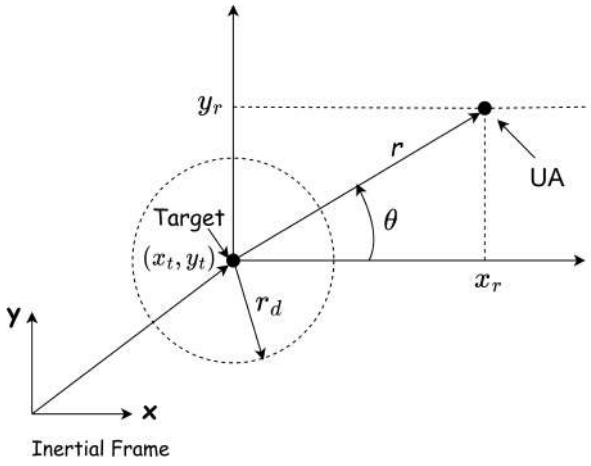


Fig. 1: Tracking geometry with the ground target of interest at the origin of the target frame.

$$\begin{aligned} \dot{x} &= u_1 \cos\psi \\ \dot{y} &= u_1 \sin\psi \\ \dot{\psi} &= u_2 \end{aligned} \tag{4}$$

where $[x, y]^T \in \mathcal{R}^2$ is the inertial position of the aircraft and ψ is the aircraft heading angle. The control signals are u_1 and u_2 which denotes the commanded air speed (m/s) and the turning rate of the aircraft (rad/s) respectively. For the above kinematic model, the relative position between UA and a given target located at $[x_t, y_t]^T$ is given by $x_r = x - x_t$, $y_r = y - y_t$. We assume the position of the target is continuously differentiable in time. For convenience, we also define here the radial distance as

$$r = \sqrt{x_r^2 + y_r^2} \tag{5}$$

Let the steady wind components which is assumed to be constant be given by $[w_x, w_y]^T (m/s)$. Then, the modified kinematic model w.r.t. the target and the wind takes the form

$$\begin{aligned} \dot{x}_r &= u_1 \cos\psi + w_x - \dot{x}_t \\ \dot{y}_r &= u_1 \sin\psi + w_y - \dot{y}_t \\ \dot{\psi} &= u_2 \end{aligned} \tag{6}$$

where $[\dot{x}_t, \dot{y}_t]^T$ denotes the velocity of the target. For simplification purpose, let us take $[T_x, T_y]^T = [w_x - \dot{x}_t, w_y - \dot{y}_t]$. The main guidance objectives are as follows:

- To make a UA loiter around a stationary target at a constant stand-off radius (r_d), with and without wind disturbance.
- Loitering of a UA around a moving target with a constant velocity in the presence and absence of disturbance.
- To drive the error between the relative position of UA to the stand-off radius to zero in finite-time while tracking a stationary and a moving target.

IV. Finite-time Target Tracking

In this section, we present the results for the tracking of a stationary and a moving target by an UA. For the kinematic model defined in (6) and taking (5) into account, we take $\dot{x}_d$ and $\dot{y}_d$ as the desired relative velocity of the UA. Thus, we present a guidance vector field as given below:
For $r_\alpha \leq r < r_d$*:

$$\begin{bmatrix}\dot{x}_d \\ \dot{y}_d\end{bmatrix} = \frac{\lambda v_0}{r^2}\begin{bmatrix}P_1 + R_1 \\ P_2 - R_2\end{bmatrix} \tag{7}$$

For $r \geq r_d$:

$$\begin{bmatrix}\dot{x}_d \\ \dot{y}_d\end{bmatrix} = -\frac{\lambda v_0}{r^2}\begin{bmatrix}Q_1 - S_1 \\ Q_2 + S_2\end{bmatrix} \tag{8}$$

where

$$P_1 = x_r(r_d^2 - r^2)^{(2\alpha-1)}, R_1 = y_r\sqrt{r^2 - (r_d^2 - r^2)^{2(2\alpha-1)}}$$
$$P_2 = y_r(r_d^2 - r^2)^{(2\alpha-1)}, R_2 = x_r\sqrt{r^2 - (r_d^2 - r^2)^{2(2\alpha-1)}}$$
$$Q_1 = x_r(r^2 - r_d^2)^{(2\alpha-1)}, S_1 = y_r\sqrt{r^2 - (r^2 - r_d^2)^{2(2\alpha-1)}}$$
$$Q_2 = y_r(r^2 - r_d^2)^{(2\alpha-1)}, S_2 = x_r\sqrt{r^2 - (r^2 - r_d^2)^{2(2\alpha-1)}}$$

$\alpha \in (0, 1)$, r_α denotes the boundary below which (7) is not defined and λ is known as scaling factor which is evolved in time according to the following expression

$$\lambda^2(\dot{x}_r^2 + \dot{y}_r^2) - 2\lambda(\dot{x}_r T_x + \dot{y}_r T_y) + (T_x^2 + T_y^2 - v_0^2) = 0 \tag{9}$$

where v_0 is a given nominal speed for the UA. The desired heading angle of the UA is given by

$$\psi_d = \tan^{-1}\left(\frac{\lambda\dot{y}_d - T_y}{\lambda\dot{x}_d - T_x}\right) \tag{10}$$

Remark 1: For the case of stationary target with no wind, the scaling factor takes the constant value unity and the desired heading angle is given by

$$\psi_d = \tan^{-1}\left(\frac{\dot{y}_d}{\dot{x}_d}\right) \tag{11}$$

The result for an UA to loiter around a target at a stand-off radius r_d, is given below.

Theorem 1: Let $\rho_d(r_d) = \{(x, y) \in \mathcal{R}^2 : (x - x_t)^2 + (y - y_t)^2 = r_d^2\}$ be a given desired circular path of radius r_d centered at the target, and $v_0 > 0$ a given nominal speed for the UA. Then, the position of the UA converges to $\rho_d(r_d)$ in finite-time by selecting the control signals

$$u_1 = \lambda v_0 \tag{12}$$
$$u_2 = -sgn(e)|e|^\eta + \dot{\psi}_d \tag{13}$$

where $\eta \in (-1, 1)$, $e = \psi - \psi_d$ denotes the error between the actual and the desired heading angle.

*For the expression given in (7), $r < r_\alpha$ is not in the domain of the function defined. In the simulations we choose the optimum value $\alpha = 3/4$ for which $r_\alpha = r_d/\sqrt{2}$. However, any value in the range $2/3 \leq \alpha \leq 3/4$ can be preferred.

Proof: Consider a Lyapunov function $V(x, y) = (r^2 - r_d^2)^2$, and suppose that $(\dot{x}_r, \dot{y}_r) = (\dot{x}_d, \dot{y}_d)$. Then, it follows that

$$\dot{V} = 4(r^2 - r_d^2)(x_r\dot{x}_d + y_r\dot{y}_d) \tag{14}$$

Then, in case of stationary target with no wind i.e., $[w_x, w_y]^T = [0, 0]^T$ and $[\dot{x}_t, \dot{y}_t]^T = [0, 0]^T$, it follows that for $r_\alpha \leq r < r_d$ and $r \geq r_d$, putting Eq. (7) and (8) respectively in (14) we obtain

$$\begin{aligned}\dot{V} &= -4\lambda v_0(r^2 - r_d^2)^{2\alpha} \\ &= -4\lambda v_0 V^\alpha\end{aligned} \tag{15}$$

