UNIVERSITY OF TURKU

FEDERATED LEARNING ENHANCED MULTI-MODAL SENSING AND PERCEPTION IN A COLLABORATIVE MULTI-ROBOT SYSTEM

Xianjia Yu

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
FEDERATED LEARNING ENHANCED MULTI-MODAL SENSING AND PERCEPTION IN
A COLLABORATIVE MULTI-ROBOT SYSTEM

University of Turku

Faculty of Technology
Department of Computing
Information and Communication Technology
Doctoral Programme in Technology (DPT)

Supervised by

Professor, Tomi Westerlund
University of Turku

Professor, Zhuo Zou
Fudan University

Professor, Jukka Heikkonen
University of Turku

Reviewed by

Lecturer (Privatdozent), Jens Behley
University of Bonn

Assistant Professor, Tomasz Kucner
Aalto University

Opponent

Associated Professor, John Folkesson
KTH Royal Institute of Technology

Serial F 46
ISBN 978-951-29-9888-3 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)
Painosalama Oy, Turku, Finland, 2024

To my wife, Yuki, my daughter, Yori, my parents, Changjin Yu and Shangying Zhu, my sister, Qingxiu Yu, my brother-in-law, Zuying Bai, and my nephew, Sheng Bai

UNIVERSITY OF TURKU
Faculty of Technology
Department of Computing
Information and Communication Technology
YU, XIANJIA: Federated Learning Enhanced Multi-Modal Sensing and Perception in a Collaborative Multi-Robot System
Doctoral dissertation, 183 pp.
Doctoral Programme in Technology (DPT)
June 2024

ABSTRACT

Multi-robot systems are increasingly essential across a wide array of sectors, such as industrial automation, transportation, and search and rescue. The key to these systems lies in the capabilities of agents to collaboratively perceive, comprehend, and reason about their surroundings, thereby attaining advanced situational awareness. Recent advances in artificial intelligence, especially in the field of deep learning (DL), have increased the ability of multi-robot systems to effectively utilize and understand data produced by various sensors. Despite numerous efforts to integrate multiple sensors, this area remains complex and challenging due to heterogeneous, unstructured, and cluttered deployment environments. Furthermore, these operating scenarios vary considerably across different settings, including hospitals, private residences, ports, and other contexts where privacy and security prevail.

This dissertation addresses these challenges by integrating multi-modal sensors to enhance high-level robot perception across multiple agents while ensuring security and privacy through Federated Learning (FL). FL, a privacy-preserving DL method, distributes learning across isolated data silos, enabling secure knowledge sharing among robots via model transfers instead of direct data exchanges.

The research begins by investigating the limitations of existing multi-modal sensor datasets and employing diverse sensors, including LiDAR (spinning and solid-state LiDARs), visual sensors, Inertial Measurement Units (IMUs), and Ultra-Wideband (UWB), to create more comprehensive datasets. After benchmarking current state-of-the-art SLAM and LiDAR odometry (LO) algorithms, the study develops novel multi-robot relative localization approaches as a foundation for other perception tasks. It then explores using LiDAR-generated images and solid-state LiDAR to enhance UAV tracking and LO. Finally, the effectiveness of FL is demonstrated through a case study on multi-robot visual obstacle avoidance (VOA), transitioning from simulation to real-world scenarios. By incorporating LiDAR and cameras in real-world applications, the research achieves lifelong learning on VOA within the FL framework. This case study highlights FL's practical applications and advantages, suggesting its potential generalizability across a broad range of robotic perception tasks.

KEYWORDS: Multi-modal sensors, LiDAR, Solid-state LiDAR, LiDAR as a camera, UWB, FL, Odometry, relative localization, UAV tracking, obstacle avoidance

TURUN YLIOPISTO
Faculty of Technology
Department of Computing
Information and Communication Technology
YU, XIANJIA: Federated Learning Enhanced Multi-Modal Sensing and Perception in a Collaborative Multi-Robot System
Väitöskirja, 183 s.
Doctoral Programme in Technology (DPT)
June 2024

TIIVISTELMÄ

Monirobottijärjestelmät ovat yhä tärkeämpiä monilla eri aloilla, kuten teollisuuden automaatiossa, kuljetuksessa ja etsintä- ja pelastustehtävissä. Näiden järjestelmien yhtenä mahdollistajana on niiden kyky havaita, ymmärtää ja selittää ympäristöään saavuttaen näin kehittyneen tilannetietoisuuden. Tilannetietoisuuden parantumisessa merkittävänä tekijänä on ollut viime aikojen edistysaskeleet tekoälyssä, erityisesti syväoppimisessa, jotka ovat lisänneet monirobottijärjestelmien kykyä hyödyntää erilaisten sensorien tuottamaa dataa. Edistysaskeleista huolimatta tämä alue on säilynyt monimutkaisena ja haastavana heterogeenisten ja jäsentymättömien käyttöympäristöjen vuoksi. Lisäksi toimintaskenaariot vaihtelevat huomattavasti eri ympäristöissä, kuten sairaaloissa, satamissa ja muissa yksityisyyttä ja turvallisuutta korostavissa paikoissa.

Tämä väitöskirja käsittelee näitä haasteita integroimalla multimodaalisia sensoreita parantaakseen moni-robottijärjestelmien kyvykkyyttä havainnoida ympäristöä. Samalla varmistetaan turvallisuus ja yksityisyys hyödyntämällä yhdistettyä oppimista. Tutkimus keskittyy aluksi tunnistamaan olemassa olevien monimodaalisten sensoridata-aineistojen rajoituksia ja hyödyntämään erilaisia sensoreita, kuten laserkeilausta, visuaalisia sensoreita, inertiamittausta ja ultralaajakaistaa luodakseen kattavampia data-aineistoja. Sen jälkeen, kun tämän hetken parhaimmat algoritmit samanaikaiseen paikannukseen ja kartoitukseen sekä laserkeilainpohjaiseen matkan mittaamiseen on arvioitu, tutkimuksessa kehitetään uusia robottiparven jäsenten suhteelliseen paikantamiseen liittyviä lähestymistapoja muiden havainnointitehtävien perustaksi. Seuraavaksi tutkitaan erilaisten laserkeilaimien tuottamien kuvien käyttöä droonien seuraamisessa ja niiden kulkeman matkan arvioimisessa. Lopuksi yhdistetyn oppimisen tehokkuutta testataan siirtämällä simulaatioympäristössä toteutettu monirobottien visuaalinen esteiden välttämistehtävä oikeilla roboteilla suoritettavaksi. Testissä hyödynnettiin sekä laserkeilausta että kameroita, joiden avulla saavutettiin pitkänajan oppiminen visuaalisten esteiden välttämisessä yhdistetyn oppimisen ympäristössä. Käytännössä suoritettu testi korostaa yhdistetyn oppimisen käytettävyyttä ja hyödyllisyyttä käytännön sovelluksissa antaen viitteitä sen mahdolliseen yleistettävyyteen laajalle joukolle robottien havaintotehtäviä.

ASIASANAT: Monimodaaliset sensorit, laserkeilaus, kamerat, ultralaajakaista, satelliittipaikannus, yhdistetty oppiminen

Acknowledgements

The completion of this doctoral research has been a long and challenging journey, especially after spending several years in industry before returning to academia. As I write these words, I am filled with a deep sense of gratitude, and I wish to take this opportunity to thank all those who have supported me throughout this journey.

First and foremost, I would like to express my sincere gratitude to my supervisor and friend, Prof. Tomi Westerlund, for his invaluable guidance and support, both in my academic endeavors and personal life over the past years. I have known Tomi since my master's studies, and he has always treated us with kindness, offering his assistance and advice whenever possible. I am also deeply thankful to Prof. Zhuo Zou for his insightful discussions and for arranging my visit to Fudan University, Shanghai, China. Additionally, I extend my appreciation to Prof. Jukka Heikkonen for the enriching conversations we shared during the early stages of my research.

I would like to express my deepest gratitude to my wife, Yuki Takahashi, for her unwavering support in both my personal life and academic pursuits. Without her encouragement, this doctoral degree would likely remain a mere figment of my imagination. Although I am certain she may never read this thesis, I hope that my daughter, Yori, will one day inform her mother that I expressed my heartfelt thanks in these pages. Yori, now two years old and 3 month old, has been a source of immense joy, even as she occasionally distracts her father while writing this acknowledgment. Nonetheless, I must say "Kiitos" to her. I am also and always deeply appreciative of my parents, Changjin Yu and Shangying Zhu, and sister, Qingxiu Yu for their unconditional support throughout my life. I hope that we will always remain a close-knit family and that they continue to live healthy and happy lives.

Moreover, I would like to express my sincere gratitude to my irreplaceable friend, Jorge, who inspired me to start this doctoral journey and provided tremendous support throughout. His intelligence, creativity, and boundless energy have been a constant source of inspiration. At the beginning of my doctoral studies, Jorge, Qingqing, and I spent countless hours together conducting experiments and teaching. These memories are ones I will cherish and proudly share with others. I also extend my deepest thanks to Qingqing. Our academic paths have been closely intertwined since our master's studies, sharing the same thesis supervisor, the same former employer, and the same doctoral supervisor and lab. I am grateful for all the support you have provided.

I would also like to acknowledge my primary co-authors and friends, Sahar Salimpour, Hasier, Iacopo Catalano, and Haizhou Zhang, for their invaluable assistance and the many productive discussions we had. Without their collaboration, I would not have made it this far. My gratitude also extends to my friends Salma Salimi, Morón Paola Torrico, Jiaqiang Zhang, Farhad Keramat, Lei, Qianqian, and all others who have supported me throughout this journey. I will always treasure the time we spent together.

Furthermore, I would like to extend my thanks to Markku Noroaho from Kaptas Oy. We've had many insightful conversations over the past year, and his help has been invaluable. I would also like to thank all my colleagues at Kaptas Oy for their kindness and support. I wish them all the best and look forward to us working together to make Kaptas even better in the future.

Lastly, I would like to express my appreciation to the pre-examiners for their valuable feedback. I also wish to thank the staff at the University of Turku Graduate School, the Finnish Foundation for Technology Promotion (TES), and the Nokia Foundation. Their support has been instrumental in the completion of this thesis.

Thank you all. A doctoral degree is the highest academic degree an individual can attain. However, the pursuit of knowledge and research is an ongoing journey, akin to an endless game; we cannot stop at this checkpoint. Let us continue to encourage and support each other in our future endeavors.

September 18, 2024
Xianjia Yu

XIANJIA YU

Xianjia Yu holds a M.Eng in Electronics Engineering from Fudan University, Shanghai, China, and a MSc in Information and Communication Technology from the University of Turku, Finland, both obtained in 2018. Since January 2021, Xianjia has been a doctoral researcher at the Turku Intelligent and Embedded Robotic System (TIERS) Lab, Department of Computing, University of Turku, Finland. Previously, he worked as a senior robotic algorithm engineer in Shanghai, China. His research interests include robotics, multi-modal sensing and perception, machine learning, and computer vision.

Table of Contents

Abbreviations

AUC	Area Under the ROC Curve
APE	Absolute Pose Error
ATE	Absolute Trajectory Error
BA	Bundle Adjustment
CNN	Convolutional Neural Network
CT-ICP	Continuous-Time Iterative Closest Point
CTRV	Constant Turn Rate and Velocity
DL	Deep Learning
DLTs	Distributed Ledger Technologies
DoF	Degrees of Freedom
DRL	Deep Reinforcement Learning
EKF	Extended Kalman Filter
EMG	Electromyography
FL	Federated Learning
FoV	Field of View
GICP	Generalized Iterative Closest Point
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
ICI	Inverse Covariance Intersection
ICP	Iterative Closest Point
IIoT	Industrial Internet of Things
IMU	Inertial Measurement Unit
IoT	Internet of Things
KD-Tree	K Dimension Tree
KF	Kalman Filter
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
LiDAR	Light Detection and Ranging
LO	LiDAR Odometry
LOAM	LiDAR Odometry and Mapping
LSTM	Long Short-Term Memory
MAV	Micro Aerial Vehicle
MOCAP	Motion Capture
MSE	Mean Square Error

MUI	Multi-User Interference
NDT	Normal Distributions Transform
NLP	Natural Language Processing
NLOS	Non Line of Sight
NNS	Nearest-Neighbor Search
PCL	Point Cloud Library
PF	Particle Filter
PTP	Precision Timestamp Protocol
ROI	Region of Interest
RL	Reinforcement learning
RMSE	Root Mean Square Error
RTK	Real-Time Kinematic Position
ROC	Receiver Operating Characteristic
ROS	Robot Operating System
RUGD	Robot Unstructured Ground Driving
SfM	Structure from Motion
Sim2Real	Sim-to-Real
SLAM	Simultaneous Localization and Mapping
SVM	Support Vector Machine
TDoA	Time of Arrival
ToF	Time of Flight
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VIO	Visual Inertial Odometry
UWB	Ultra-Wideband
VO	Visual Odometry
VSLAM	Visual Simultaneous Localization and Mapping

List of Original Publications

This dissertation is based on the following original publications, which are referred to in the text by their Roman numerals:

I **Yu Xianjia**, Jorge Peña Queralta, Jukka Heikkonen, Tomi Westerlund. Federated learning in robotic and autonomous systems. Procedia Computer Science, 2021; 191: 135-142.

II **Yu Xianjia**, Li Qingqing, Jorge Peña Queralta, Jukka Heikkonen, Tomi Westerlund. Applications of uwb networks and positioning to autonomous robots and industrial systems. Mediterranean Conference on Embedded Computing (MECO), 2021.

III **Yu Xianjia**, Li Qingqing, Jorge Peña Queralta, Jukka Heikkonen, Tomi Westerlund. Cooperative UWB-Based Localization for Outdoors Positioning and Navigation of UAVs aided by Ground Robots. IEEE International Conference on Autonomous Systems (ICAS). IEEE.

IV **Xianjia Yu**, Jorge Peña Queralta, Tomi Westerlund. Towards lifelong federated learning in autonomous mobile robots with continuous sim-to-real transfer. Procedia Computer Science. 2022;210:86-93.

V **Xianjia Yu**, Jorge Pena Queralta, Tomi Westerlund. Federated learning for vision-based obstacle avoidance in the internet of robotic things. International Conference on Fog and Mobile Edge Computing (FMEC), 2022.

VI **Xianjia Yu**, Sahar Salimpour, Jorge Peña Queralta, Tomi Westerlund. General-Purpose Deep Learning Detection and Segmentation Models for Images from a Lidar-Based Camera Sensor. Sensors, 23(6), p.2936.

VII **Xianjia Yu**, Haizhou Zhang, Sier Ha, Tomi Westerlund. LiDAR-Generated Images Derived Keypoints Assisted Point Cloud Registration Scheme in Odometry Estimation. Remote Sensing. 2023 Oct 23;15(20):5074.

VIII Li Qingqing, **Yu Xianjia**, Jorge Peña Queralta, Tomi Westerlund. Multimodal lidar dataset for benchmarking general-purpose localization and mapping algorithms. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2022 (pp. 3837-3844). IEEE.