Thus, Eq. (15) satisfies the finite-time criteria given by Eq. (2) in proposition 1, where $c = 4\lambda v_0$. To end this part of the proof, we need to show that the proposed desired guidance vector field can be generated through (12)-(13). To this end, we set

$$u_1 = \sqrt{\dot{x}_d^2 + \dot{y}_d^2} = \lambda v_0 \tag{16}$$

To show the convergence of actual heading angle to the desired heading angle ψ_d we take the error $e = \psi - \psi_d$ and the Lyapunov function $V(e) = (1/2)e^2$ whose time derivative is

$$\dot{V} = e(u_2 - \dot{\psi}_d) \tag{17}$$

Putting Eq. (13) in Eq. (17), we get:

$$\begin{aligned}\dot{V} &= e\left(-\frac{e}{|e|}|e|^\eta\right) = -e^2|\pm\sqrt{e^2}|^{(\eta-1)} \\ &= -2V|\sqrt{2V}|^{\eta-1} = -2^{\left(\frac{\eta+1}{2}\right)}V^{\left(\frac{\eta+1}{2}\right)} = -cV^\alpha\end{aligned} \tag{18}$$

where, $\alpha = \left(\frac{\eta+1}{2}\right)$, $c = 2^\alpha$, and $-1 < \eta < 1$. Since, Eq. (18) satisfies finite-time criteria given by (2), we can conclude that ψ converges to ψ_d in finite-time. Now, let us consider the case of moving target in the presence of wind. We define a polar coordinate system (r, θ), centered at the inertial target position $[x_t, y_t]^T$, with $r = \sqrt{x_r^2 + y_r^2}$ and $\theta = \tan^{-1}\left(\frac{y_r}{x_r}\right)$. Thus, the guidance vector fields defined in (7) and (8) can be expressed respectively in polar form as

$$\begin{bmatrix}\dot{r} \\ r\dot{\theta}\end{bmatrix} = \frac{-\lambda v_0}{r}\begin{bmatrix}-P_1/x_r \\ R_1/y_r\end{bmatrix} + \begin{bmatrix}B_1 \\ B_2\end{bmatrix} \tag{19}$$

$$\begin{bmatrix}\dot{r} \\ r\dot{\theta}\end{bmatrix} = \frac{-\lambda v_0}{r}\begin{bmatrix}Q_1/x_r \\ S_1/y_r\end{bmatrix} + \begin{bmatrix}B_1 \\ B_2\end{bmatrix} \tag{20}$$

where $B_1 = T_x\cos\theta + T_y\sin\theta$ and $B_2 = T_y\cos\theta - T_x\sin\theta$. From (19) and (20), it follows that at $r = r_d$, $\dot{r} = T_x\cos\theta + T_y\sin\theta$. Note that apart from the stationary target with no wind case, the vector field will always have a non-zero radial component. Thus, if we use the vector fields defined in (7) and (8) in (6), it does not give a constant commanded speed of v_0. Therefore, in order to maintain the UA at the desired stand-off radius (r_d), we make use of the scaling factor λ and set the desired velocity u_1 of the UA such that

$$\begin{bmatrix}u_1\cos\psi \\ u_1\sin\psi\end{bmatrix} = \begin{bmatrix}\lambda\dot{x}_d - T_x \\ \lambda\dot{y}_d + T_y\end{bmatrix} \tag{21}$$

The updated value of λ is obtained by (9) which is used to calculate the desired heading angle given by (10). Thus, utilizing the updated scaling factor λ and using the control function given in (13), the path of UA converges to desired stand-off radius in finite-time for the case of moving target in presence of wind. Utilizing the similar approach, results for the path convergence of UA to the desired stand-off radius in case of stationary target with wind and moving target without wind can be obtained. ■

V. Simulation and Experimental Results

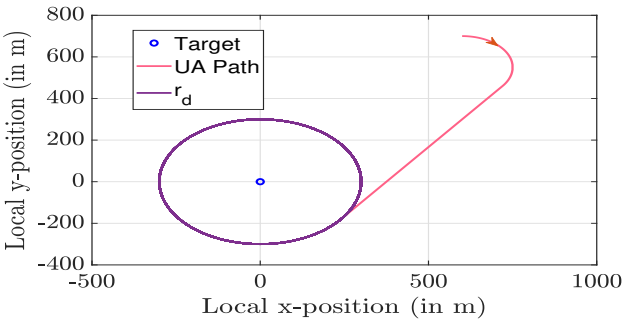
We evaluate the proposed vector field based guidance through simulations and hardware experiments. Initially, we will describe the simulation setup and the simulation results followed by hardware setup and results.

A. Simulation setup

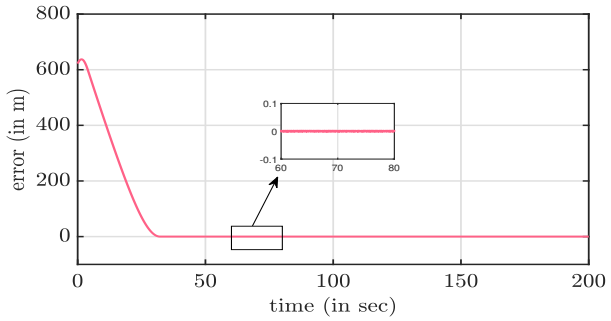
We consider the following simulation setup: initial position of UA: $[600, 700]^T$, initial heading angle = 0 rad/s, UA velocity = 30 m/s, Scaling factor, $\lambda = 1$, $\alpha = \eta = 3/4$ and stand-off radius $(r_d) = 300$ m, controller turning rate gain: 10, $\omega_{max} = 0.2$ rad/s , wind component: $[w_x, w_y]^T = [2, 3]^T m/s$, velocity of the target: $[0, 10]^T m/s$.

B. Simulation results

Figure 2a shows the convergence of UA path to the desired stand-off radius of 300 m while loitering around a stationary target in absence of wind while fig. 2b represents the corresponding error between the UA position to the desired stand-off distance, which is converging to zero in finite-time. Figure 3a represents the trajectory of an UA loitering around a stationary target with wind while fig. 3b represents the corresponding error between the UA position to the desired stand-off distance, which is converging to zero in finite-time. Figure 4a shows the efficiency of the derived finite-time steering controller when there is a zero mean Gaussian noise with standard deviation $\sigma = 0.01$ along with a constant wind for the case of tracking a stationary target while fig. 4b represents the error between the UA path and the desired stand-off radius for the same case. Figure 5a shows the trajectories of moving target while an UA loiters around it at a desired standoff distance in absence of wind. Figure 5b represents the corresponding error between the UA position to the desired stand-off distance, which is converging to zero in finite-time. Figure 6a shows the trajectories of moving target and an UA loiters around it at a desired standoff distance in presence of wind while fig. 6b represents the corresponding error between the UA position to the desired stand-off distance, which is converging to zero in finite-time in presence of wind. Figure 7a represents the tracking of a moving target by an UA in present of disturbances i.e., wind with a constant velocity and a zero mean Gaussian noise with $\sigma = 0.01$ whereas fig. 7b is the corresponding error between the UA path and the moving target for the same case.

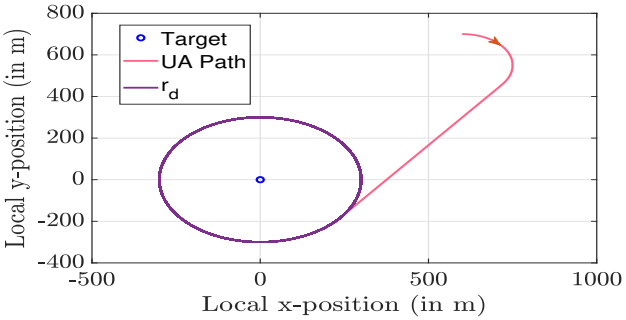


(a) Trajectory of an UA tracking a stationary target while maintaining a standoff distance with the target in absence of wind.

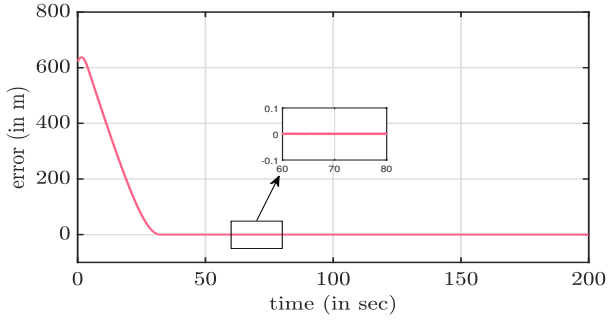


(b) Error between the position of UA and the desired target stand-off radius of 300m.

Fig. 2: An UA tracking a stationary target in absence of wind.

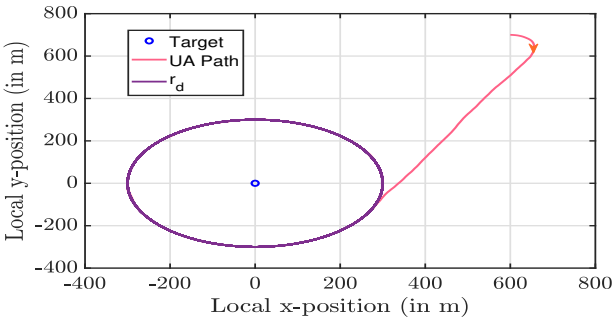


(a) Trajectory of an UA tracking a stationary target while maintaining a standoff distance with the target in presence of wind.