IX Li Qingqing, **Yu Xianjia**, Jorge Peña Queralta, Tomi Westerlund. Adaptive lidar scan frame integration: Tracking known mavs in 3d point clouds. IEEE International Conference on Advanced Robotics (ICAR), 2021.

X Ha Sier, **Xianjia Yu**, Iacopo Catalano, Jorge Pena Queralta, Zhuo Zou, Tomi Westerlund. Uav tracking with lidar as a camera sensor in gnss-denied environments. International Conference on Localization and GNSS (ICL-GNSS), 2023.

XI Ha Sier, Li Qingqing, **Yu Xianjia**, Jorge Peña Queralta, Zhuo Zou, Tomi Westerlund. A benchmark for multi-modal lidar slam with ground truth in gnss-denied environments. Remote Sensing, 2023;15(13):3314.

XII Iacopo Catalano, **Xianjia Yu**, Jorge Peña Queralta. Towards robust uav tracking in gnss-denied environments: a multi-lidar multi-uav dataset. IEEE International Conference on Robotics and Biomimetics (ROBIO), 2023.

XIII Iacopo Catalano, Ha Sier, **Xianjia Yu**, Tomi Westerlund, Jorge Peña Queralta. Uav tracking with solid-state lidars: dynamic multi-frequency scan integration. International Conference on Advanced Robotics (ICAR), 2023.

The following related publications are not directly included in this thesis:

I Queralta Jorge Pena, Farhad Keramat, Salma Salimi, Lei Fu, **Xianjia Yu**, and Tomi Westerlund. Blockchain and emerging distributed ledger technologies for decentralized multi-robot systems. Current Robotics Reports 4, no. 3 (2023): 43-54.

II Morón Paola Torrico, Sahar Salimpour, Lei Fu, **Xianjia Yu**, Jorge Peña Queralta, and Tomi Westerlund. Benchmarking UWB-based infrastructure-free positioning and multi-robot relative localization: dataset and characterization. IEEE Sensors Applications Symposium (SAS), 2023.

III Salimpour Sahar, Paola Torrico Morón, **Xianjia Yu**, Tomi Westerlund, and Jorge Peña-Queralta. Exploiting redundancy for UWB anomaly detection in infrastructure-free multi-robot relative localization. Frontiers in Robotics and AI, 2023.

IV Zhang Jiaqiang, Farhad Keramat, **Xianjia Yu**, Daniel Montero Hernández, Jorge Peña Queralta, and Tomi Westerlund. Distributed robotic systems in the edge-cloud continuum with ros 2: A review on novel architectures and technology readiness. International Conference on Fog and Mobile Edge Computing (FMEC), 2022.

1 Introduction

In recent years, the integration of robotic technology has expanded beyond traditional manufacturing and industrial applications, penetrating a variety of sectors such as research, search and rescue operations, healthcare, agriculture, and logistics, and among others. This diversification in application areas underscores not only the advancements of robots but also highlights the growing dependency of various fields on automated systems for efficiency, precision, and safety. The actual applications of multiple agents in different scenarios require comprehensive understanding of the surroundings which are typically sophisticated and harsh at times. A single sensor modality cannot adequately address all scenarios due to either the complexity of the environment or the inherent limitations of the sensor. Despite the recent progress from related aspects including computation power, novel algorithms, Deep Learning (DL) methods, and multi-modal sensors. This topic is challenging yet popular.

The advancements in multi-modal sensors have provided researchers with enhanced opportunities to develop more sophisticated algorithms, enabling robots to perform tasks with increased efficiency and precision. Key sensors, including Light Detection and Ranging (LiDAR), visual sensors, and Inertial Measurement Units (IMU), are fundamental for perception tasks such as localization, mapping, and tracking. Among these, LiDAR has become an essential component of many autonomous systems, from state-of-the-art self-driving stacks [1] to aerial robots [2]. LiDAR odometry (LO), localization and mapping algorithms find applications in areas such as autonomous driving vehicles [3], Unmanned Aerial Vehicles (UAV) [4], and forest survey [5]. Key factors motivating the adoption of LiDARs in such systems include their long-range, accurate detection ability in 3D and robust performance in a variety of scenarios and environmental conditions. High-resolution multi-beam spinning LiDARs have enabled high degrees of situational awareness in mobile robotic solutions. However, the larger number of vertical channels in a mechanically spinning LiDAR has meant until recently a cost too high for wide adoption in mobile platforms. While costs have significantly reduced in recent years as the technology matured, solid-state LiDARs capable of generating high-density point clouds have emerged as an alternative. With a closer resemblance to visual sensors with limited field of view (FoV) and dense scene scanning, solid-state LiDARs offer high performance and relatively lower cost [6; 7]. These sensors also offer non-repetitive scanning patterns allowing for higher resolution outputs than spinning LiDARs [8].

(a) (b) (c) (d) (e)

Figure 1: The UAV point cloud data, obtained from solid-state LiDARs with varying integration times at the distance around 10 m, is presented. The point clouds depicted correspond to the UAV in Figure 1e during flight. Specifically, Figures 1a, 1b, 1c, and 1d illustrate the point clouds generated with integration times of 0.03 s, 0.05 s, 0.1 s, and 0.2 s, respectively. The color of the points is the intensities projected using a rainbow color palette.

However, the current body of research and existing datasets exhibit a notable deficiency in data derived from solid-state LiDARs. Furthermore, the studies specifically addressing solid-state LiDARs remain limited in scope. These LiDARs often present limited FoV, due to the absence of mechanical rotation and the use of various scanning modalities. Additionally, in contrast to spinning LiDARs, non-repetitive scanning patterns have emerged across different products, alongside variations in patterns even when they are repetitive. Figure 63a on page 126 shows the different point cloud integration times in a non-repetitive solid-state LiDAR scanning device. The restricted FoV, coupled with diverse scan patterns, introduces substantial differences for many standard LO, localization, mapping, and tracking algorithms. If existing algorithms can be adapted to better support these new types of LiDAR sensors, solid-state LiDARs have the potential to significantly enhance perception pipelines, including object detection and tracking, as well as scene understanding in various scenarios. Figure 1 shows the point cloud of an UAV with different integration time at the distance around 10 m. In comparison to the point cloud generated by a spinning LiDAR for the same type of UAV, as depicted in Figure 59 on page 119, the resulting point cloud exhibits significantly higher density. The detailed information about this data sequence can be found in Figure 23 on page 58.

We see particular potential in tasks in unstructured environments where geometric features are often too few and sparse compared to urban structured environment. While there are a large number of LiDAR datasets recorded with spinning LiDAR in diverse environments, there is a lack of multi-modal datasets enabling further adoption and understanding of different types of LiDARs and data processing algorithms.

As LiDARs measure the time of flight of a laser signal to objects in the environment, they are not influenced by changes in light such as darkness and daylight. Nowadays, apart from point cloud, LiDARs can obtain low-resolution images with

(a) Outdoors (RGB) (b) Outdoors (LiDAR)

(c) Indoors (LiDAR) (d) Indoors (RGB)

Figure 2: Samples of images utilized in this work. The outdoors sample includes a bicycle that is seen in both the RGB and LiDAR data, as well as several cars. In both indoors and outdoors images, a person behind the sensors does appear in the 360° LiDAR image but not in the RGB frame.

360° horizontal FoV by encoding either depth, reflectivity, or near-infrared light in the image pixels referred to LiDAR as a camera. Compared to conventional images from vision sensors, these images are more robust in harsh environments, including varying light conditions, rain, and fog. A sample of the data used in this dissertation is shown in Figure 2. Over the last decade, robotic perception algorithms have significantly benefited from the rapid advances in DL. Indeed, a significant amount of the autonomy stack of different commercial and research platforms relies on DL for situational awareness, especially vision sensors. In several studies, LiDAR point cloud data and image data from vision sensors have been used together in a variety of computer vision tasks, such as 3D object detection [9; 10]. However, the potential of general-purpose DL perception algorithms, specifically detection and segmentation neural networks, for processing image-like outputs of advanced LiDAR sensors is not fully explored. Only recently, such potential has been identified [11], but the existing literature lacks a more in-depth analysis of the potential of images captured from LiDAR sensors. We refer the reader to existing dataset papers with this type of data for a more in-depth characterization of the different types of images that the Ouster LiDARs can generate [12].

In addition to LiDAR as the primary sensor for sensing and perception, UWB technology, when fused with other sensors such as stereo cameras, has been employed for multi-robot relative localization. This integration lays the groundwork for

other perception tasks explored in this dissertation. Since UWB ranging sensors offer low-cost and centimeter-level out-of-the-box accuracy, they have gradually gained attention in autonomous systems applications, including UWB-based state estimation with and without fixed infrastructure, and UWB mesh sensor networks [13; 14]. This technology holds significant potential for relative state estimation in multi-robot systems [15; 16], a crucial yet still challenging research topic in GNSS-degraded environments and outdoors [17; 18]. Moreover, it can serve as the basis for collaborative tasks such as search and rescue, and terrain inspection [19], or extended to transitions of multi-robot systems between indoor and outdoor scenarios [20].

As a computation-consuming and data-driven approach, DL in robotics faces significant bottlenecks, particularly in scenarios involving low-cost robots and data-sensitive environments. In such cases, the collaborative performance of tasks by multiple robots and the sharing of knowledge, specifically DL models, becomes crucial. However, the deployment environments for these robots often include privacy-sensitive locations such as hospitals, private residences, and ports. With the exponential increase in the number of connected devices within the Internet of Things (IoT), the volume of generated and transmitted data has surged dramatically. The inefficiency of processing all this data centrally in the cloud has led to the emergence of new computing and networking paradigms in recent years [21]. Edge computing, which processes data close to the sources, offers clear benefits in terms of latency reduction and bandwidth savings [22]. Additionally, it enhances data privacy by ensuring that raw data does not travel far [23]. Meanwhile, data is being integrated into increasingly complex AI models, with DL becoming ubiquitous across multiple fields and application domains.

There has also been a growing awareness of the risks and drawbacks associated with sharing personal data over the internet. Federated Learning (FL) offers a solution to the challenge of distributed computing at the edge while preserving data privacy and leveraging DL solutions [24]. FL enables the distributed training of complex models across isolated data sources from remote nodes. The local training results, or updates to local models, are aggregated, for example, on a cloud server, and a global generalized model is shared back to the nodes. This process occurs without any raw data transmission [25].

FL provides a framework for more efficient learning in distributed autonomous systems and multi-robot systems. It enables collaborative learning across heterogeneous cloud and edge nodes, encompassing a wide range of robotic and autonomous systems. Figure 3 illustrates the conceptual framework of FL-enabled lifelong learning with sim-to-real (Sim2Real) transfer as proposed in this dissertation. Although cloud robotics can enhance scalability, collaborative knowledge sharing, and the development and operation of robotics and autonomous systems, challenges persist, particularly in areas such as security and reliable connectivity.

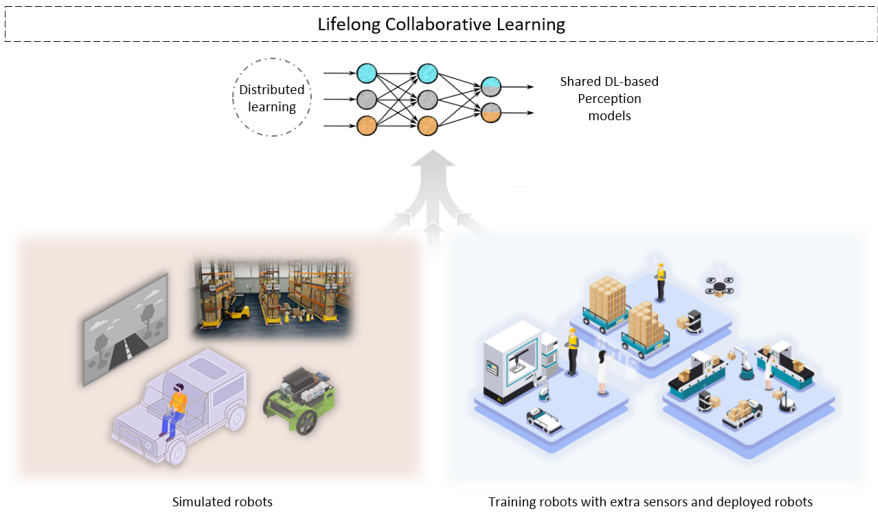


Figure 3: Conceptual illustration of federated lifelong learning with sim-to-real transfer as a deep learning-based perception model is trained in both a simulation and real robots, with potentially continuous updates.

1.1 Problem Statements

This thesis endeavors to investigate various challenges in the domain of robot perception within the framework of multi-robot systems. These challenges include the fusion strategy of sensors, the collaboration of multiple robots for the purpose of robotic perception, and the development of an overall framework for the understanding of situated environment across multiple robots while ensuring data privacy. These challenges, from different aspects, are outlined as follows:

- **Sensor Integration and Environmental Comprehension**
  How can diverse sensor technologies, such as LiDARs, visual sensors, UWB and IMUs, be effectively leveraged and integrated to enable robots to accurately perceive and navigate complex environments?

- **Efficiency in Collaborative Robotics**
  How can the efficiency of collaborative robotic systems be optimized to ensure synchronized task execution, seamless information exchange, and minimal operational redundancies among multiple robots?

- **Secure and Privacy-Preserving Knowledge Sharing**
  In the context of shared data environments, how can FL be effectively integrated into robotic systems without affect the quality of robotic perception tasks to enhance collaborative learning while ensuring data security and privacy?

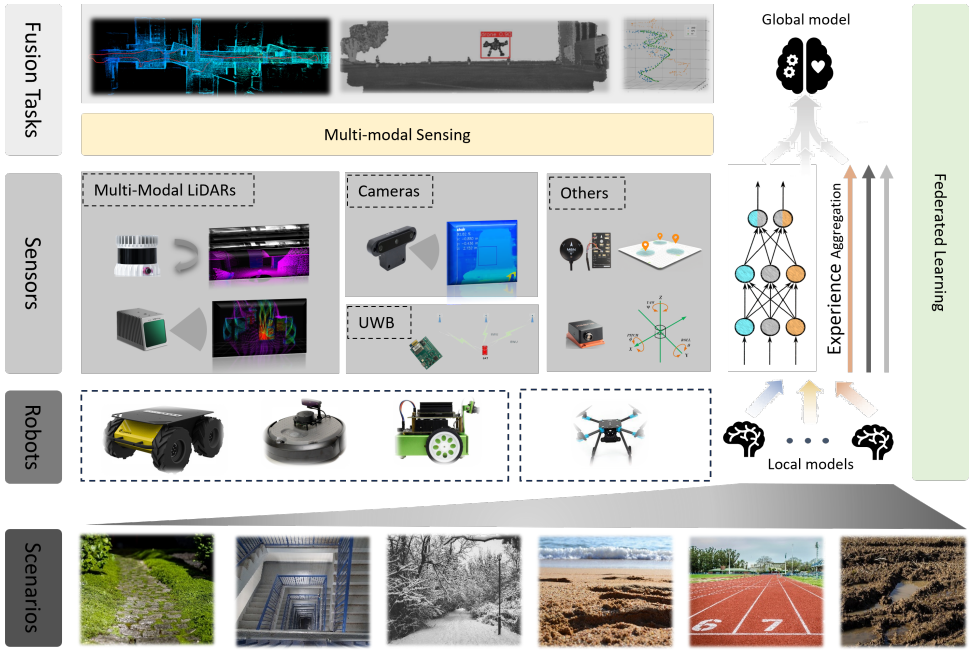


Figure 4: The conceptual diagram of this dissertation illustrates the integration of various robot platforms and sensors employed in the research under Federated Learning framework for various perception tasks. All the robot platforms and sensors depicted are those utilized in this dissertation.