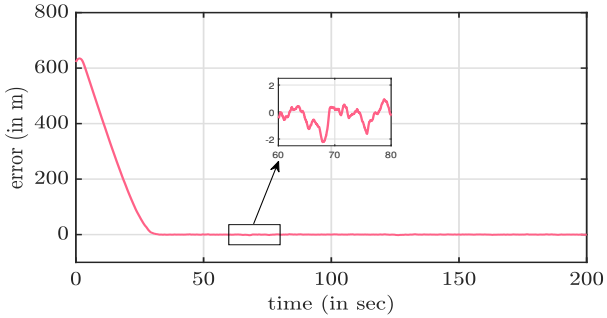


(b) Error between the position of UA and the desired target stand-off radius of 300m with wind.

Fig. 3: An UA tracking a stationary target in presence of wind.

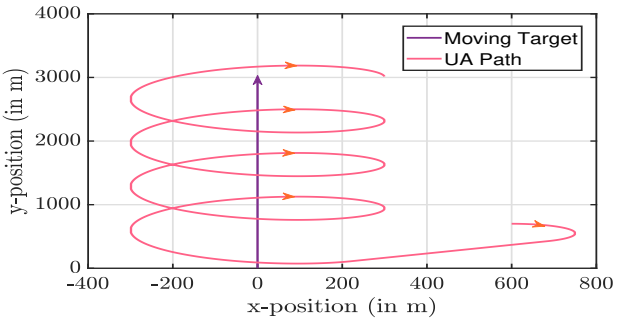


(a) Trajectory of an UA tracking a stationary target while maintaining a standoff distance with the target in presence of wind and a Gaussian noise.

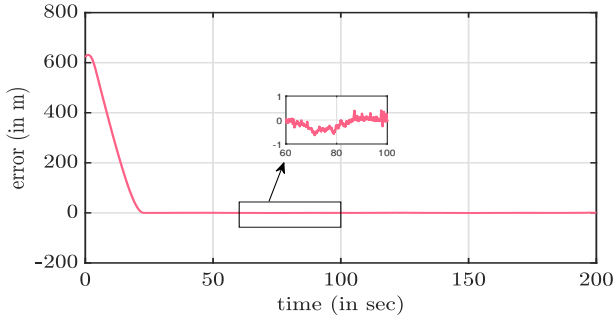


(b) Error between the position of UA and the desired target standoff radius of 300m with wind.

Fig. 4: An UA tracking a stationary target in presence of disturbances (wind and Gaussian noise).

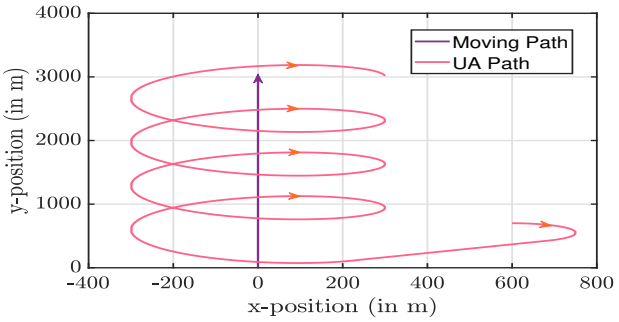


(a) Trajectory of an UA tracking a moving target in absence of wind and maintaining a standoff distance from it.

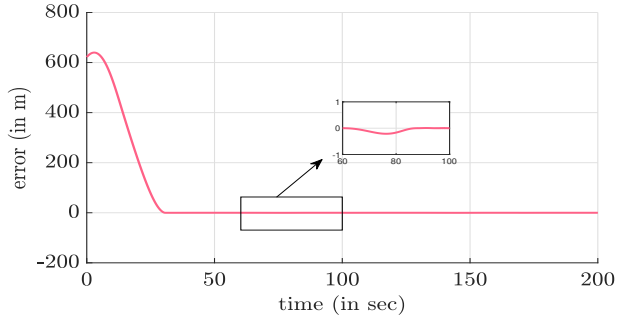


(b) Error between the position of UA and the desired target standoff radius of 300m without wind.

Fig. 5: An UA tracking a moving target in absence of wind.

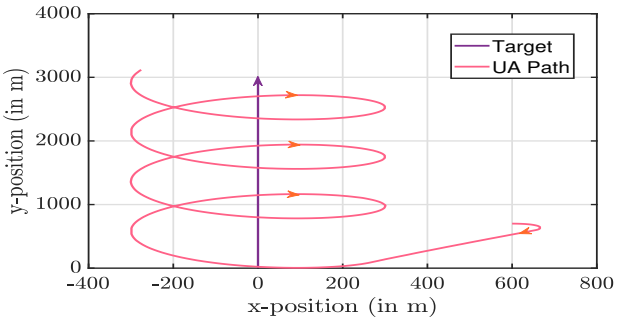


(a) Position of moving target and UA in presence of wind.

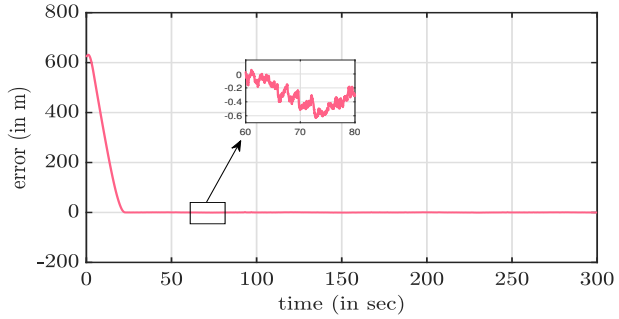


(b) Error between the position of UA and the moving target to the desired stand-off radius of 300m with wind.

Fig. 6: An UA tracking a moving target in presence of wind.



(a) Trajectory of an UA tracking a moving target while maintaining a standoff distance with the target in presence of wind and a Gaussian noise.



(b) Error between the position of UA and the desired target standoff radius of 300m with wind.

Fig. 7: An UA tracking a moving target in presence of disturbances (wind and Gaussian noise).

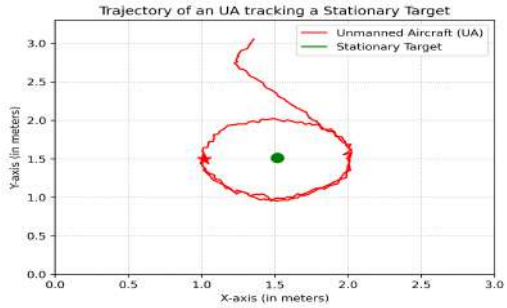


Fig. 8: Experimental plot of trajectory of an UA tracking a stationary target.

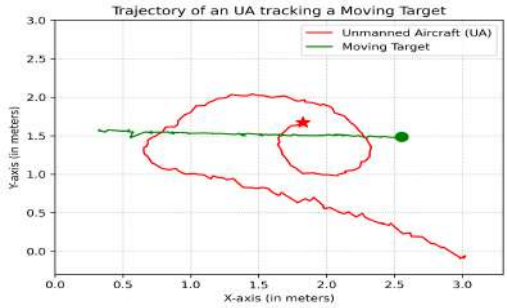


Fig. 9: Experimental plot of trajectory of an UA tracking a moving target.

C. Hardware Implementation

In this experiment, we employ a real-time hardware configuration to evaluate the practical implementation of stationary and a moving target tracking by an UA with a finite-time Lyapunov guidance field. We used BitCraze Crazyflie 2.1 nano-copter as UA and Dagu Wild Thumper 4WD as the target vehicle to achieve our objective. We employed the Lighthouse System [11], which tracked the location of the CrazyFlie, equipped with Lighthouse positioning decks, in a $3.5m \times 3.0m \times 2.0m$ arena using four SteamVR Base stations. In addition, we mounted a Lighthouse positioning deck-equipped Crazyflie drone on the Dagu rover to enable the real-time monitoring of its position and movement. The companion computer, running on Ubuntu 20.04 with Nvidia RTX 3060 GPU, executed our algorithm and processed the live locations of the UA and the target through CrazyFlie as input from the CrazySwarm ROS package's appropriate rostopics to compute the real-time next iterative waypoints of the UAs [12] [13]. Subsequently, the companion computer sent the calculated actions to the CrazyFlie via the CrazySwarm ROS package. The experimental videos are given in [14] along with the error and convergence results.