The resolution of these itemized problems is essential for advancing the capabilities of autonomous robotic systems, enabling them to operate with greater autonomy, efficiency, and safety in a variety of settings.

1.2 Objectives

The research goals presented here involve not only practical engineering applications but also theoretical research into the possibilities and limitations of multi-model sensing and perception for a collaborative multi-robot system.

A conceptual overview is shown in Figure 4 while more detailed tasks and there relations explained in Figure 5. It is worth noting the robot platforms, sensors, and perception task are all involved in the real-world research in this thesis. Additionally, the majority of the experiments conducted in this dissertation were performed with real robots in real-world environments.

The study we propose effectively addresses:

- **Design and development of a collaborative and distributed robot sensing and perception framework**

The design and development of a framework for FL in a collaborative multi-robot system. The framework should work for heterogeneous sensors and robots in various scenarios. In particular, we will incorporate techniques from the edge computing domain for long-term online FL at the edge.

- **The exploration and fusion of multi-modal sensing technologies**
  The ability to accommodate not only visual sensors but also deal with multi-modal sensor fusion including LiDARs, UWB, IMUs, and other sensors for high level robot perception or situational awareness.

- **The application of the framework mentioned above for DL-based multi-modal perception in multi-robot systems**
  Towards this end, we will not only combine the state of the art in these two fields but also focus our research efforts on the optimization of the knowledge-sharing framework for single- and multi-robot systems. And, finally,

- **Deployment in real-world robots**
  The development and deployment of techniques for securing communication and providing efficient distributed computation toward real-robot deployments of certain robotic perception applications.

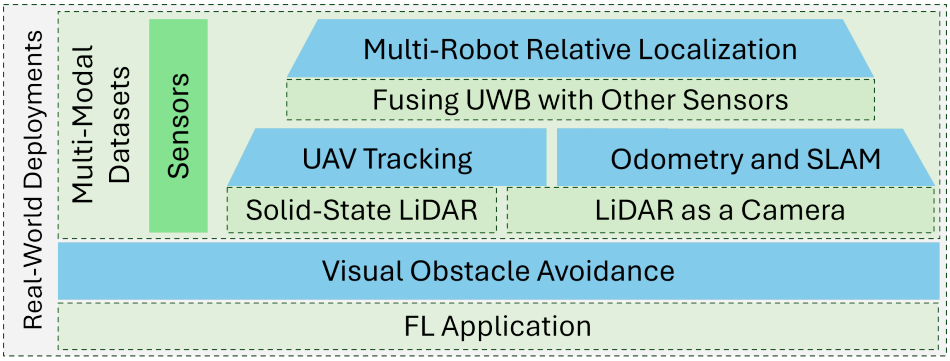


Figure 5: Real world research tasks (light blue box) within different aspects (light green box with dash lines) using various sensors (green box) covered in this dissertation.

1.3 Significance and Contribution

To address the issues mentioned, this dissertation encompasses various research aspects, illustrated in Figure 5, in a comprehensive manner. Initially, we present a multi-modal dataset collected and SLAM benchmarking for diverse purposes. Subsequently, we conduct research studies on both the dataset and real-world deploy-

ment scenarios. As a foundation for other robotic perception tasks, we first explore UWB-based, low-cost relative localization approaches that integrate with other sensors. Following this, we employ LiDAR as a camera and utilize solid-state LiDAR for UAV tracking and localization. Finally, we demonstrate the application of FL in visual obstacle avoidance, exemplifying the effectiveness of privacy-preserving learning among multiple robots.

The primary contributions of this dissertation can be summarized as follows, with detailed descriptions provided in the corresponding chapters and sections as indicated:

1. Multi-modal sensor datasets accompanied by benchmarks of contemporary state-of-the-art methodologies as a reference for the research community, as well as our independent study. The datasets collected in this dissertation include multi-modal LiDAR dataset, LiDAR based UAV tracking dataset, and UWB based multi-robot localization dataset. (in Chapter 3 on page 32)

   (a) The multi-modal LiDAR dataset includes various types of LiDARs, such as spinning LiDARs, solid-state LiDARs, and LiDAR cameras. The data collection occurred in various scenarios, with ground truth provided for all sequences. Benchmarks of state-of-the-art SLAM algorithms compare odometry accuracy, memory usage, and computational resources. (in Section 3.1 on page 32)

   (b) The LiDAR UAV tracking dataset includes LiDAR data from different types of UAVs using different types of LiDARs including a spinning LiDAR, two different solid-state LiDARs, and a RGB-D camera. (in Section 3.2 on page 51)

   (c) The UWB dataset is designed for infrastructure-free multi-robot relative localization. This dataset includes data from four moving robots, each equipped with a UWB transceiver. In addition to UWB sensor data, the dataset provides odometry data and MOCAP data to serve as ground truth. (in Section 3.3 on page 61)

2. UWB based infrastructure-free multi-robot relative localization. (in Chapter 4 on page 64)

   (a) Evaluating how UWB-based relative localization can improve the positioning of UAVs when supported by ground robots comparing the accuracy of the GNSS, UWB, and VIO approach to localization with field tests in an urban environment. (in Section 4.1 on page 64)

   (b) A novel and computationally efficient particle filter-based relative localization method fusing odometry, cooperative spatial detection information from stereo cameras, and ranging measurements, effective even with

only a single range input from UWB. The demonstration of a practical multi-robot deployment utilizing ROS2 and Zenoh, a zero overhead network protocol. The seamless integration of Zenoh addresses challenges associated with data flooding in the network, ensuring efficient communication even with numerous robots. (in Section 4.2 on page 71)

3. Leveraging LiDAR generated images (LiDAR as a camera) for diverse robotic perception applications. (in Chapter 5 on page 87)

   (a) The analysis of the performance of various DL-based visual perception models on LiDAR-generated image data is conducted. We assess the viability of applying object detection and instance segmentation models to low-resolution, 360° images. (in Section 5.1 on page 88)

   (b) Keypoint detectors and descriptor based on LiDAR generated images for the purpose of improving point cloud matching with LiDAR odometry. This research investigates the effectiveness of existing keypoint detectors and descriptors on LiDAR-generated images using various specialized metrics for quantitative evaluation. Based on the above evaluation, this work proposes a novel approach that leverages detected keypoints and their neighbors to downsample and extract a reliable point cloud, aimed at improving point cloud registration while reducing computational overhead and minimizing deficiencies in valuable point acquisition. (in Section 5.2 on page 96)

   (c) A UAV tracking approach based on the integration of images and 3D point clouds generated by a LiDAR. The UAV can be detected in LiDAR generated images instead of manually giving its initial position as it is needed in other point-cloud-only approaches. (in Section 5.3 on page 114)

4. Solid-state LiDAR based UAV tracking. (in Chapter 6 on page 124)
   We introduce a novel adaptive LiDAR scan integration method to enhance object recognition and tracking accuracy from 3D point clouds, specifically for UAV tracking. This includes a multi-modal tracking system that processes point clouds with varying integration times for improved accuracy and persistent tracking. Additionally, the method features an algorithm that dynamically adjusts the LiDAR frame integration time based on UAV speed and distance, integrating consecutive scans in a sliding window manner. A dual tracking approach employing a Kalman filter variant combines the two scan integration frequencies into a single state estimation using inverse covariance intersection. The tracking performance is evaluated using ground truth data from a motion capture system.

5. FL enhanced visual obstacle avoidance as an example showing how FL can keep the efficiency of robotic perceptions while ensuring the data privacy. (in Chapter 7 on page 150)

   (a) A review to provide a comprehensive view of how FL can be leveraged to raise the level of autonomy and degree of intelligence of robotic systems.

   (b) The design, implementation, deployment, and evaluation of a vision-based DL approach to obstacle avoidance in mobile robots within heterogeneous simulated and real scenarios, utilizing highly photorealistic and physically accurate virtual environments to study sim-to-real (Sim2Real) transfer. This includes investigating federated and continuous learning within hybrid teams of simulated and real robotic agents. Two deep obstacle avoidance neural networks were evaluated with synthetic and real-world data, comparing FL-based knowledge sharing to centralized training. Results indicate that FL methods outperform centralized data aggregation methods, with validations conducted in both simulated and real-world environments.

   (c) We integrate LiDAR-based navigation for automated labelled data gathering. We implemented an online FL-based visual obstacle avoidance system both in a simulator and real-world environment. With such a system, we can continuously collect data from obstacles and free paths and train the model while the robots operate other tasks.

1.4 Structure of the Dissertation

This dissertation is organized as follows.

- Chapter 2 reviews the advancements in multi-modal sensors and their applications in robotics and autonomous systems. Additionally, this chapter explores the integration of FL in these domains.

- Chapter 3 details the collection of the multi-modal dataset, highlighting its novel aspects with benchmarks included.

- Chapter 4 focuses on UWB based infrastructure-free multi-robot relative localization, paving the way for the other perception tasks.

- Chapter 5 explores the possibility of applying computer vison DL models within LiDAR as a camera technology and utilize this for the purpose of LiDAR odometry and UAV tracking.

- Chapter 6 introduces the novel UAV tracking algorithm leveraging the unique feature of solid-state LiDARs by dynamic integration of LiDAR scans.

Table 1: Sections Corresponding to the List of Original Publications

Chapter	Section	The number in List of Original Publication
Chapter 2	Section 2.1.2	II
	Section 2.5	I
Chapter 3	Section 3.1	VIII
	Section 3.2	XII
	Section 3.3	-
Chapter 4	Section 4.1	III
	Section 4.2	-
Chapter 5	Section 5.1	VI
	Section 5.2	VII
	Section 5.3	X
Chapter 6	Section 6.1	IX
	Section 6.2	XIII
Chapter 7	Section 7.1	V
	Section 7.2	IV

- Chapter 7 discusses the application of FL for visual obstacle avoidance in multiple robots, demonstrating FL as a framework for knowledge sharing among robotic systems.
- Chapter 8 concludes the dissertation and outlines the future work.

The work presented here integrates multiple original publications, each contributing to different aspects of the overarching research theme. These publications are essential components of the thesis and are referenced throughout the document. The chapters corresponding to these publications are presented in Table 1.

2 Research Background

This chapter explores the fundamental aspects and recent development of multi-modal sensor technologies and FL, which form the foundation of this dissertation. It provides a critical analysis of existing related research, evaluating the progress in the field while also identifying existing research gaps and opportunities.

Apart from benchmarking existing SLAM and LO technologies, this dissertation also explores UWB-based multi-robot relative localization. We introduce the role of UWB in the recent robotics field, focusing specifically on its application in multi-robot localization. Following this, we discuss UAV tracking using LiDAR technology and review the current state of research on the use of LiDAR as a camera. Finally, we examine the application of FL in robotics.

2.1 Multi-Modal Sensors in Robotics

2.1.1 3D LiDARs

Recent Advancement of LiDARs

The advancement of LiDAR technology has revolutionized various industries by providing high-resolution, accurate, and real-time data about the environment. Initially used in atmospheric studies, LiDAR has expanded its applications to autonomous vehicles, topographic mapping, forestry, and urban planning. The latest developments in LiDAR technology have led to significant improvements in range, resolution, and data processing speed. Modern LiDAR systems can generate millions of data points per second, offering detailed three-dimensional models of terrains and structures. Integration with AI and ML algorithms has further enhanced the capabilities of LiDAR, enabling more efficient data analysis and decision-making processes. This rapid evolution of LiDAR technology continues to open new frontiers in research, environmental monitoring, and smart city development, making it an indispensable tool in the advancement of science and industry.

High-resolution multi-beam spinning LiDARs have enabled high degrees of situational awareness in mobile robotic solutions. However, the larger number of vertical channels in a mechanically spinning LiDAR has meant until recently a cost too high for wide adoption in mobile platforms. While costs have significantly reduced in recent years as the technology matured, solid-state LiDARs capable of generating

high-density point clouds have emerged as an alternative. With a closer resemblance to visual sensors with limited FoV and dense scene scanning, solid-state LiDARs offer high performance and relatively lower cost [6; 7]. These sensors also offer non-repetitive scanning patterns allowing for higher resolution outputs than spinning LiDARs [8]. Despite these benefits, their unique features challenge traditional LiDAR-based odometry, localization and mapping methods owing to the limited FoV, the irregular scanning patterns, the non-repetitive sensing and the need for different techniques to overcome motion-induced distortions. If the existing algorithms can be adapted to better support new types of LiDAR sensors, solid-state LiDAR have the potential to significantly benefit perception pipelines including object detection and tracking and scene understanding in different scenarios. We see particular potential in tasks in unstructured environments where geometric features are often too few and sparse compared to urban structured environment. While there are a large number of LiDAR datasets recorded with spinning LiDAR in diverse environments, there is a lack of multi-modal datasets enabling further adoption and understanding of different types of LiDARs and data processing algorithms.

Existing Datasets

Datasets showcasing 3D LiDAR data and enabling benchmarking of approaches have had a significant impact within the research efforts in robust LiDAR odometry, localization and mapping algorithms. They have been particularly impactful within self-driving. One of the pioneers and perhaps the most significant dataset to date is arguably the KITTI benchmark [26]. The KITTI dataset includes a 64-beam 3D laser scanner, four gray-scale and color cameras, and a GPS/IMU navigation system within a single data-gathering platform. The KITTI benchmark has become an essential tool to evaluate the performance of algorithms in multiple tasks such as odometry, SLAM, object detection, or tracking, among others, in both academia and industry. Several similar datasets have also been published with a system composed of multiple cameras and spinning LiDARs, providing images and the corresponding point clouds in urban environment. Some relevant examples include the Oxford Robocar dataset [27], nuScences [28], or the EU long-term dataset [29]. Multiple spinning LiDARs are often employed in these data collecting platform, albeit mostly sharing the same sensing modality or technology.

In addition to the myriad of datasets captured in urban road environments and focused towards research in autonomous driving, the literature also showcases efforts in off-road environments. For example, the NCLT dataset provides a large-scale indoor and outdoor dataset with multi-modal sensors, including spinning LiDARs, cameras and IMU attached on a wheeled robot [30]. In another work, a handheld device comprised of one spinning LiDAR and depth camera was utilized to collect data from urban outdoor and vegetated environments [31]. With a larger variety of

environments. a multi-sensor SLAM benchmark has been presented in [32], with data captured in both indoor and outdoor environments. In relation to these works, we provide a wider variety of sensor data as well as more accurate ground truth in a selection of sequences.

There is also a number of datasets available in unstructured environments. For instance, the Robot Unstructured Ground Driving (RUGD) dataset captured from a small, unmanned mobile robot traversing in unstructured environments has been introduced in [33]. The RUGD dataset contains different terrain types focusing on visual perception tasks like semantic segmentation. Several similar datasets in unstructured environments have been presented for tasks such as scene depth prediction [34], terrain roughness understanding [35], off-road pedestrian detection [36]. Compared to these datasets, we provide MOCAP-based ground truth in a forest environment, while also including a wider variety of sensors.