D. Hardware results

Using the above hardware setup, a single vehicle target stand-off experiment was carried out for two cases i.e., stationary and a moving target. The target speed was 2cm/s, and the UA speed was 5cm/s. Figure 8 shows the trajectory of the stationary target and the single UA following the target while maintaining the desired stand-off radius of 0.5m while fig. 9 shows the same for the moving target (desired stand-off radius = 0.5m). From the figure, we can see that the vehicle tracks the target.

VI. Conclusion and Future Works

In this paper, the problem of tracking a stationary and a moving target by an UA in the absence and presence of disturbance is investigated. The expression for the vector guidance field is proposed in such a way that an UA is guided to a stand-off radius in finite-time from the target and loiter around it in absence and presence of disturbance. Convergence analysis using Lyapunov theory is carried out. Moreover, the control function is derived for the heading angle which takes the UA to the desired heading angle in finite-time.

The proposed guidance law can be further generalized for any number of vehicle target stand-off distance. Effect of wind is considered but the effect of communication delay needs to be analyzed. Further extensions can be done for (i) tracking a group of targets with varying desired radius (ii) integrating target velocity estimation and stand-off distance (iii) Coordinated standoff tracking by multiple UAs.

References

[1] E. W. Frew, D. A. Lawrence, and S. Morris, "Coordinated standoff tracking of moving targets using lyapunov guidance vector fields," *Journal of guidance, control, and dynamics*, vol. 31, no. 2, pp. 290–306, 2008.

[2] K. Wu, Z. Cai, J. Zhao, and Y. Wang, "Target tracking based on a nonsingular fast terminal sliding mode guidance law by fixed-wing uav," *Applied sciences*, vol. 7, no. 4, p. 333, 2017.

[3] S. Huang, Y. Lyu, J. Shi, Q. Zhu, L. Su, and C. Lin, "Coordinated standoff tracking of a moving target based on the lateral offset," in *Advances in Guidance, Navigation and Control: Proceedings of 2022 International Conference on Guidance, Navigation and Control*. Springer, 2023, pp. 680–688.

[4] T. Harinarayana, S. Hota, and R. Kushwaha, "Vector field guidance for standoff target tracking," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 236, no. 14, pp. 2963–2973, 2022.

[5] C. Lin, J. Shi, W. Zhang, and Y. Lyu, "Standoff tracking of a ground target based on coordinated turning guidance law," *ISA transactions*, vol. 119, pp. 118–134, 2022.

[6] M. Kim, "Error dynamics-based guidance law for target observation using multiple uavs with phase angle constraints via evolutionary algorithms," *Journal of Control, Automation and Electrical Systems*, vol. 32, no. 6, pp. 1510–1520, 2021.

[7] S. Bhagat and P. Sujit, "Uav target tracking in urban environments using deep reinforcement learning," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020, pp. 694–701.

[8] Z. Li, Z. Duan, and G. Chen, "On h∞ and h2 performance regions of multi-agent systems," *Automatica*, vol. 47, no. 4, pp. 797–803, 2011.

[9] Y. Zhao, Z. Duan, and G. Wen, "Distributed finite-time tracking of multiple euler–lagrange systems without velocity measurements," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 11, pp. 1688–1703, 2015.

[10] Y. Tang, "Terminal sliding mode control for rigid robots," *Automatica*, vol. 34, no. 1, pp. 51–56, 1998.

[11] A. BITCRAZE, "Lighthouse positioning system," *Bitcraze. io*, 2021.

[12] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3299–3304.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[14] M. Lab, https://moonlab.iiserb.ac.in/research_page/loiter.html.

Late-Fusion Multimodal Human Detection based on RGB and Thermal Images for Robotic Perception

Elísio Sousa[1*], Kennedy O. S. Mota[1], Iago P. Gomes[2], Luís Garrote[1], Denis F. Wolf [2], and Cristiano Premebida[1]

***Abstract*— This paper addresses the problem of detecting humans in RGB and Thermal (long-wave IR) images taken by cameras mounted onboard a mobile robot. Human/Pedestrian detection is currently one of the most pertinent object detection problems, mainly due to safety concerns in autonomous vehicles. The majority of approaches apply deep-learning techniques based solely on RGB images. However, they have a few shortcomings, namely that during foggy weather, nighttime, and low-light scenarios, these images may not contain sufficient information. To address these issues, this work studies the use of thermal cameras as a complementary source of information for human detection in indoor and outdoor environments. The proposed approach uses YOLOv5 to detect pedestrians in both thermal and RGB images. Moreover, the different modalities are combined using early and late fusion techniques. Evaluation of the proposed approach is carried out in the FLIR Aligned dataset and in a new in-house dataset. Results indicate that the use of fusion techniques highlights a promising way to improve the overall performance in this application domain.**

I. Introduction

The importance of human detection has been established in the scientific community for a long time now. From indoor patrolling to autonomous driving, the applications of detecting humans, pedestrians, and intruders are many [1]–[5]. Most of the work done in this area has been using either RGB cameras, LIDAR sensors, or a combination of both [6]–[10]. However, there are scenarios where both RGB images and LiDAR point clouds show their limitations, such as when dealing with varying lighting conditions or shadows in RGB images, or when the subjects of interest are too far away and the LiDAR-points that represent the human are too scattered. As part of multi-modal approaches (see Fig. 1), the use of Thermal cameras can overcome some limitations found in RGB cameras and LiDARs.

Thermal sensors represent the environment solely by detecting the thermal (long wave infrared) energy emissions. Unlike RGB cameras, thermal sensors are more invariant to lighting conditions and robust to a wide range of light variations and weather conditions [11]. As such, Thermal cameras can be used in poor lighting or weather conditions in which regular RGB cameras may produce poor results, such as rain and fog. RGB cameras can, in particular, also show particularly poor performance when the scenarios of interest include partial or total darkness. In this case, the use of night vision cameras is arguable, as thermal images present an advantage against occlusion: to support such an argument, the set of RGB and Thermal image pairs taken from the FLIR-Aligned [12] dataset and our in-house dataset, shown in Fig. 2, highlight some of these limitations.

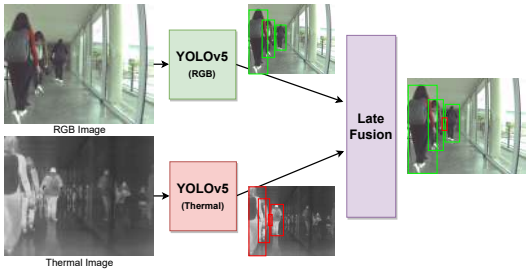


Fig. 1. Proposed illustration of a pipeline encompassing two YOLOv5 models and a late detection fusion stage.

Additionally, an interesting aspect of thermal cameras that can also be highlighted is that they do not represent people's faces in as much detail as RGB cameras do. This characteristic can also be important in human-populated and/or outdoor scenarios since they conserve much more of the pedestrian's privacy. However, thermal images do not come without their downsides. For example, clear glass can filter out the infrared radiation that thermal sensors detect, whereas it will not filter radiation in the visible spectrum (i.e., RGB).

Multi-modal approaches, on the other hand, present an important way to combine key characteristics/attributes of both RGB and Thermal cameras to improve the overall performance in human detection tasks. The work proposed hereafter (see Fig. 1) aims to use thermal images as a source of information for human/pedestrian detection, by using YOLOv5. The obtained model will be combined with a YOLOv5 model, trained with RGB images, in a multimodal configuration using a late fusion stage (illustrated in Fig. 1). Therefore, each modality can complement the shortcomings of the other one. Moreover, an early fusion strategy will be studied as well.

The contributions of this study are related to multi-modality combinations employing early and late fusion strategies, both having state-of-the-art deep-detectors as a baseline. More specifically, the contributions are the following:

[1] Authors are with the Institute of Systems and Robotics, Department of Electrical and Computer Engineering, University of Coimbra, Portugal.
[2] Authors are with the Institute of Mathematics and Computer Science, University of São Paulo, Brazil
* Corresponding Author: elisio.alex.sousa@gmail.com

Fig. 2. Selection of scenarios from the FLIR-Aligned dataset [12] (outdoor) and the in-house dataset (indoor) containing RGB and Thermal image pairs, where some limitations of RGB cameras are highlighted. The scenarios contain occlusions, lens flare artifacts, and glow artifacts.

• Early fusion: RGB and Thermal images are integrated in a state-of-the-art deep-detector (YOLOv5). The data loaders and the first layer of the YOLOv5 model were modified to train and infer on an RGB-Thermal image.

•Late fusion: For each image of the validation set, a list of all the detections is compiled, from both modalities, and a rescoring strategy is applied. This list already removes duplicates *i.e.*, if both modalities contain bounding boxes that match the same ground truth, only the thermal detection is kept.

• Dataset: a new "in-house" dataset for indoor environments has been built on RGB and Thermal data, using cameras mounted onboard a mobile robot, encompassing representative conditions found in the real world.

In terms of structure, Section II provides a brief review of the use of thermal images for pedestrian detection and related works. In Section III the proposed approaches and respective methodologies are presented. Section IV discusses the results of experiments carried out in the FLIR-Aligned dataset (outdoor) and in an in-house dataset (indoor). Finally, Section V concludes the paper with remarks and suggests future directions.

II. Related work: a brief overview

Investigation into studies related to pedestrian or human detection in thermal images is a long-standing topic, with papers dating as far back as 2004 [13]. Back then, *i.e.*, before the deep-learning wave, the main strategies were based on background subtraction and/or hand-engineered descriptors (*e.g.*, HOG) to obtain the contours of pedestrians, aided by the fact that these kinds of thermal images offer better contrast between pedestrians and the background [14], [15].

The lack of thermal datasets led some researchers to try to use RGB detectors for the thermal domain, again trusting that the contrast offered by thermal images would increase the detection rate [16]. Furthermore, due to the small number of publicly available datasets a few years ago, some researchers attempted synthesizing artificial thermal images [17], [18] or using thermal information to enhance the RGB input [19].

Meanwhile, as a meritorious attempt to fill in this gap, more and more researchers collected datasets containing RGB and Thermal images captured at the same time. Datasets such as in [20], [21] are good examples of those that have been made publicly available. In the same direction, KAIST [22] and FLIR [12] are some of the meaningful datasets that have been used for benchmarking state-of-the-art detectors. Nevertheless, notice that both of them encompass outdoor scenarios *i.e.*, more suitable for autonomous vehicles. Table I shows other datasets used in the literature. The release of additional datasets coincided with the advance of deep learning methods, steering current research into using deep learning detectors such as YOLO [23].

Additionally, there has been an increasing number of works centered around investigating sensor/data fusion techniques, so that RGB and thermal images can complement each other, rather than being treated as separate use cases [24]–[26]. These techniques are classified into early, middle, and late fusion. Early-fusion techniques include approaches that combine the different modalities in the model input domain (*i.e.*, raw data), before feeding them to a model. In the middle-fusion techniques, specific layers extract features of each modality, which are combined by different operations between the layers of the model. Finally, late-fusion methods are responsible for combining outputs from different models, specialized in each modality.

For human detection, the most common modalities are RGB, depth, and multispectral (e.g., thermal) images, and point cloud. Pei et al. [24] combined RGB with infrared (IR) images using two-branches deep convolutional neural networks, that fuse the features of each modality extracted by specialized convolutional layers. They explored three fusion operations (*i.e.*, sum, concatenation, and max), where the sum of feature vectors achieved the best results. Other primary studies employed different operations, such as attention-mechanism [27]–[30], residual connections [25], [26], and feature pyramid [31].

For early fusion methods, techniques range from concatenating image channels from different modalities [32], using one modality to define regions of interest (ROI), or using channels from one modality to enhance features of another [33]. The disadvantage of early and middle fusion lies, respectively, in the ability to relate significant cross-modality features; and, high computational and memory cost, and limitations for real-time applications (i.e., especially as it requires more layers and complexity of networks).

In this sense, late-fusion approaches aggregate the output of different models to enhance the reliability of detections. The aggregate operation can be performed by probabilistic frameworks [34], rule-based [32], or data-driven methods [35].

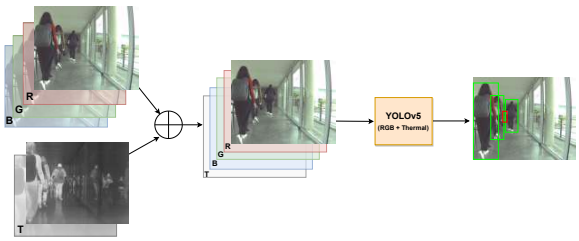


Fig. 3. Early-fusion Model with Channel Concatenation for Human Detection.

This paper explores data-driven late-fusion for rescoring detections using RGB and thermal images. The rescoring process combines the detections of each modality and estimates new confidence values based on spatial and contextual features. The advantages of this approach are the lightness of the fusion network, which allows its application in real-time, and the versatility in combining different modalities. The experimental results showed the feasibility of the proposal with performance gains in relation to the baselines evaluated in the FLIR dataset (outdoor) and an in-house dataset (indoor).

III. Data-driven Late Fusion Human Detection using RGB and Thermal Images

This section describes the data-driven human detection deep learning approach that combines RGB and thermal modalities and reevaluates their confidence scores (*i.e.*, rescoring). Moreover, it also presents the early-fusion approach used for comparison in Section IV.

A. Early-Fusion

In contrast with late fusion, which involves combining the detections of separate RGB and Thermal image models, early fusion, in this context, involves combining RGB and Thermal images into a single input image, which is then fed into the detector. Figure 3 shows a flowchart that illustrates the model with channel concatenation. The advantage of early fusion lies in the simplicity of modeling both RGB and Thermal modalities, which can lead to a better performance of the detector at the expense of a more computationally expensive architecture. However, this approach assumes that the cross-modality interactions are implicitly learned during the model training.

In this work, the YOLOv5 [42] model was modified to accept as input a 4-channel image corresponding to the 3 channels of the RGB image and 1 channel of the Thermal image. To guarantee the correctness of the models, the data loader was updated to import 4 channel images from the datasets, and data augmentation strategies were modified to propagate affine transformations to the thermal channel and to avoid equalization/brightness/contrast transformations from the RGB image to be applied to the thermal channel.

B. Late-Fusion

The late-fusion approach combines the detections performed in each modality. Therefore, a modality-specific YOLOv5 estimates bounding boxes with scores for each modality. These outputs are combined on a third network, which also estimates new scores based on the fusion data. Figure 1 illustrates this process.

The data-driven rescoring approach is based on Asvadi et al. [35], that proposed a multimodal vehicle detection system with real-time capabilities, that combined RGB images with dense depth and reflectance images estimated by a LiDAR sensor. Moreover, they employed a Multi-Layer Perceptron (MLP) for estimating new scores based on the detections and scores of each modality. The input of the rescoring module is a feature vector with spatial features and detection scores of bounding boxes that were matched based on their overlap. However, in this paper, besides the spatial features, we used contextual features to improve the rescoring model.

1) Rescoring Neural Network Training Method:

As previously stated, one of the late fusion techniques applied in this work consists of a rescoring method, which relies on a Multi-Layer Perceptron (MLP). For the training stage of this model, we used Mean Square Error as the loss function

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(p-t)^2,$$

where N is the number of detections, p is the rescored confidence and t is the target confidence, defined by the Intersection over Union (IoU) between the detected bounding boxes and the ground-truth.

For each detection, the following feature vector is built:

• **Spatial features**: these features refer to information on the positions and dimensions of the bounding boxes.

$$S = (h, w, x_c, y_c, a, c_{IoU}) \tag{1}$$

where (h, w) are the height and width of the bounding box, (x_c, y_c) are the geometrical center, a is the area, and c_{IoU} is the Intersection over Union (IoU) between the thermal and RGB detections (0 if the detection does not have a match).

• **Confidence features**: these features represent the confidence that modality-specific models associate with their detections.

$$P = (s, t_{bs}, rgb_{bs}) \tag{2}$$

where s is the confidence score of the detection, and, (t_{bs}, rgb_{bs}) are the Brier scores of the thermal and RGB detection, respectively.