In general, the number of publicly available datasets with solid-state LiDAR data is scarce. Among them, the PandaSet collects driving scenarios in urban environments with data from a forward-facing solid-state LiDAR and a 64-channels spinning LiDAR [37]. Additionally, Lin *et al.* presented an outdoor and indoor dataset with a solid-state LiDAR in college environment to test a novel LiDAR odometry and mapping (LOAM) algorithm tailored to solid-state LiDAR sensors [6]. In the presented dataset, we provide a significantly higher number of sensors as well as ground truth both indoors and outdoors.

3D LiDAR SLAM

The primary types of 3D LiDAR SLAM algorithms today are LiDAR-only [38], and loosely-coupled [39] or tightly-coupled [40] with IMU data. Tightly-coupled approaches integrate the LiDAR and IMU data at an early stage, in opposition to SLAM methods that loosely fuse the LiDAR and IMU outputs towards the end of their respective processing pipelines.

In terms of LiDAR-only methods, an early work by Zhang et al. on LOAM introduced a method that can achieve low-drift and low-computational complexity already in 2014 [41]. Since then, there have been multiple variations of LOAM that enhance its performance. By incorporating a ground point segmentation and a loop closure module, LeGO-LOAM is more lightweight with the same accuracy but improved computational expense and lower long-term drift [42]. However, LiDAR-only approaches are mainly limited by high susceptibility to featureless landscapes [43; 44]. By incorporating IMU data into the state estimation pipeline, SLAM systems naturally become more precise and flexible.

In LIOM [40], the authors proposed a novel tightly-coupled approach with LiDAR-IMU fusion based on graph optimization which outperformed the state-of-the-art LiDAR-only and loosely coupled methods. Owing to the better performance

of tightly-coupled approaches, subsequent studies have focused on this direction. Another practical tightly-coupled method is Fast-LIO [45], which provides computational efficiency and robustness by fusing the feature points with IMU data through an iterated extended Kalman filter. By extending FAST-LIO, FAST-LIO2 [46] integrated a dynamic structure ikd-tree to the system that allows for the incremental map update at every step, addressing computational scalability issues while inheriting the tightly-coupled fusion framework from FAST-LIO. Among the rest of the tightly-coupled frameworks, LIO-SAM [47] stands out by smoothing and mapping, i.e. de-skews point clouds by IMU and sub-match at a local scale instead of global scale.

The vast majority of these algorithms function well with spinning LiDARs. Nonetheless, new approaches are in demand since new sensors such as solid-state Livox LiDARs have emerged novel sensing modalities, smaller FoVs and irregular samplings have emerged [48]. Multiple existing studies using enhanced SLAM algorithms are being researched to fit these new LiDAR characteristics. Loam livox [49] is a robust and real-time LOAM algorithm for these types of LiDARs. LiLi-OM [50] is another tightly-coupled method that jointly minimizes the cost derived from LiDAR and IMU measurements for both solid-state LiDARs and conventional LiDARs.

It is worth mentioning that there are other studies addressing LiDAR odometry and mapping by fusing not only IMU but also visual information or other ranging data for more robust and accurate state estimation [51; 52].

SLAM with solid-state LiDAR

One of the key limitations of LiDAR technology preventing more widespread adoption for localization and mapping in mobile robots is the high cost of the sensors, specially compared to vision sensors. However, with lower-cost models becoming available, mainly solid-state LiDARs, multiple research efforts have been directed towards optimizing existing algorithms for the new sensing modalities and scanning patterns.

A robust, real-time LOAM algorithm for solid-state LiDAR with small FoV and irregular samplings has been presented in [6] to address several fundamental challenges arising from solid-state LiDARs. In another work, Lin *et al.* proposed a decentralized framework for SLAM tasks with multiple solid-state LiDARs to increase the FoV and improve overall system robustness [53]. Inspired by local Bundle Adjustment (BA) techniques utilized in visual SLAM, a BA approach with an adaptive voxelization method to search feature correspondence and solve the problem of sparse features points in three-dimensional LiDAR data was presented in [54]. More recent works have also worked towards improving LiDAR-based SLAM system robustness, with tightly-coupled LiDAR-inertial odometry and mapping schemes for

both solid-state and mechanical LiDARs presented in [7; 55]. Additionally, a camera and solid-state LiDAR fusion SLAM framework have also been proposed in [56].

In summary, the current research in the adaptation and tuning of algorithms for new LiDAR sensors lacks the support of a dataset for benchmarking and comparing the different approaches. Moreover, the lack of a truly heterogeneous and multi-modal dataset with various types of LiDAR sensors is preventing further comparisons between the methods to advance towards general-purpose LiDAR-based SLAM algorithms. To bridge these gaps, we focus on providing a dataset that can serve as an initial benchmark for odometry, localization and mapping in diverse environments and with different types of LiDAR sensors. We also hope that this dataset will further motivate research in the fusion of LiDAR data from different types of sensors.

SLAM benchmarks

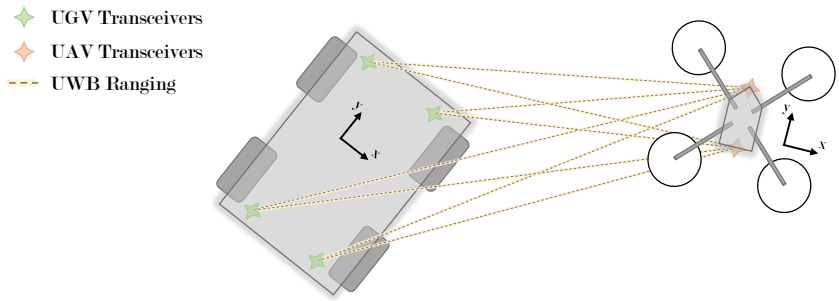
There are various multi-sensor datasets available online. We had a systematic comparison of the popular datasets in our prior work [48]. Among these datasets, not all of them have an analytical benchmark of 3D LiDAR SLAM based on multi-modality LiDARs. KITTI benchmark [26] is the most significant one with capabilities of evaluating several tasks including, for example, odometry, SLAM, object detection, and tracking.

2.1.2 UWB in Robotics

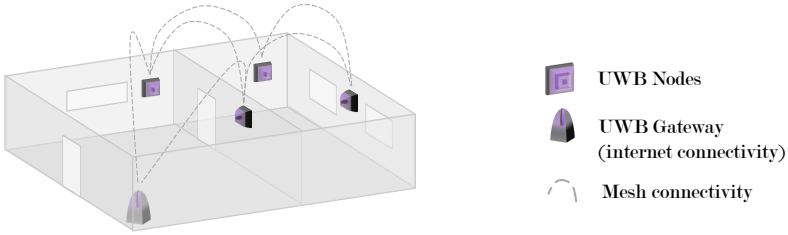
Location data is essential in multiple types of autonomous systems, whether it is from an operational perspective (i.e., to enable mobility) or from the point of view of data gathering (i.e., for aggregating data from multiple and/or mobile sources). Specifically, within the robotics domain, accurate localization methods are instrumental for autonomous robots operating in GNSS-denied environments [57]. At the same time, with the rapid development and adoption of wireless technologies in the Industrial IoT (IIoT), high-precision location services are in increasing need [58]. Ultimately, simultaneous high-bandwidth wireless data transmission and localization (situated communication) can accelerate the adoption and ubiquity of distributed autonomous systems [59; 60].

Wireless ranging technologies have significant benefits as a localization system solution owing to their higher degree of independence to environmental conditions when compared to, e.g., visual sensors [61]. Among these, ultra-wideband (UWB) wireless technology can provide significantly higher ranging accuracy than Wi-Fi or Bluetooth, among other active radio solutions [62; 63]. Multipath resilience and time resolution are additional advantages [64]. Moreover, state-of-the-art UWB-based positioning systems can achieve, out-of-the-box, localization accuracy and stability levels sufficient for the operation of complex autonomous robots such as micro-aerial

(a) Traditional positioning system based on fixed UWB anchors in known locations.



(b) Collaborative localization approach based on UWB ranging measurements from multiple transceivers in different robots.



(c) UWB Wireless sensor networks can be used for adding location metadata in the IIoT.

Figure 6: Illustration of different use cases for UWB systems.

vehicles [65], at a fraction of the price of more traditional solutions such as MOCAP systems.

Figure 6 illustrates three example use-cases of UWB wireless technology. Positioning systems based on a series of fixed nodes in known locations (or anchors), and ranging measurements between these and mobile nodes (or tags), can be used for consistent, long-term localization of mobile robots [66]. Alternatively, applications involving multi-robot systems can leverage the use of UWB transceivers onboard different robots for full relative pose estimation [67], and to the level of swarm-wide state estimation even with a single transceiver on each robot [19]. The third example then shows a UWB mesh network enabling simultaneously a networking and positioning solution in wireless sensor networks [60].

Recent surveys and reviews on UWB technology have focused on either the signal processing aspects [68], the design of UWB antennas [69; 70], the potential communication modalities [71], or the comparison of UWB localization systems with other technologies [64]. Within the robotics field, recent reviews have focused on the different positioning modalities with little attention paid to the technology itself, or the networking possibilities [61].

Accurate localization in the robotics field is still an open problem across multiple deployment scenarios. Especially in GNSS-denied environments, owing to the different accuracy, availability, area coverage, scalability, cost, and privacy requirements [72]. UWB systems are flexible and can operate in mixed indoor-outdoor environments, enabling centimeter-level accuracy localization with either global or relative positioning [73; 74]. In this section, we focus on describing how UWB technology has been leveraged in different types of robotic systems and in various application domains.

Accurate localization in the robotics field is still an open problem across multiple deployment scenarios. Especially in GNSS-denied environments, owing to the different accuracy, availability, area coverage, scalability, cost, and privacy requirements [72]. UWB systems are flexible and can operate in mixed indoor-outdoor environments, enabling centimeter-level accuracy localization with either global or relative positioning [73; 74]. In this section, we focus on describing how UWB technology has been leveraged in different types of robotic systems and in various application domains.

UWB in Mobile and Aerial Robots

Autonomous mobile robots are already a reality across multiple industries and domains, from home cleaning to in-warehouse transportation and including rapidly emerging autonomous last-mile delivery solutions. In controlled environments where anchors can be installed, UWB systems can benefit autonomous operation aiding in long-term autonomy, persistent localization, and pose initialization, among others.

Moreover, it enhances well-established onboard odometry approaches (e.g., LiDAR- or visual-based) for SLAM, with the external UWB positioning system reducing the drift and accumulated error [75; 76]. Consider e.g. extreme cases and feature-less environments such as tunnel-like environments, where LiDAR-based system will suffer from degenerate geometry. Or similarly, vision sensors in low-light or over-exposure conditions. In these scenarios, even ranging to a single fixed UWB transceiver without full external position estimation can provide significant improvements to the overall localization [77]. It is also worth mentioning that UWB positioning systems do not provide orientation when a single tag is installed on a mobile robot. With multiple tags, the mean UWB ranging error can be of the same order of magnitude as the size of the robot, thus excluding the possibility of directly calculating orientation from the position of several tags. In this direction, multiple works in the literature been devoted to investigating different methods and their accuracy to estimating orientation with multiple tags attached on single object [78; 79]. This has also been a topic of study in cooperative relative localization systems involving two or more robots [80; 81]. For example, a full 6-DoF pose estimator is presented in [82], where UWB ranging data is fused with with additional sensors (e.g., IMU).

Unprecedented advantages in the field of aerial robotics have occurred over the past decade. The first part of it was largely due to the possibilities of MOCAP systems, with advances in VIO and LiDAR-based SLAM further pushing the frontiers of the field [83]. UWB provides a competitive alternative to MOCAP systems that is accurate and stable enough by itself whenever centimeter-level accuracy suffices [65; 84; 85]. In any case, UWB ranging or localization is rarely used as a standalone system and is very often integrated with IMU data (and/or other sources of odometry or location data) to add orientation estimation and improve the oveall accuracy [86]. From the perspective of aerial systems, part of the research to date has focused on investigating the properties of ground-to-air UWB-based ranging [87; 88]. This is an important aspect to consider, as most transceivers include antennas with radiation patterns designed for largely two-dimensional spaces.

Much attention has also been put to integrating UWB within swarms of aerial robots [19; 89; 90]. In aerial swarms, external UWB positioning systems supporting multiple nodes are applicable. They can be considered as a competitive replacement, in some scenarios, to GNSS-RTK systems, with simpler deployment and initialization. In any case, most research has shifted to relative estate estimation within the swarm. Compared, e.g., to vision-based relative localization estimation, UWB systems are less dependent on environmental conditions and provide omnidirectional mutual detection.

Sensor Fusion in UWB-Based Systems

While UWB ranging enables accurate positioning, multipath interference, Non-line-of-sight (NLOS) transmission degrade the positioning performance. Therefore, UWB ranging data is often integrated and fused with different sensors and sources of location, inertial or odometry data. These include raw IMU data [86], VIO estimators [91], or LiDAR [76]. With single UWB transceivers only providing positions in 2D or 3D spaces, full pose estimation requires fusion with at least IMU. Multiple works have presented different approaches, e.g., with Extended Kalman Filters (EKF), to complete the full pose state estimation [92; 93; 94]. In outdoor environments, GNSS technology is at the core of positioning methods, with unparalleled coverage and effortless integration. However, GNSS accuracy is easily influenced by multipath interference in clustered environments and weak penetration. To achieve seamless positioning in hybrid outdoors-indoors environments, a GPS/UWB/MARG collaborative positioning system is introduced in [18].

In general terms, the most significant trend at the moment is in integrating UWB with visual sensors and, in particular, VIO estimators [95; 19; 96; 97]. In multi-robot systems involving aerial robots, vision-based sensors have typically been used for achieving relative localization and autonomous landing. In [91], a UWB-Vision fusion method is presented for autonomous docking of UAVs on mobile platforms. The authors designed a relative localization scheme by using distance measurements from sequential ranging to UWB anchors and relative displacement measurements from visual odometers. These two data streams were fed to a recursive least square algorithm to estimate the relative position to the moving target. The final docking was then done based mostly on fiducial markers using vision. From extensive indoor and outdoor experiments show the integrated UWB vision approach can be achieved from a distance up to 50m.

UWB in Swarm and Multi-Robot Systems

Collaborative multi-robot systems have been a recurrent research topics over the past decades [98]. Applications range from surveillance [99] to dense scene construction [100], and including search and rescue [101], among many others. Accurate and robust relative localization is one essential aspect in tasks such as cooperative manipulation, collaborative sensing, exploration, and transportation applications. UWB-based relative state estimation method in multi-robot systems can be divided into centralized [67; 90] and decentralized [19; 96; 102] approaches. In UWB-based decentralized relative state estimation systems, each robot performs relative state estimation individually and then exchanges such information with other robots. Compared with a centralized system where all the relative position estimation is processed on one device or based on an external system, decentralized systems have advantages

including [19; 96]: (i) flexibility, i.e., only onboard sensors and computing power are utilized, so no external off-board sensors or computing power is required; and (ii) robustness, i.e., the system can handle the lack of detection results as well as malfunctions of individual robots.