• **Contextual features**: these features correspond to the semantic and contextual information of the environment, such as lighting, density of people, and sensors.

$$C = (t_n, rgb_n, night, type) \tag{3}$$

where (t_n, rgb_n) are, respectively, the number of thermal and RGB detections; *night* is a categorical variable that defines if the image was taken at night (*i.e.*, $night = 1$) or not (*i.e.*, $night = 0$). And $type = 1$ if the detection came from the thermal modality, or $type = 0$ if RGB.

• **Rescoring Feature Vector**: the overall rescoring feature vector is defined as the union of all aforementioned features, that is:

$$F = S \cup P \cup C \tag{4}$$

TABLE I
SUMMARY OF DATASETS FOR OBJECT DETECTION THAT CONTAIN RGB AND THERMAL IMAGES

Dataset	Year	Sensors	Description
OSU-CT [14]	2007	TIR, RGB	Thermal and RGB pictures taken in an outdoor urban environment.
KAIST [22]	2015	TIR, RGB	Dataset built purposely to increase the amount and quality of datasets available for thermal/RGB image pairing.
CVC-14 [36]	2016	FIR, RGB	Day and night sets of Thermal and RGB sequences.
MODAV [37]	2017	NIR, MIR, FIR, RGB	A novel multispectral dataset generated for autonomous vehicles which consists of RGB, NIR, MIR, and FIR images, taken at day and nighttime.
VIPER [38]	2017	LWIR, RGB	Data from pictures and videos taken in an outdoor train station in Brugge, Belgium.
PST900 [39]	2019	LWIR, RGB	Data set containing indoor Thermal and RGB images in the context of the DARPA Subterranean Challenge.
FLIR [12]	2020	TIR, RGB	The aligned version of the FLIR dataset, which contains synced annotated thermal imagery and non-annotated RGB imagery for reference.
LLVIP [40]	2021	TIR, RGB	This dataset contains 30976 images, or 15488 pairs, most of which were taken at very dark scenes, and all of the images are strictly aligned in time and space.
UMA-SAR [41]	2021	RGB, TIR	Collection of multimodal raw data captured from a manned all-terrain vehicle in the course of two realistic outdoor SAR exercises for actual emergency responders conducted in Spain in 2018 and 2019.
In-House	2023	RGB, TIR	Dataset acquired in-house by a team of students, containing 1282 aligned and manually annotated Thermal and RGB image pairs.

The feature vector is fed to a MLP, composed of a fully-connected layer with sigmoid activation function. The layer has 24 neurons. At the output of the neural network, a single value between 0 and 1 is generated, which is the new confidence score for that detection. With this rescoring approach, applied at the end of the thermal and RGB detection pipelines, the aim is to confer greater detection flexibility that allows for the model to be calibrated by considering context clues from the multi-modal detections.

The model was trained for $N_e = 1000$ epochs. Thus, for each epoch, a fitness value was calculated, which follows the same rules as YOLOv5, *i.e.*, the fitness considers 10% of the AP@50 value and 90% of the AP@50:95 value, granting a fitness value between 0 and 1 expressed by

$$fitness = 0.1(AP_{50}) + 0.9(AP_{50:95}).$$

If the fitness value did not improve for $N_p = 75$ consecutive epochs (patience parameter), the learning rate was then reduced by 10%, and the epoch with the best results was loaded so that the training could continue from there. Furthermore, if the fitness value did not improve for N_{tp}=500 consecutive epochs, despite the previous patience mechanism, then the training would stop and the epoch with the best results would be the chosen-trained model. The initial learning rate was $lr = 0.035$ and the batch size was 256. These hyperparameters were tuned experimentally.

IV. EXPERIMENTS AND RESULTS

In this section, the datasets used to support the experiments are described, followed by the performance measures and the achieved results.

A. Datasets

Two datasets were used to train and evaluate the proposed models, being: FLIR (outdoor) and a Thermal/RGB dataset made in-house (indoor). The FLIR dataset contains around 10 000 manually annotated Thermal images and their corresponding RGB images for reference, collected during both day and nighttime conditions. The version used in this work comes from [12], where the authors removed the unaligned visible-thermal pairs and ended up with 5 142 image pairs. Furthermore, the thermal resolution of the images in this dataset closely resembles our in-house dataset. The in-house dataset contains 1 282 thermal and RGB images manually annotated (757 training and validation, and 525 for testing), acquired using a mobile robot, driving in human-populated indoor scenes, containing samples from areas in low lighting conditions.

B. Evaluation Metrics

To evaluate the proposed approaches, we employed supervised detection metrics used in the COCO detection challenge, being: $mAP@50$, the mAP (Mean Average Precision) applied with an IoU threshold of 0.5; $mAP@50{:}95$ the mAP applied at IoU thresholds between 0.5 and 0.95 at a step of 0.05. And, an additional metric called Log-Average Miss Rate (LAMR) [43].

C. Results

Table II shows the results of the baseline (*i.e.*, modality-specific networks), and early and late fusion approaches on the FLIR and in-house datasets. In the **baseline** methods, in which a Yolov5-small was trained for each specific modality, the thermal modality achieved the best performance, probably due to the contrast between the humans and the background. Moreover, the results also demonstrated that human detection in outdoor environments presents a greater challenge due to the complexity of the environment, especially for RGB images alone.

The **early-fusion** method improved by 18.73% the $mAP@50$ on the FLIR dataset compared to the RGB

TABLE II
RESULTS OF THE FUSION METHODS IN THE TESTING DATASET

Model	Dataset	Modality	$\mathbf{mAP@50}$ *(%)*	$\mathbf{mAP@50:95}$ *(%)*	***LAMR (%)***
Baseline	FLIR	*thermal*	77.236	40.524	**43.049**
		RGB	57.311	23.178	63.738
	In-House	*thermal*	83.868	**52.864**	**25.738**
		RGB	79.674	46.269	34.383
Early Fusion	FLIR	*thermal + RGB*	76.041	37.421	44.946
	In-House	*thermal + RGB*	75.11	43.09	41.51
Late Fusion	FLIR	*thermal + RGB*	**79.355**	**41.08**	43.321
	In-House	*thermal + RGB*	**84.216**	52.158	27.628

TABLE III
PERFORMANCE OF BOUNDING BOXES MERGING TECHNIQUES

Merging Strategy	Metric	FLIR	In-House
Biggest	*AP@50*	83.308	85.519
	AP@50:95	40.201	44.091
	LAMR	29.038	23.99
Mean	*AP@50*	83.412	86.015
	AP@50:95	41.346	46.149
	LAMR	28.59	20.798
Weighted Mean	*AP@50*	83.191	86.067
	AP@50:95	40.907	45.891
	LAMR	29.749	20.348
Neural Network	*AP@50*	**83.484**	**86.189**
	AP@50:95	41.531	50.985
	LAMR	28.128	19.610

baseline. However, it decreases by 1.19% compared to the thermal modality. These results support the premise that thermal data can help improve detection, especially in low-light environments, present in the FLIR dataset. However, in the in-house dataset, the thermal data concatenated to the RGB channels showed no improvement for an early-fusion approach.

Similarly, the **late-fusion** improved by 22.04% and 2.12% the $mAP@50$ when compared to the RGB and thermal **baselines** on the FLIR dataset, respectively. In the in-house dataset, it improves by 4.54% (RGB) and 0.348% (thermal). These results show that rescoring can be a good technique to slightly improve detection results by calibrating the detection scores. On detectors with already high baseline values, such as the in-house dataset baseline, the rescoring method does not seem to produce as many beneficial effects. On the other hand, the results seem to show a correlation between the number of samples in the dataset and the performance of the rescoring algorithm on the metrics, since performances overall seem to be better on the FLIR dataset than the in-house dataset.

Additionally, Table III shows the performance that four bounding box merging techniques have on the **late-fusion** method, being:

• **Biggest**: this case, both bounding boxes are merged such that the new bounding box covers the entire areas of both bounding boxes;

• **Mean**: in this case, the (x, y) center of the new bounding box is halfway between the RGB and Thermal bounding box geometrical centers;

•**Weighted Mean**: same strategy as *mean*, but instead of the new center being halfway between RGB and Thermal centers, it is shifted to be closer to the bounding box with the biggest area;

•**Neural Network**: instead of taking predetermined variations of (x, y), width and height like in the previous strategies, this one consists of adding four more outputs to the MLP. These four outputs correspond to the x, y, width and height variations that the merged bounding box may take, and the MLP is trained to not only rescore the detection, but also to get the variations that best merge the two bounding boxes together.