Another important consideration that is worth addressing here is signal interference in multi-robot UWB-based positioning systems. In the presence of multiple robots, signals from UWB tags can suffer from multipath transmission being reflected in other robots, in turn causing interference at the receiver. Multiple research efforts have delved into investigating the impact of MUI (Multi-User Interference) [103] and mitigating MUI using a coherent receiver in [104].

UWB in Human-Robot Interaction

Within another important research area in robotics, UWB systems have also been employed for human-robot interaction. With the objective of accurately and precisely capturing and recognizing human motion, different systems have been focusing, e.g., in in hand gestures and body movement [105], or processing electromyography (EMG) signals [106]. In a human motion capture system, as has been previously introduced for robots, global positioning from fixed UWB anchors can also be ported to mobile tags attached to the human body. Other sensors such as IMU can then aggregate acceleration and orientation data [105; 107].

Trends and Open Research Questions

This section discusses the main research trends in terms of applicability of UWB connectivity and ranging to autonomous systems, and the most important research directions regarding the scalability and security of such systems.

Within the robotics field, there is a growing trend in integrating UWB ranging and localization systems within well-established state estimation and autonomy stacks [95]. In general, fixed UWB systems based on anchor nodes placed in known locations are able to provide a competitive alternative to MOCAP systems or GNSS-RTK systems. While anchor-based systems are indeed becoming increasingly ubiquitous, we see the main research trends being directed towards relative estate estimation in systems comprising multiple robots. Finally, there is also significant traction in the area of exploiting UWB connectivity and not just its ranging capabilities, enabling situated communication in distributed autonomous systems.

From the perspective of scalability, there has been significant traction in recent years building towards the design of methods for more scalable ranging. This is increasing the number of UWB transceivers that can be located in a given area in real-time. At a lower level, this is being done by increasing the concurrency or transmissions and exploiting interference [108; 109]. At an application level, col-

laborative localization approaches in distributed and multi-robot systems waive the need for fixed anchors. At the same time, by fusing UWB data with other sensor data, they enable higher accuracy and consistent positioning between UWB ranging estimations [19; 100; 96; 95].

In terms of enhancing the security of UWB networks and positioning systems, there is a clear need for systems that are both *secure* and *scalable* [110]. This is of paramount importance for a technology that is being applied within safety-critical industrial settings [111; 112; 113], or in autonomous robots requiring high-degrees of reliability [65].

Within the IIoT, we see the most relevant research being directed towards the design, development, and deployment of dependable wireless sensor networks in harsh operational environments. These vary from underground mines to wireless networking aboard spacecraft. Furthermore, UWB positioning systems are providing a competitive solution for asset and personnel tracking, and they can be integrated into different types of intelligent industrial systems ranging from autonomous robots in logistic warehouses to complex machinery in production floors.

2.2 UWB-based Multi-Robot Relative Localization

2.2.1 UWB for Cooperative Positioning

The majority of UWB positioning systems are based on ranging between a mobile node or transceiver and a set of fixed nodes in known locations, or anchors. This is the case of commercial, out-of-the-box systems as well as most UWB-based localization methods in the literature [65]. We are however more interested in infrastructure-free relative localization methods where all UWB transceivers are potentially mounted on mobile robots. This approach has the benefit of being more flexible from an ad-hoc deployment perspective. In multi-robot systems, infrastructure-free localization can also significantly facilitate the positioning transition for robots from indoor to outdoor scenarios. The authors in [114] adopted a Monte Carlo localization approach to compute the relative localization between two aerial robots. This involved attaching one UWB tag to a mobile robot and placing three UWB anchors on another robot in a stationary position.

More conventional approaches include multilateration with least squares estimators [17] and different EKF approaches depending on the sensor data being fused [52]. Alternatively, ML approaches, including LSTM networks, have also been proposed [115]. Some of the state-of-the-art works fuse UWB with other sensors and estimators (IMU, LO, or VIO) through sliding window optimization methods [15; 19]. However, to the best of our knowledge, current methods achieving high accuracy still require other estimators (e.g., LiDAR or visual odometry) or need a higher number of ranges to be measured (either for each pair of robots or with a

higher number of robots).

2.2.2 UWB Ranging Error Mitigation

Despite UWB ranges outperforming other wireless technologies, common measurement errors such as multipath interference, electrical interference, and thermal noise persist. Accurately addressing these errors in ranging measurements is crucial for accurate state estimation. Several studies have adopted ML or deep neural network methods to model these errors [116; 117; 118]. In [117], an algorithm combining ML with the time resolution capabilities of UWB, along with adaptive physical settings was proposed. This approach enables the automatic calibration of anchor positions, effectively reducing ranging error.

Alternatively, Wang et al. [116] propose a semi-supervised support vector machine (SVM) to identify NLOS conditions and mitigate ranging errors by 10%. Furthermore, Fontaine et al. [118] demonstrated in their work that autoencoders could achieve a decent estimation accuracy.

2.2.3 LSTM Networks in UWB Positioning System

LSTM networks are particularly well suited for analyzing time-series data, making them an ideal choice for processing UWB ranging measurements. While previous studies have primarily focused on anchor-based UWB positioning systems, recent research has explored the application of LSTM in various contexts.

Poulose et al. [115] directly applied LSTM to estimate user positions with anchors, achieving competitive accuracy compared to conventional trilateration methods in simulated environments. In another study by Wang et al. [119], LSTM was employed to estimate fixed anchor-based UWB ranges using time of arrival (TDoA) information, leveraging prior UWB measurements exclusively. Moreover, Kim et al. [14] utilized LSTM to classify channel conditions based on the channel impulse response of received UWB signals. This approach effectively mitigated positioning degradation stemming from NLOS situations, leading to improved localization accuracy, particularly when coupled with EKF-based methods.

2.3 LiDAR based UAV Tracking

2.3.1 UAV Tracking with LiDARs

While LiDAR systems are often employed for detecting and tracking objects, they pose unique challenges in detecting and tracking UAVs due to their small size, varied shapes and materials, high speed, and unpredictable movements.

When deployed from a ground robot, a crucial parameter is relative localization

between different devices. Li et al. [85] suggest a new approach for tracking UAVs using LiDAR point clouds. They take into account the UAV speed and distance to adjust the LiDAR frame integration time, which affects the density and size of the point cloud to be processed.

By conducting a probabilistic analysis of detection and ensuring proper setup, as shown in [120], it is possible to achieve detection using fewer LiDAR beams, while performing continuous tracking only on a small number of hits. The limitations in the 3D LiDAR technology can be overcome by moving the sensor to increase the field of view and improve the coverage ratio. Additionally, combining a segmentation approach and a simple object model while leveraging temporal information in [121] has been shown to reduce parametrization effort and generalize to different settings.

Another approach, departing from the typical sequence of track-after-detect, is to leverage motion information by searching for minor 3D details in the 360° LiDAR scans of the scene. If these clues persist in consecutive scans, the probability of detecting a UAV increases. Furthermore, analyzing the trajectory of the tracked object enables the classification of UAVs and non-UAV objects by identifying typical movement patterns [122; 123].

2.3.2 Applications of UAV Tracking

Recently, researchers have shown interest in tracking and detecting UAVs due to two primary reasons: the rising demand for identifying and detecting foreign objects or drones in areas with controlled airspace, like airports [124; 125], and the potential for optimizing the utilization of UAVs as versatile mobile sensing platforms through tracking and detection [101].

The ability to track UAVs from UGVs allows for miniaturization and greater flexibility in multi-robot systems, reducing the need for high-accuracy onboard localization. This was demonstrated in the DARPA Subterranean challenge [126; 127], where UAVs were dynamically deployed from UGVs in GNSS-denied environments. Localization and collaborative sensing were key challenges, with reports indicating that LiDAR-based tracking was useful in domains where VIO has limitations, such as low-visibility situations [100; 128].

Similarly, tracking UAVs is crucial in the landing phase of the aerial system. Different methods using a ground-based stereo camera [129] or having the UAV carry an infrared camera to detect signals from the destination [130] have been proposed. As these works employ cameras as their main sensory system, they can be easily affected by background lighting conditions while in our approach we prefer a LiDAR which is more resilient in these environmental conditions.

2.4 LiDAR as a Camera

The literature on the processing of low-resolution LiDAR-based images is scarce. In [131], Ouster's CEO introduces the technology, showcasing the performance of the car and road segmentation using a re-trained DL model with a video. The author also comments on the potential for using this data as input to a pre-trained network from DeTone et al.'s SuperPoint project for odometry estimations. However, in both cases, the code is not available, and neither are quantitative results shown. Minimal research has been carried out in this direction. In [11], Tsiourva et al. analyzed the potential of the same Ouster LiDAR sensors that we study for saliency detection. This work already demonstrates more consistent performance and data quality in adverse environments (e.g., rainy weather). We further analyze DL-based perception performance beyond essential computer vision preprocessing such as saliency detection. A relevant recent work in the literature is [12], where the authors present a novel dataset of LiDAR-generated images with the same LiDAR-as-a-camera sensor that we use in this dissertation. The work in [12] shows the potential of these images as they remain almost invariant across seasonal changes and environmental conditions. For example, unpaved roads can be perceived in very similar ways in summer weather, snow cover, or light rain. Therefore, there is a clear advantage of these images over standard RGB or even infrared cameras, despite the limited vertical resolution.

Within the realm of robotics, some studies over the years have delved into the utilization of LiDAR-based images. But before exploring specific applications, it is vital to know the process by which range images, signal images are generated from point cloud, as detailed in [132; 133]. And it's also essential to understand the effectiveness of LiDAR-based images, through an extensive evaluation in the article [134], showing that LiDAR-based images have remarkable resilience to seasonal and environmental variations.

Perception emerges as the indisputable first step to use LiDAR within robotics. Apart from the aforementioned application, in the research[135], Sier et al. explored using LiDAR-as-a-camera sensors to track Unmanned Aerial Vehicles (UAVs) in GNSS-denied environments, fusing LiDAR-generated images and point clouds for real-time accuracy. The work [136] explores the potential of general-purpose deep learning perception algorithms, specifically detection and segmentation neural networks, based on LiDAR-generated images. The study provides both a qualitative and quantitative analysis of the performance of a variety of neural network architectures, proving that the DL models built for visual camera images also offer significant advantages when applied to LiDAR-generated images.

Delving deeper into subsequent applications, e.g., localization, research in [137], explores the problem of localizing mobile robots and autonomous vehicles within a large-scale outdoor environment map, by leveraging range images produced by 3D

LiDAR.

2.5 Federated Learning in Robotics

2.5.1 Introduction to Federated Learning

With a staggering increase in the number of connected devices being deployed worldwide within the IoT, the amount of data that is generated and transmitted has grown at exponential rates. The inefficiency of processing all this data in a centralized manner at the cloud has brought forward new computing and networking paradigms in recent years [21]. Computing at the edge, closed to where the data sources are, has evident benefits in terms of latency and bandwidth savings [22]. Another key advantage is the inherent benefits to data privacy, as raw data does not travel too far [23]. At the same time, the data is being fed to increasingly complex AI models, with DL in particular becoming pervasive across multiple fields and application domains. Recent years have also brought an increasing awareness to the risks and drawbacks of sharing personal data over the internet. The solution to distributed computing at the edge while preserving privacy of data and leveraging DL solutions is federated learning [24]. FL enables distributed training of complex models over isolated data islands from remote nodes (data sources). The local training results (updates to local models) are then aggregated, e.g. in a cloud server, and a global generalized model is shared back to the nodes. All this with zero raw data transmission [25].

From the perspective of robotic and autonomous systems, which are becoming increasingly ubiquitous, cloud solutions have enabled higher degrees of intelligence by eliminating constraints of onboard computational and storage resources [138]. Cloud robotics and AI robotics are now an essential part of state-of-the-art robotic systems. Furthermore, as mobile connectivity evolves, 5G and beyond networks are set to further bring the integration of AI, robotics and distributed networking solutions [139]. Applications of AI in robotics include, e.g., the deployment of DL for natural language processing (NLP) [140], computer vision [141], or in navigation and mapping [142]. In control, Reinforcement learning (RL) has been successfully applied in complex games [143] and its relevance for dexterous manipulation extensively demonstrated [144]. Deep reinforcement learning (DRL) is particularly relevant to autonomous robots [145].

FL provides a framework for more efficient learning in distributed autonomous systems and multi-robot systems, enabling collaborative learning across heterogeneous cloud and edge nodes and a wide range of robotic and autonomous systems. Even though cloud robotics can promote scalability, collaborative knowledge sharing, and development operation of robotics and autonomous systems, challenges remain in, e.g., security or reliable connectivity.

Multiple reviews and survey papers in the literature [146; 147] have been devoted

to studying design approaches, implementation details and application possibilities of FL. Compared to current works focused on security and privacy [148], personalized FL [149], or communication at the edge [150], the presented work aims to provide a comprehensive view of how FL can be leveraged to raise the level of autonomy and degree of intelligence of robotic systems. We look at different application opportunities at the edge and within autonomous mobile robots. We provide an overview of the most important concepts, and pay particular attention to synergies between FL and distributed ledger technologies (DLTs), among which blockchain technology has gained significant attention. A conceptual illustration of FL applications and approaches to connectivity is shown in Figure 7.

Owing to the flourish of the Internet of things (IoT) in recent years, voluminous amounts of data have been generated including sensitive and private data and a large number of edge devices have been connected by networks. As an important part of IoT, robotic and autonomous systems have been deployed in a largely growing number [151]. Cloud robotic and autonomous systems have made the limitation of onboard resources such as computation and storage no longer exist via data sharing and multi-robot cooperation through networks [138]. Thanks to the flexible provisioning supported by 5G [139], cloud infrastructure as a service provides great convenience for design small and decoupled components such as multi robots using off-the-shelf tooling.

As a result of the increasing complexity, scalability, and diversity of robotic and autonomous systems, robot learning has become a promising tool for coping with these situations [152]. DL is increasingly implemented across robotic and autonomous systems [153; 145]. Applications include the deployment of DL for NLP [140], computer vision [141], navigation and mapping [142], or in control algorithms. In the latter area, RL has been successfully applied in complex games [143] and its relevance for dexterous manipulation extensively demonstrated [144]. DRL is particularly relevant to autonomous robots [145]. DRL has been utilized in different robot fields such as search and rescue robots for exploring unknown cluttered environments [154], robot control and manipulation [155], and navigation [156].

An increasingly large number of edge devices and computation resources has made distributed learning, distributed DL in particular become a necessity of utilizing computation platforms for model training process [157]. Distributed learning made the robot learning across heterogeneous devices of cloud and edge robotic and autonomous systems feasible in a relatively secure manner.

Even though cloud robotics can promote the scalability, collaborative knowledge sharing, and development operation of robotics and autonomous systems, it is still facing challenges from security, connectivity, and work distribution ability. FL, as a novel and promising framework for robot learning, involves the local models training process over isolated data islands from remote devices and aggregates the local models in the cloud server to be a global generalized model without data transmis-

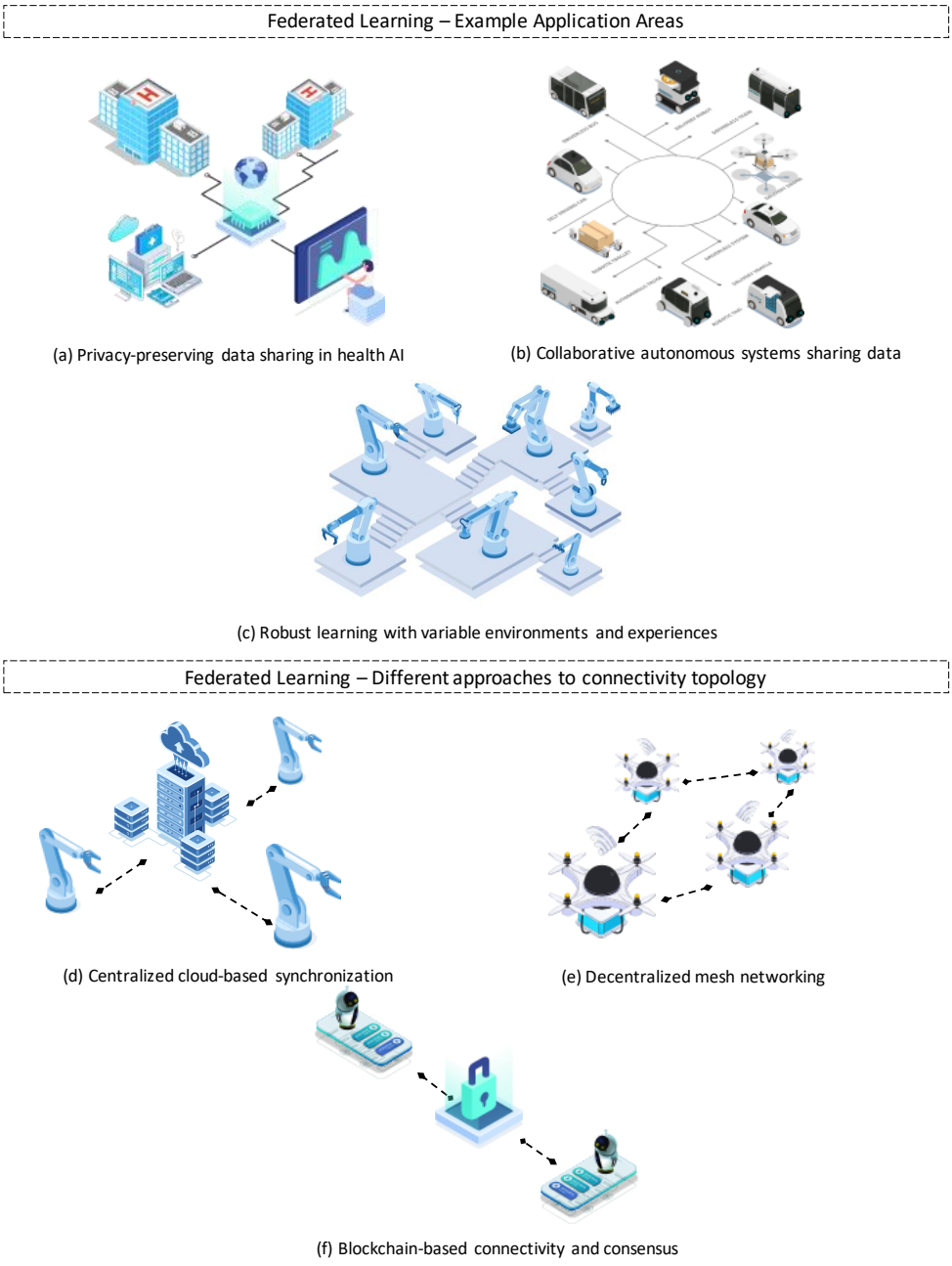


Figure 7: Conceptual illustration showing potential application areas and connectivity topologies in federated learning systems.

sion [25]. It is regarded as a secure and privacy-preserving method for robot learning which significantly meets the demands of current cloud and edge-based robotic and autonomous systems. In particular, with the blossoming of communication technologies such as 5g and 6g, FL integrated with 6g will be more potential for large-scale implementation of data-driven machine learning in our daily life [25]. Furthermore, in terms of security and privacy, FL has proved to be efficient in some data-sensitive scenarios including covid-19 related x-ray image data [158; 159; 160].

2.5.2 Federated Learning in Robotics

The adoption and development of FL frameworks have been directly or indirectly influenced by other technological and paradigm trends in robotics and autonomous systems. Since the invention of FL, there are a number of works on optimization of FL itself. Different research directions include increasing the adaptiveness, enhancing the privacy-preserving properties, or building towards more efficient collaboration for distributed robot learning, among others. In this section, we briefly introduce the different identifiable research directions (can be seen in Figure 8) from the literature, and the concepts that underpin the popularity of FL in robotics and autonomous systems.

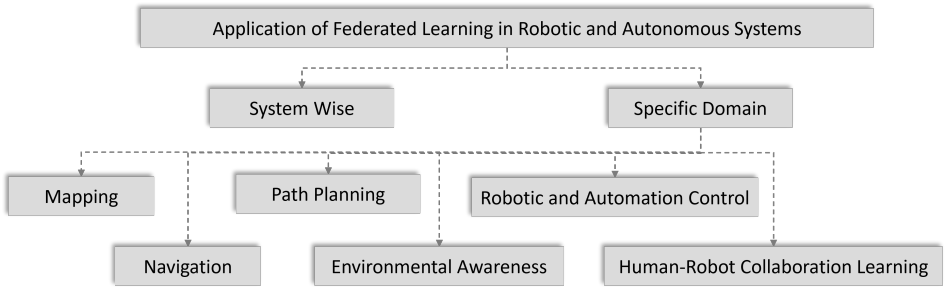


Figure 8: Applications of Federated Learning in Robotic and Autonomous Systems

Cloud Robotics and Automation

Cloud robotics is a field of robotics that capitalizes on cloud technologies. The cloud infrastructure can provide robots and autonomous systems with extensive resources and potential benefits including big data, cloud computing, collective robot learning, and human learning [138]. Under cloud infrastructures, robotic systems have access to more collaborative approaches to autonomy, faster processing of deep learning models, and more powerful computational capabilities in general. A collection of robots in different areas or states can cooperate in a variety of tasks such as disaster management identifying several critical challenges [161] and manufacturing environ-

ment [162]. There are a number of examples and implementations of cloud robotics platforms such as AWS RobotMaker [163] from Amazon, Dex-Net [164] from UC Berkeley Automation Lab, and Google cloud robotics [165].

Distributed Deep Learning

With the increasing amount of data and complexity of DL models, the process of training models becomes inherently costly, computation-intensive, and time-consuming. Distributed DL was proposed to utilize the multiple processors to accelerate DL training process by parallel the computation and the data [166; 157; 167]. There is a significant amount of work in the literature dedicated to distributed DL in the pursuit of closer collaboration between cloud and edge computing [168; 169; 170]. This balance between the two paradigms is set to become increasingly pervasive with a well-established IoT era. Immediate concerns that raise with the deployment of distributed DL across cloud and edge is the security of data and privacy of users. In consequence, multiple research directions have emerged to make distributed learning processes more scalable, secure and privacy-preserving through [171; 172; 173; 174; 175]. Additionally, other research efforts are directed towards utilizing distributed DL for processing and learning from sensitive data such as health data [176], video surveillance data [177] and medical data from multiple private or public institutions [178].

Privacy and Security in Deep Learning and Federated Learning

With the wider adoption of DL over the past decade, issues regarding data security and privacy of data sources became increasingly studied. Some of the main types of security-related issues in DL appear with evasion attacks during model inference and poisoning attack during model training [179]. Adversarial attacks to the algorithms, and model reconstruction attacks are other examples. Multiple solutions have been proposed to deal with these and other attack vectors, including differential privacy, homomorphic encryption, data anonymization, pseudonymization, algorithm encryption, or hardware security implementations, among others [180]. Despite the efforts, new attack vectors have appeared such as re-identification attacks (identification of individual data sources despite data anonymization techniques based on other information in the datasets), dataset reconstruction attacks, or tracing attacks (also referred to as membership inference, though which the inclusion of a specific individual in a dataset is inferred). While FL itself offers privacy-preserving attributes, the security robustness depends largely on the implementation and deployment methodologies. A recent survey on the topic [148] presents a comprehensive study on the current security and privacy concerning aspects with the conclusion that fewer privacy-specific threats than security-specific ones exist. Among these

are, e.g., communication bottlenecks, poisoning, and backdoor attacks, especially inference-based ones.

Recent Trends in Federated Learning

Federated learning has arguably raised the possibilities for collaborative learning across multiple independent agents. In this section, we give an overview of works that have focused towards improving specific aspects of FL.

With a focus on scalability, a high-level designed FL system based on TensorFlow has been developed that draws significant conclusions on existing challenges and future research directions [146]. From the perspective of system security, a Byzantine-robust FL proposed in [181] shows different approaches to secure FL systems and make them more robust against local model poisoning attacks. A similar approach in [182], instead considers a solution to detect the malicious model updates in every round of training process before aggregating the locals models in the centralized cloud server. Owing to a wide range of approaches relying on a the centralized cloud server for aggregation of local model updates, FL frameworks may fail if a malicious aggregation server takes over the central FL node. To cope with this problem, dispersed FL [183] has been proposed, where a global model is yielded in either a centralized or distributed manner through the aggregation of sub-global models, which are iteratively computed based on different groups similar to traditional FL approaches.

Machine learning itself can also play a role in improving the performance of FL systems. In [184], deep reinforcement learning is used to select the optimized edge nodes and the learned model parameters are integrated into a blockchain-based FL scheme for enhanced security and reliability. Furthermore, combining with other privacy-preserving machine learning methods such as differential privacy [185] and modern cryptography techniques such as homomorphic encryption [186], FL can achieve high level privacy-preserving and secure capabilities.

It is also worth meaning at this point that FL solutions are specialized in aggregating local models to a global model for knowledge sharing. Nonetheless, in terms of the characterization of heterogeneous data collected across large-scale deployments of edge devices, it is often essential to the application to make the models discriminative in each device. In this direction, personalized FL was proposed to tackle the aforementioned problem by further performing a series of learning steps locally after receiving the global model from the cloud server, based mostly on locally available data for which the model needs to be tailored [187; 188].

3 Multi-Modal Sensor Datasets

3.1 Multi-Modal LiDAR Dataset

One of the main limitations of existing datasets that we aim to overcome is the lack of availability of data from solid-state LiDARs. These LiDARs often present limited FoV, owing to the lack of mechanical rotation, and different scanning modalities. In addition, compared to spinning LiDARs, non-repetitive patterns have emerged in different products, as well as different patterns even when they are repetitive. The limited FoV together with the different scan patterns pose significant differences for many of the standard LiDAR odometry, localization and mapping algorithms. As a result, we believe that more data is necessary to advance research towards general-purpose and sensor-agnostic LiDAR data processing algorithms. To bridge this gap, we present this novel multi-LiDAR dataset with spinning LiDARs of three different resolutions, two solid-state LiDARs with different FoVs and scan pattern, and a small LiDAR-camera.

There are a number of datasets available that are relevant to this work, mostly gathered within the autonomous driving community. In Table 2, we compare the most relevant related datasets from the literature with ours. In the rest of this section, we address the key differences between the proposed dataset and existing ones.

The main contributions of this work and the presented dataset are the following:

1. A dataset with data from 5 different LiDAR sensors and one LiDAR camera in a variety of environments.
   To the best of our knowledge, this dataset is the most diverse dataset in terms of LiDAR sensors for these environments. It includes spinning LiDARs with 16 channels (Velodyne VLP-16), 64 channels (Ouster OS1-64), and 128 channels (Ouster OS1-128), each with varying vertical FoVs. Additionally, the dataset comprises two different solid-state LiDARs (Livox Horizon and Livox Avia), each with distinct scanning patterns and FoVs. Furthermore, a LiDAR camera (RealSense L515) is included, providing RGB images and LiDAR-aided depth images. The dataset is completed by low-resolution images with depth, near-infrared, and laser reflectivity data from the Ouster sensors. These are illustrated in Figure 9a.

2. Ground truth data is provided for all sequences.
   The ground truth are provided with different ways including MOCAP-based

Table 2: Comparison of related datasets with ours.

Dataset	Year	Environment	Ground Truth	LiDARs	Other
KITTI [26]	2013	Urban road	RTK_GPS/INS	3D-Velodyne HDL-64E @10 Hz	4× cameras , accel/gyro
NCLT [30]	2017	Urban Indoor Outdoor	GPS/INS	3D-Velodyne HDL-64E@10 Hz 2× 2D-Hokuyo @10/40 Hz	camera
Oxford RobotCar [27]	2017	Urban Road	GPS/INS	2× 2D-SICK @50 Hz 3D-SICK @12.5Hz	4 Camera; accel/gyro
RUGB Dataset [33]	2019	Unstructured outdoor	-	3D-Velodyne HDL-32E @10 Hz	GPS&IMU ; 3× cameras
nuScences [28]	2020	Urban Road	-	3D-32-Beams Lidar @20 Hz	6x Camera (RGB);GPS&IMU; 5x Radar@13Hz
Newer Colleage [31]	2020	Urban outdoor Vegetated	6DOF ICP	3D-Ouster-64 @10 Hz	D435i (Infrared); accel/gyro
PandaSet [37]	2021	Urban road	-	3D-Hesai-Pandar64 @10 Hz 3D solid-state lidar@10 Hz	6x Cameras. GNSS&IMU
M2DGR [32]	2022	Urban In/Outdoors	Laser 3D tracker RTK_GPS/INS	3D VLP-32C @10 Hz	3 Cameras. GNSS&IMU
Our Dataset	2022	Urban indoor Urban road Forest	6DOF MoCAP SLAM	3x 3D-Spinning lidar(16,64,128) @10 Hz 2x 3D-Solid-State-lidar @10 Hz LiDAR-Camera @30 Hz	2x accel/gyro @200 Hz 2x accel/gyro @100 Hz

ground truth in both indoors and outdoors environments (This is, to the best of our knowledge, the first LiDAR dataset to provide such accurate ground truth in forest environments in addition to indoor areas, albeit the limited trajectory length (see samples in Figure 9b)), GNSS/RTK in outdoors, and a ground truth trajectory generation method for environments where MOCAP or GNSS/RTK are unavailable.

3. Benchmarking of multiple state-of-the-art filter-based and optimization-based SLAM methods on our proposed dataset in terms of the accuracy of odometry, memory and computing resource consumption. The results indicate the limitations of current SLAM algorithms and potential future research directions.