For both FLIR and in-house dataset, the best results seem to come from the "Neural Network" strategy. For the FLIR dataset, throughout all strategies, we see an increase in the $mAP@50$ between 5.732 and 6.248, in comparison with Non-Maximum Suppresion (NMS) used in the rescoring algorithm, which provided only an increase of 2.119. For the in-house dataset, we see an increase ranging between 1.652 and 2.322 in comparison to 0.348. This means that creating a new bounding box out of the geometric information of the thermal and RGB bounding boxes provides better results overall.

V. CONCLUSION AND FUTURE WORK

This work concentrated on the human/people detection domain, using RGB and/or Thermal (long-wave IR) cameras, which finds applications in robotics, surveillance, and autonomous driving. This study is focused on the development and implementation of fusion or combination strategies that use both modalities to output new detections with calibrated confidence scores and geometrical properties which increase the accuracy and robustness of the detections. YOLOv5 has been used as the baseline for both single and combined modalities. Besides the FLIR dataset, we have collected RGB and Thermal images from cameras mounted onboard a mobile robot in real-world indoor conditions. This 'in-house' dataset complemented the experiments.

In terms of sensor/data fusion strategies, in this paper, we have implemented both early and late fusion approaches and reported the finding results. Detailed experiments are described, and the achieved results allow us to conclude that the combination of RGB and Thermal improves the results in terms of performance measures in comparison to the single-modality baselines.

In challenging real-world applications involving robots or autonomous vehicles, safety is of major concern. Thus, combining more than one sensor would allow more redundant and complementary solutions, reflecting on the robustness of the systems. The use of contextual information and multimodal combination strategies shows great promise in performance results, but also great flexibility; many more strategies can be suggested and studied, which can be considered for future work.

VI. ACKNOWLEDGMENTS

This paper has been partially supported by the project GreenBotics ref. PTDC/EEI-ROB/2459/2021 funded by FCT, Portugal.

REFERENCES

[1] C. Urmson and W. R. Whittaker, "Self-driving cars and the urban challenge," *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 66–68, 2008.

[2] A. Solichin, A. Harjoko, and A. E. Putra, "A survey of pedestrian detection in video," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 10, 2014.

[3] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[4] T. Gandhi and M. M. Trivedi, "Pedestrian protection systems: Issues, survey, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, pp. 413–430, 2007.

[5] D. Gerónimo, A. M. López, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1239–1258, 2010.

[6] C. Premebida, J. Carreira, J. Batista, and U. Nunes, "Pedestrian detection combining RGB and dense LIDAR data," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4112–4117.

[7] A. González, G. Villalonga, J. Xu, D. Vázquez, J. Amores, and A. M. López, "Multiview random forest of local experts combining RGB and LIDAR data for pedestrian detection," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015.

[8] M. Fürst, O. Wasenmüller, and D. Stricker, "LRPD: Long range 3D pedestrian detection leveraging specific strengths of LiDAR and RGB," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[9] J. Schlosser, C. K. Chow, and Z. Kira, "Fusing LIDAR and images for pedestrian detection using convolutional neural networks," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[10] M. M. Islam, A. A. R. Newaz, and A. Karimoddini, "A pedestrian detection and tracking framework for autonomous cars: Efficient fusion of camera and LiDAR data," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2021.

[11] D. Olmeda, C. Premebida, U. Nunes, J. M. Armingol, and A. de la Escalera, "Pedestrian detection in far infrared images," *Integr. Comput.-Aided Eng.*, vol. 20, no. 4, p. 347–360, oct 2013.

[12] H. Zhang, E. Fromont, S. Lefevre, and B. Avignon, "Multispectral fusion for object detection with cyclic fuse-and-refine blocks," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020.

[13] J. Davis and V. Sharma, "Robust detection of people in thermal imagery," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 2004.

[14] V. James W.Davis, "Background-subtraction using contour-based fusion of thermal and visible imagery," *Computer Vision and Image Understanding*, 2007.

[15] W. Li, D. Zheng, T. Zhao, and M. Yang, "An effective approach to pedestrian detection in thermal imagery," in *2012 8th International Conference on Natural Computation*, 2012.

[16] A. Königs and D. Schulz, "Evaluation of thermal imaging for people detection in outdoor scenarios," in *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2012.

[17] T. Guo, C. P. Huynh, and M. Solh, "Domain-adaptive pedestrian detection in thermal images," in *2019 IEEE international conference on image processing (ICIP)*. IEEE, 2019, pp. 1660–1664.

[18] C. Devaguptapu, N. Akolekar, M. M Sharma, and V. N Balasubramanian, "Borrow from anywhere: Pseudo multi-modal object detection in thermal imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[19] F. Almasri and O. Debeir, "Multimodal sensor fusion in single thermal image super-resolution," in *Computer Vision–ACCV 2018 Workshops: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers 14*. Springer, 2019.

[20] L. Susperregi, A. Arruti, E. Jauregi, B. Sierra, J. M. Martínez-Otzeta, E. Lazkano, and A. Ansuategui, "Fusing multiple image transformations and a thermal sensor with kinect to improve person detection ability," *Engineering Applications of Artificial Intelligence*, 2013.

[21] C. Palmero, A. Clapés, C. Bahnsen, A. Møgelmose, T. B. Moeslund, and S. Escalera, "Multi-modal RGB–Depth–Thermal human body segmentation," *International Journal of Computer Vision*, vol. 118, pp. 217–239, 2016.

[22] S. Hwang, J. Park, N. Kim, Y. Choi, and I. S. Kweon, "Multispectral pedestrian detection: Benchmark dataset and baselines," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[23] M. P. Marina Ivašić-Kos, Mate Krišto, "Human detection in thermal imaging using YOLO," in *ICCTA '19: Proceedings of the 2019 5th International Conference on Computer and Technology Applications*, April 2019.

[24] D. Pei, M. Jing, H. Liu, F. Sun, and L. Jiang, "A fast RetinaNet fusion framework for multi-spectral pedestrian detection," *Infrared Physics & Technology*, vol. 105, p. 103178, 2020.

[25] Z. Cao, H. Yang, J. Zhao, S. Guo, and L. Li, "Attention fusion for one-stage multispectral pedestrian detection," *Sensors*, vol. 21, no. 12, p. 4184, 2021.

[26] Y. Xue, Z. Ju, Y. Li, and W. Zhang, "MAF-YOLO: Multi-modal attention fusion based yolo for pedestrian detection," *Infrared Physics & Technology*, vol. 118, p. 103906, 2021.

[27] H. Zhang, E. Fromont, S. Lefèvre, and B. Avignon, "Guided attentive feature fusion for multispectral pedestrian detection," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 72–80.

[28] F. Qingyun and W. Zhaokui, "Cross-modality attentive feature fusion for object detection in multispectral remote sensing imagery," *Pattern Recognition*, 2022.

[29] K. Chen, J. Liu, and H. Zhang, "Igt: Illumination-guided rgb-t object detection with transformers," *Knowledge-Based Systems*, 2023.

[30] H. Zhou, M. Sun, X. Ren, and X. Wang, "Visible-thermal image object detection via the combination of illumination conditions and temperature information," *Remote Sensing*, 2021.

[31] Y. Zhu, X. Sun, M. Wang, and H. Huang, "Multi-modal feature pyramid transformer for rgb-infrared object detection," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[32] K. Roszyk, M. R. Nowicki, and P. Skrzypczyński, "Adopting the yolov4 architecture for low-latency multispectral pedestrian detection in autonomous driving," *Sensors*, vol. 22, no. 3, p. 1082, 2022.

[33] J. Wagner, V. Fischer, M. Herman, S. Behnke, *et al.*, "Multispectral pedestrian detection using deep fusion convolutional neural networks." in *ESANN*, vol. 587, 2016, pp. 509–514.

[34] Z. A. Shaikh, D. Van Hamme, P. Veelaert, and W. Philips, "Probabilistic fusion for pedestrian detection from thermal and colour images," *Sensors*, 2022.

[35] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto, and U. J. Nunes, "Multimodal vehicle detection: fusing 3d-lidar and color camera data," *Pattern Recognition Letters*, vol. 115, pp. 20–29, 2018.

[36] A. González, Z. Fang, Y. S. Salas, J. Serrat, D. Vázquez, J. Xu, and A. M. López, "Pedestrian detection at day/night time with visible and fir cameras: A comparison," *Sensors (Basel, Switzerland)*, 2016.

[37] K. Takumi, K. Watanabe, Q. Ha, A. Tejero-De-Pablos, Y. Ushiku, and T. Harada, "Multispectral object detection for autonomous vehicles," in *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, 2017.

[38] K. Van Beeck, K. Van Engeland, J. Vennekens, and T. Goedemé, "Abnormal behavior detection in LWIR surveillance of railway platforms," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2017.

[39] S. S. Shivakumar, N. Rodrigues, A. Zhou, I. D. Miller, V. Kumar, and C. J. Taylor, "PST900: RGB-Thermal calibration, dataset and segmentation network," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[40] X. Jia, C. Zhu, M. Li, W. Tang, and W. Zhou, "LLVIP: A visible-infrared paired dataset for low-light vision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2021.

[41] J. Morales, R. Vázquez-Martín, A. Mandow, D. Morilla-Cabello, and A. García-Cerezo, "The UMA-SAR dataset: Multimodal data collection from a ground vehicle during outdoor disaster response training exercises," *The International Journal of Robotics Research*, 2021.

[42] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[43] M. Braun, S. Krebs, F. Flohr, and D. M. Gavrila, "Eurocity persons: A novel benchmark for person detection in traffic scenes," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1844–1861, 2019.

Index of Authors

A

Achterhold, Jan 185
Aerts, Peter 150
Agarwal, Rajat 374
Aguiar, A. Pedro 374
Ahmad, Aamir 221
Ajoudani, Arash 89
Akhtyamov, Timur 192
Amigoni, Francesco 171
Anand, Pallov 374
Andreasson, Henrik 42
Angarano, Simone 1
Annabi, Louis 330
Anthierens, Cedric 354
Antonazzi, Michele 48
Arapis, Dimitrios 83
Arlotta, Andrea 318
Arrigoni, Stefano 6
Arzberger, Fabian 200
Atas, Fetullah 229

B

Baptista, Rui 367
Barber, Ramon 69, 76
Basilico, Nicola 48, 171
Bayer, Jan 277
Bedín, Sebastián 263
Bektaş, Kemal 348
Belter, Dominik 243
Bertoglio, Riccardo 6, 12
Bhattacharyya, Raunak 164
Bildstein, Hugo 336
Boche, Simon 117
Boedecker, Joschka 185
Bonetti, Alessandro 131
Bonetto, Elia 221
Borghese, N. Alberto 48
Borrmann, Dorit 200
Bozma, H. Işıl 348
Bregler, Kevin 237
Brudermüller, Lara 164

C

Cadenat, Viviane 19, 336
Cai, Hanyu 157
Carini, Veronica 6
Carmesin, Sarah 311
Castaman, Nicola 97
Catalano, Nico 12
Chiaberge, Marcello 1
Chiatti, Agnese 12
Christensen, Anders 63
Cielniak, Grzegorz 26, 229
Civera, Javier 263
Cvišić, Igor 324

D

Deckerová, Jindřiška 179
Demeester, Eric 150
Dogru, Sedat 298
Durand-Petiteville, Adrien ... 19, 336

F

Faigl, Jan 179, 277, 343
Ferrer, Gonzalo 192
Fraundorfer, Friedrich 34
Frommel, Christoph 97

G

Galland, Stephane 124
Garrote, Luís 380
Gasparri, Andrea 318
Gatti, Matteo 12
Geles, Ismail 34
Ghidoni, Stefano 97
Gomes, Iago 380
Gottardi, Alberto 97
Grimstad, Lars 229
Gross, Horst-Michael 250
Guidetti, Simone 131
Gupta, Himanshu 42
Guttikonda, Suresh 185

H

Hadviger, Antea 324
Hanheide, Marc 26
Hawes, Nick 164
Henke, Christoph 269
Hertzberg, Joachim 257
Hilaire, Vincent 124
Hroob, Ibrahim 26
Hugel, Vincent 354
Hulchuk, Vsevolod 277
Höfer, Timon 144

I

Iqbal, Naeem 257

J

Jami, Milad 83
Julier, Simon 42

K

Kashirin, Aleksandr 192
Keyvan, Erhan Ege 305
Khaldi, Belkacem 305
Khorrami, Farshad 111
Kiefer, Benjamin 144
Kirsch, André 138
Koenig, Matthias 138
Koledić, Karlo 214
Krajník, Tomáš 56
Kraus, Werner 237
Krause, Christoph 257
Krishnamurthy, Prashanth ... 111
Kucner, Tomasz Piotr 292
Kulich, Miroslav 311
Kučerová, Kristýna 179

L

Lacerda, Bruno 164
Lamotte, Olivier 124
Li, Haolong 185
Lilienthal, Achim 42
Lippi, Martina 318
Liu, Xinyu 56
Lodigiani, Giacomo 171
Lopez, Blanca 76
Luperto, Matteo 48

M

Magalhães, Hugo 367
Magnusson, Martin 42
Mansouri, Masoumeh ... 311, 360
Marković, Ivan 214, 324
Marques, Lino 298, 367
Martini, Mauro 1
Marusic, Aleksa 330
Masuzawa, Hiroaki 208
Matsuzaki, Shigemichi 208
Matteucci, Matteo 6, 12, 285
Mendez, Alberto 69
Menegatti, Emanuele 97
Mentasti, Simone 285
Mishra, Manav 374
Mitrevski, Alex 269
Miura, Jun 208
Mohr, Ludwig 34
Molina, Sergi 26
Montenegro, Sergio 200
Mora, Alicia 69
Moreno, Luis 76
Mota, Kennedy 380
Müller, Steffen 250
Müller, Trsitan 250

N

Nalpantidis, Lazaros 83
Navone, Alessandro 1
Neto de Carvalho de Andrade Tavares, João 105
Nguyen, Sao Mai 330
Niemeyer, Mark 257
Nitsche, Matias Alejandro . . . 263
Niturkar, Pranav 374
Nüchter, Andreas 200

O

Okunevich, Iaroslav 124
Ostuni, Andrea 1
Ou, Ni 157

P

Petrović, Ivan 214, 324
Plöger, Paul G. 269
Polvara, Riccardo 26
Postnikov, Aleksey 192
Prados, Adrian 76
Premebida, Cristiano 380

Q

Qizilbash, Agha Ali Haider . . 237

R

Raiola, Gennaro 89
Rano, Inaki 63
Riechmann, Malte 138
Rollo, Federico 89
Rothe, Julian 200
Rozsypálek, Zdeněk 56
Ruichek, Yassine 124

S

Sabattini, Lorenzo 131
Sahin, Erol 305
Şahin, Mehmet 305
Schoenheits, Manfred 97
Sharma, Ekansh 269
Shi, Junyi 292
Shilova, Liubov 124
Silva, Carlos A. 298
Slaets, Peter 150
Sousa, Elísio 380
Speranza, Claudia 285
Staniaszek, Michal 164
Steidle, Florian 117
Stephan, Benedict 250
Štironja, Vlaho-Josip 324
Street, Charlie 360
Stueckler, Joerg 185
Stürzl, Wolfgang 117
Sujit, P. B. 374

T

Tapus, Adriana 330
Terreran, Matteo 97
Tortorici, Ornella 354
Triebel, Rudolph 117
Tsagarakis, Nikolaos 89
Turgut, Ali Emre 305
Tzes, Anthony 111

U

Unlu, Halil Utku 111
Uzawa, Yoshinobu 208

V

Vale, Alberto 105
Vallone, Andrea 83
Valouch, David 343
Vatan, S. Batuhan 348
Ventura, Rodrigo 105
Villemazet, Antoine 19
Vražić, Sacha 324

W

Wang, Junzheng 157
Wiecha, Fabian 200
Wolf, Denis 380
Woller, David 311

Y

Yadav, Anoj Kumar 237
Yan, Zhi 124

Z

Zell, Andreas 144
Zevering, Jasper 200
Zhang, Weijian 360
Zieliński, Mikołaj 243
Zunino, Andrea 89