3.1.1 System Overview

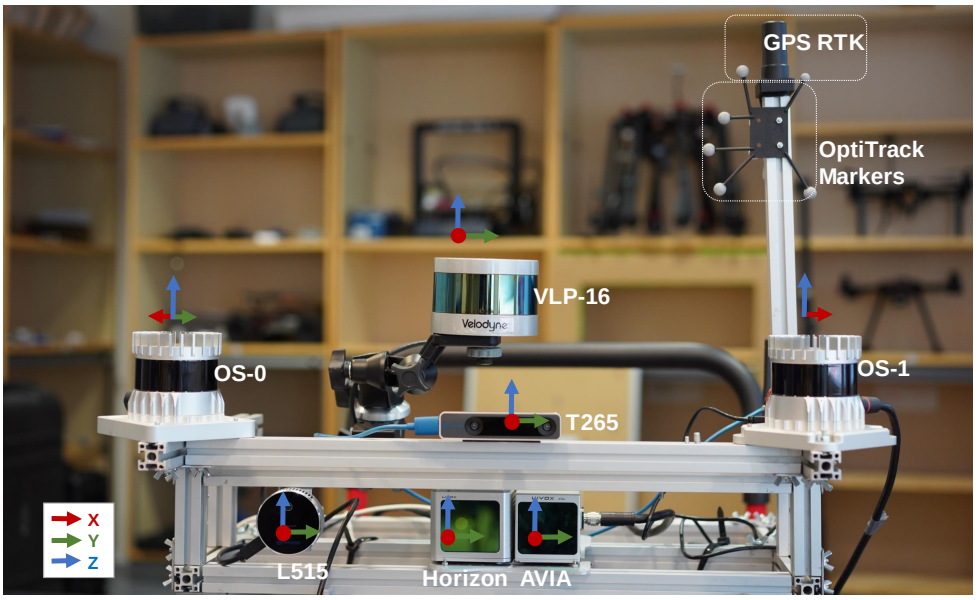
Sensory Devices

Our data collection platform is shown in Figure 9a, and details of sensors are listed in Table 3. Owing to the variety of environments where the platform has been used, it has been mounted on different types of mobile platforms. In road-like environments and large indoor halls, a Clearpath Husky mobile robot has been used. In forests outdoors with snow, it has been handheld. In small indoor spaces, it has been mounted on a mobile wheeled platform, manually pushed.

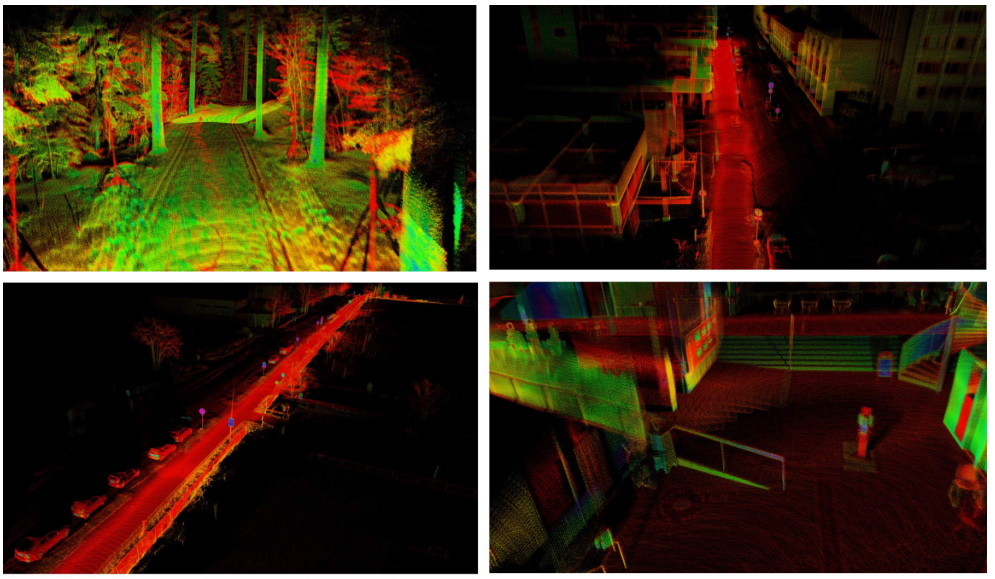
In order to increase the usability of the dataset for benchmarking general-purpose algorithms, pitch and roll rotations have been applied in different configurations when handheld, in addition to standard horizontal settings where only the yaw angles varies if the surface where it is moving is horizontal.

Approaches for Providing Ground Truth

Generating accurate ground truth data in complex environments is a challenging task, as has been identified in multiple existing datasets. Many vehicular benchmarks uti-



(a) Front view of the multi-modal data acquisition system. Next to each sensor, we show the individual coordinate frames.



(b) Samples of map data form different dataset sequences. From left to right and top to down, we display maps generated from a forest, an urban area, an open road, and a large indoor hall, respectively.

Figure 9: Multi-modal LiDAR data acquisition platform and samples from maps obtained in the different environments included in the dataset.

Table 3: Sensor specification for the presented dataset. Angular resolution is configurable in the OS1-64 (varying the vertical FoV). Livox LiDARs have a non-repetitive scan pattern that delivers higher angular resolution with longer integration times. Range is based on manufacturer information, with values corresponding to 80% Lambertian reflectivity and 100 klx sunlight, except for the L515 LiDAR camera.

	IMU	Type	Channels	FoV	Angular Resolution	Range	Freq.	Points
Velodyne VLP-16	N/A	spinning	16	360°×30°	V:2.0°, H:0.4°	100 m	10 Hz	300,000 pts/s
Ouster OS1-64	ICM-20948	spinning	64	360°×45°	V:0.7°, H:0.18°	120 m	10 Hz	1,310,720 pts/s
Ouster OS0-128	ICM-20948	spinning	128	360°×90°	V:0.7°, H:0.18°	50 m	10 Hz	2,621,440 pts/s
Livox Horizon	BOSCH BMI088	solid-state	N/A	81.7°×25.1°	N/A	260 m	10 Hz	240,000 pts/s
Livox Avia	BOSCH BMI088	solid-state	N/A	70.4°×77.2°	N/A	450 m	10 Hz	240,000 pts/s
RealSense L515	N/A	LiDAR camera	N/A	70°×55°	N/A	9 m	30 Hz	-

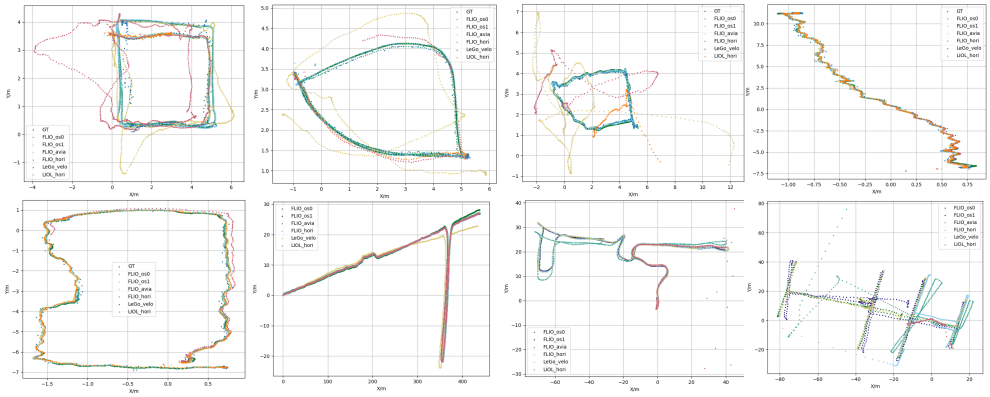


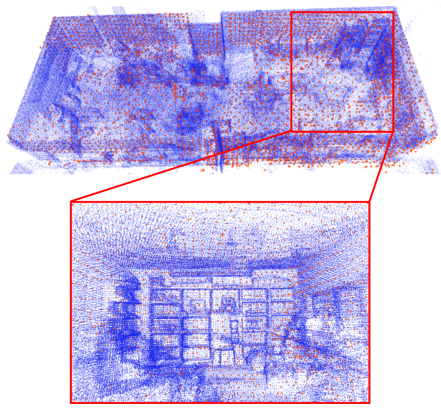
Figure 10: Estimated trajectories. Top row: *Indoor01*, *Indoor02*, *Indoor03*, *Forest02*. Bottom row: *Forest01*, *Road02*, *Indoor04*, *Indoor05*.

lize the pose generated from GNSS/INS fusion method as ground truth. However, multi-path effect can affect the accuracy of the pose estimated by GNSS sensors in forest and urban environments. For indoor environments, GNSS signals are unavailable.

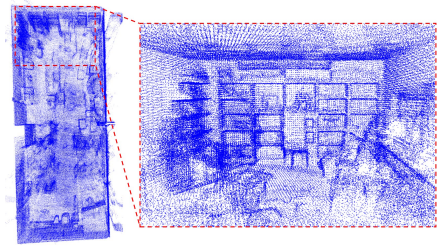
MOCAP systems have been widely adopted in indoor environments owing to their ability to provide millimeter-level accuracy in positioning data. However, the utilization of MOCAP systems is limited mainly by the range of the cameras, usually in the 10 to 20 m range. The need for relatively complex setup of the system has also prevented the adoption of such systems for outdoors environments, and specially in unstructured environment such as forests.

To meet the demands of reliable ground-truth data for diverse environments, the presented dataset includes MOCAP-based ground-truth data in both a subset of indoor and forest environments. This enables millimeter level pose estimation as ground truth for odometry algorithms in both structured and unstructured environments, which can aid in researching low-drift odometry algorithms, accurate feature

tracking, and reduction of motion-induced distortions in the data. For open space, we provide the ground truth data from GNSS/RTK. For large-scale environment, where the MOCAP system is unavailable, we also provide location information as a reference from SLAM methods [55]. In these settings, the higher-resolution LiDAR OS0-128 can be used as a baseline for the other sensors. We evaluate the SLAM algorithms in diverse environments, with a sample of environments shown in Figure 10. From the different SLAM methods further characterized in the next section, those that use data from the OS0 sensor showcase the most robust performance in a series of sampled sequences.



(a) NDT localization with ground truth map. External view and Internal view when the current laser scan (orange) is aligned with the Ground truth map (blue).



(b) Ground truth map for one of the indoor sequences generated based on the proposed approach (SLAM-assisted ICP-based prior map). This enables benchmarking of LiDAR odometry and mapping algorithms in larger environments where a motion capture system or similar is not available, with significantly higher accuracy ($< 2\,cm$) than GNSS/RTK solutions.

Figure 11: SLAM-Assisted Ground Truth Map

SLAM-assisted Ground Truth Map

MOCAP and GNSS/RTK have been used in most of the cases for ground truth. However, to provide accurate ground truth for large-scale indoor and outdoor environments, where the MOCAP system is unavailable or GNSS/RTK positioning result becomes unreliable due to the multi-path effect, we propose a SLAM-assisted solid-state LiDAR-based ground map generation framework.

Inspired by the prior map generation methods in [31], where a survey-grade 3D imaging laser scanner Leica BLK360 scanner is utilized to obtain static point clouds of the target environment, we employed a low-cost solid-state LiDAR Livox Avia and high resolution spinning LiDAR to collect undistorted point clouds from environments. According to the Livox Avia datasheet, the range accuracy of the Avia sensor is 2 cm with a maximum detection range of 480 m. Due to the non-repetitive scanning pattern, the environment coverage of the point cloud within the FoV increases with time. Therefore, we integrated multiple frames when the platform is stationary to get more detailed undistorted environmental sampling. Each integrated point cloud contains more than 240,000 points. The Livox built-in IMU is used to detect the stationary state of the platform when the acceleration values are smaller than 0.01 m/s^2 along all axes. After gathering multiple undistorted point cloud submaps from the target environment, the next step is to match and merge all submap into a global map by ICP. As the ICP process requires a good initial guess, we employ a high resolution spinning LiDAR OS0 with a 360° horizontal FoV to provide raw position by performing real-time SLAM algorithms. This process is outlined in Algorithm 1. A dense and high-definition ground truth map can be obtained by denoising the map generated by the algorithm described above to remove noise. Figure 11b shows ground truth map of sequence indoor08 generated based on Algorithm 1

Let $\mathcal{P}_{sk}$ be the point cloud produced by the spinning LiDAR, $\mathcal{P}_{dk}$ be the point cloud generated by solid-state LiDAR, and $\mathcal{I}_k$ be the IMU data from built-in IMU. Our previous work has shown high resolution spinning LiDAR has the most robust performance in diverse environments. Therefore, LeGo-LOAM [42] is performed with a high resolution spinning LiDAR (OS0-128) and outputs the estimated pose for each submap.

The cached data $\mathcal{S}_{cache}$ stores submaps and the related poses. Let $\mathcal{P}_i$ be the point cloud and related pose $\mathbf{p}_i$ in $\mathcal{S}_{cache}[i]$. The submap $\mathcal{P}_i$ will be first transformed to map coordinate as $\mathcal{P}_i^m$ based on estimated pose $\mathbf{p}_i$; then GICP methods are employed on $\mathcal{P}_i^m$ to minimize the Euclidean distance between closest points against point cloud $\mathcal{M}_{ap}$ iteratively; $\mathcal{P}_i^m$ will be transformed by the transformation matrix generated from GICP process, then merged to the map $\mathcal{M}_{ap}$. The result map $\mathcal{M}_{ap}$ is treated as ground truth map. Figure 9b provides a visual display of several ground truth maps, which have been acquired through the aforementioned procedural steps.

After the ground truth map generated, we employ normal NDT method in [189]

Algorithm 1: SLAM-assisted ICP-based prior map generation for ground truth data.

Input:

Spinning LiDAR pointcloud: $\mathcal{P}_{sk}$

Solid-state LiDAR pointcloud: $\mathcal{P}_{dk}$

IMU data: $\mathcal{I}_k$

Output:

Platform state: $\mathbf{p}_k$

Prior map: $\mathcal{M}_{ap}$

while *new* $\mathcal{P}_{sk}$ **do**
    $\mathbf{p}^k \leftarrow SLAM(\mathcal{P}_{sk})$;

// Cached still clouds and raw pose
$\mathcal{S}_{cache} = \{\}$;
// Cached still cloud
$\mathcal{P}_{cache} = []$;
while *new* $\mathcal{P}_{dk}$ **do**
    if $\mathcal{I}_k.V_{angular} < th_a$, $\boldsymbol{p}_k.V_{linear} < th_v$ **then**
        $s = True$;
        $\mathcal{P}_m = \mathcal{P}_m + \mathcal{P}_{dk}$;
    else
        $s = False$;
        $\mathcal{P}_{cache}.clear()$;
        $\mathcal{S}_{cache} \leftarrow (\mathcal{P}_m, \mathbf{p}_k)$;
while $\mathcal{S}_{cache}.size() > 0$ **do**
    $\mathcal{M}_{ap} \leftarrow ICP(\mathcal{S}_{cache}, \mathbf{p}_k, \mathcal{M}_{ap})$;
    $\mathcal{S}_{cache}.clear()$;

to match the real-time point cloud data from spinning LiDAR against the high resolution map as the Figure 11a shows to get the platform position in ground truth map. The matching result from the NDT localizer is treated as the ground truth.

3.1.2 Data Collection Setup

Data Collection Platform

The data collection platform contains various LiDAR sensors with different characteristics, from traditional spinning LiDARs with different resolutions to novel low-cost solid-state LiDAR featured with non-repetition scanning patterns. There are three spinning LiDARs: a 16-channel Velodyne LiDAR (VLP-16), a 64-channel Ouster LiDAR (OS1), and a 128-channel Ouster LiDAR (OS0). The OS0 and OS1

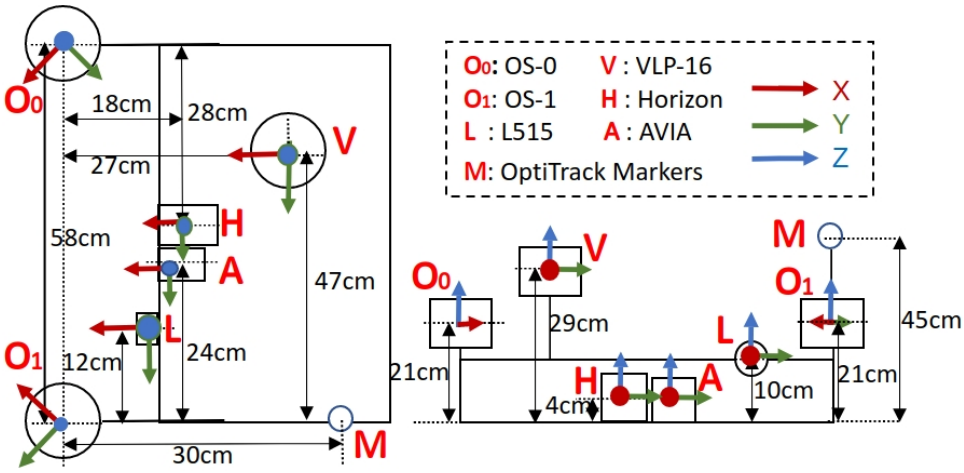


Figure 12: Our data collecting platform, top view (left) and front view (right)

sensors were mounted left and right sides, where the OS1 is turned 45° clockwise, and the OS0 is turned 45° counter clockwise. The Velodyne LiDAR is at the top-most position. The data collecting platform can be seen in Figure 9a. Additionally, please refer to the top view of Figure 12 for the detailed distances, positions and orientations.

Regarding the physical configuration, the Livox Horizon and Avia LiDARs were installed in the center of the frame facing forward. The L515 camera was attached to the front left of the platform. On the sides, the OS0 and OS1 sensors were mounted at a bit higher level, where the OS1 is turned 45° clockwise, and the OS0 is turned 45° anticlockwise. The Velodyne LiDAR is at the top-most position with the x-axis facing forward as well. The Optitrack marker set for the MOCAP-based and the antenna for GNSS/RTK ground truth are both fixed on the top of the aluminum stick to maximize its visibility and detection range.

To ensure a low-latency and high-speed transmission of all data, the LiDARs are connected to a Gigabit Ethernet router and a computer onboard the platform featuring an Intel i7-10750h processor, 64 GB of DDR4 RAM memory and 1 TB SSD storage. The Optitrack system is also physically connected via Ethernet to the onboard computer on a separate interface to the LiDARs. Finally, the RealSense L515 camera is connected using a USB 3.0 port.

Calibration and Synchronization

The extrinsic parameters of the LiDARs are calculated based on optimization methods similar to those presented in [190]. We calculate the extrinsic parameters in an indoor office environment, while the sensor platform was stationary. The coordinate

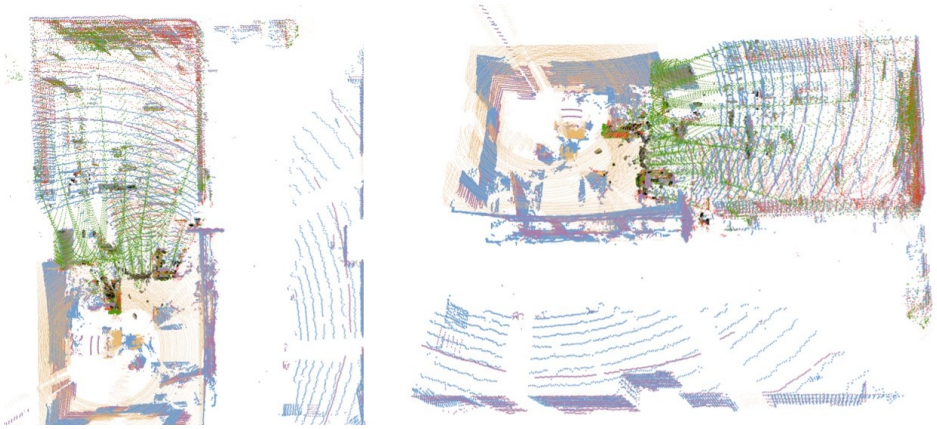


Figure 13: Top view of point cloud data generated for the calibration process with multiple LiDARs. The red and green point clouds represent data obtained from the Livox Horizon and Avia, respectively. The pruple, yellow, blue and black clouds are from the VLP-16, OS1, OS0 and L515 sensors, respectively.

system of the Horizon LiDAR sensor is treated as the reference frame during the calibration process. Ten consecutive frames of point cloud data are integrated from the solid-state LiDARs to accumulate a higher degree of detail from the environment.

The point cloud data from each different LiDAR is then transformed to the reference frame based on manual measurements of a set of features in the environment. Then, a GICP method is employed to optimize the relative transformation between the reference frame and LiDARs iteratively [191]. For reference, in Figure 13, we show sample sensor data from one of the indoor environments after calibration.

The ROS package $livox\_camera\_LiDAR\_calibration$ was utilized to calibrate the extrinsic parameter between the Horizon sensor and the L515 LiDAR camera. The intrinsic parameters of LiDARs and the LiDAR camera are given based on factory settings and manufacturer information. A specific rosbag containing raw data recorded in stationary settings at the room shown in Figure 13 is provided for end-user re-calibration and potential application of different methods.

We synchronized all LiDAR sensors in Ethernet mode via the software-based precise timestamp protocol (PTP) [192]. And we compared the orientation estimation between the sensor's built IMUs, and SLAM results with LiDARs and concluded that the latency of our system is below 5 ms.

Software Information

Our software system is based entirely on ROS Melodic under Ubuntu 18.04. The set of ROS drivers and the publishing frequency of the different sensors is shown visu-

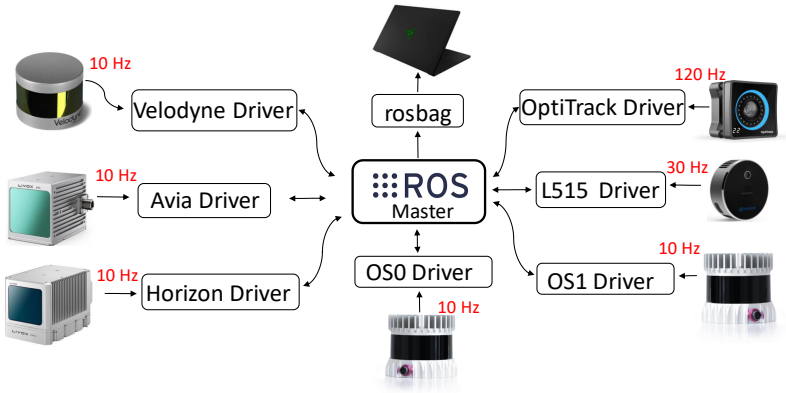


Figure 14: ROS drivers and data gathering frequency for the different LiDAR sensors used in our platform.

ally in Figure 14. Owing to the lack of hardware signals to synchronize the sensor data, as in other datasets in the literature [29], we approach the minimization of the data synchronization problem by running all the sensor drivers, data recording programs locally on a high performance computer. This, together with the networking equipment, aid in reducing the latency of data transmission at the hardware and software level (timestamped at the ROS drivers). In order to support a potentially wider use of the data, the dataset also includes the time stamp from built-in internal oscillators for both Livox and Ouster LiDARs, and for both point cloud and IMU data, in addition to the timestamp included in the header of all ROS messages. We have also compared the angular velocities of IMUs together with data from the MOCAP system to conclude that the latency of our system is less than 10 ms.

Data Sequences and Format

The different subsets of out dataset are divided into three categories based on the environment: forest, indoor, urban outdoor. Table 4 lists all the sequences in our dataset. Three sequences are provided for the forest environment. The forest data is collected at a forest in Turku, Finland $(60°28'14.3"N 22°19'54.8"E)$. The sequences *Forest01* and *Forest02* are collected in winter time with snow-covered ground. *Forest01* includes a square-shaped trajectory, while in *Forest02* the system is moved in a straight trajectory. Both of these sets include MOCAP data. A larger-scale forest recording is also provided in the *Forest03* sequence, with Horizon and VLP-16 LiDARs mounted on a smaller handheld device. These sequences can support research in areas from tree-counting to tree stem diameter estimation. The vast difference in environment structure from urban settings to forest settings can also support LiDAR-based general-purpose odometry, localization, and mapping algorithms.

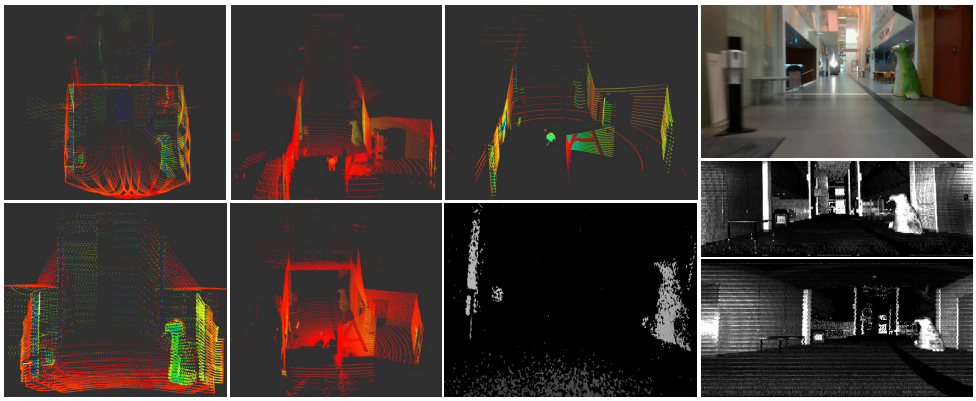


Figure 15: Our dataset was captured by a rich suite of sensors. Subsets of the data from the *Indoor04* sequence are visualized here. The leftmost column shows the LiDAR data from Avia and Horizon; the second column shows the LiDAR data from OS1 and OS0; the third column shows the data from the VLP-16 and depth image from L515. The rightmost column shows the RGB image from L515 and range images from 0S1 and OS0.

The indoor environment then adds another dimension to the dataset with five data sequences. The data is collected in rooms and open halls of ICT-City in Turku, Finland. Three sequences are collected in a large experiment room where data from the MOCAP system is available. From these, *Indoor03* contains faster rotations and sudden movements, while positioning the sensors closer to objects in front and around. In consequence, most of the solid-state LiDAR view is blocked by objects or walls, presenting a significantly more challenging situation for odometry estimation algorithms based primarily on scan matching methods. The data in *Indoor01* is recorded while maintaining a longer distance ($\approx 50\,cm$) with objects and following a square-shaped trajectory with a reduced number of rotations. The *Indoor02* sequence then features a circular trajectory with more rotation but again maintaining an even larger distance to objects than in *Indoor03*. Sequences *Indoor04* and *Indoor05* correspond to recordings in a large hall and long corridor environment, respectively.

Finally, two sequences of open-road environment around the ICT-City building in Turku, Finland, are also included in this dataset. The length of *Road01* is over 50 m, while the traversed length of the trajectory in *Road02* is about 500 m.

The data is collected in ROS and saved with the rosbag format, which has become a standard in the robotics research community. Sampled data frames from a subset of the sensors is shown in Figure 15. The detailed data format for each type included in the dataset is listed as follow:

1. **Point cloud from spinning LiDARs** from the three spinning LiDARs, namely VLP-16, OS0-128 and OS1-64. The sensor message type from spinning Li-

DARs is recorded as $sensor\_msgs :: PointCloud$. Each point in the point cloud holds four values (x, y, z, I), where x, y, z represent the local Cartesian coordinates, and I is the laser reflectance of the point measured.

2. **Point cloud from solid-state LiDAR** from the two solid-state LiDARs, namely Avia and Horizon. The message type of these solid-state LiDARs in the rosbags is Livox's custom data format named $livox\_ros\_driver/CustomMsg$. The customized message keeps the first point's timestamp of each frame as the base time and then provides an offset time relative to the base time for each point. This is needed as the non-repetitive pattern does not allowed for a posteriori estimation, unlike the spinning LiDARs, in which we can estimate the time difference between points based on the settings of the mechanical parts. With this information, the de-skew process can then be conducted on the data to compensate for the distortion in the point cloud data caused by the sensor's egomotion [6]. We have maintained this message type that contains time information for each point for algorithms that include in the processing flow the de-skew of point cloud data and other related research. However, standard ROS messages simplify the visualization of the point cloud with tools such as Rviz, and provide a format that many other LiDAR processing algorithms relying on standard ROS messages use [7]. Therefore, we provide format conversion tools to transform the Livox custom message data to the ROS standard message type $sensor\_msgs :: PointCloud$. Each point is then converted to a new one that holds five values (x, y, z, I, C), where x, y, z is the local Cartesian coordinate set, I is the intensity of the point, and where the integer part of C represents the line number and the decimal part the point timestamp.

3. **Images from LiDAR camera**. The RealSense L515 LiDAR camera is configured to publish RGB images with a size of 1920×1080, and depth images with a size of 1024×768. The message type is $sensor\_msgs :: Image$ at frequency 10 Hz. The depth estimations are aided by the built-in LiDAR sensor.

4. **Images from high-resolution spinning LiDAR**. The two high resolution LiDAR from Ouster, OS0-128 and OS1-64, can output fixed-resolution range images, near-infrared images captured by the laser sensor, and signal images. In these, each pixel represents the distance from the sensor origin to the point, the strength of the light captured, and the object's reflectivity, respectively. The images are published at frequency of 10 Hz. The image data is spatially correlated, with 16 bits per pixel and a linear photo response. The message type in the rosbags is the standard $sensor\_msgs/Image$.

5. **Inertial data from spinning and solid-state LiDARs**. There are in total four built-in 6-axis IMU sensors with 3-axis gyroscope and a 3-axis accelerometer,

one in each of the Ouster and Livox LiDARs. They publish data at a frequency of 100 Hz in the former and 200 Hz in the latter. The data type of IMU data in the rosbags is again ROS' standard $sensor\_msgs :: Imu$.

6. **Ground truth data**. The ground truth data from the MOCAP system is included in rosbags as $geometry\_msgs :: PoseStamped$ messages. They are obtained from the computer driving the set of OptiTrack cameras through a VRPN connection.

In this study, we evaluated popular 3D LiDAR SLAM algorithms in multiple data sequences of various scenarios, including indoor, outdoor, and forest environments. The indoor data is collected from the office, corridor, and halls of ICT-City, Turku, Finland. The forest data is gathered at a forest in $(6028'14.3"N, 2219'54.8"E)$, Turku, Finland. Conversely, the road dataset was assembled from data collected at an open-air skating park, also situated in Turku, Finland. Further specifications pertaining to the dataset are comprehensively elaborated in Table 4.

Table 4: List of data sequences in our dataset (V: Velodyne VLP-16, H: Livox Horizon, A: Livox Avia, O_0: Ouster OS0, O_1: Ouster OS1, L: RealSense L515, T: RealSense T265)

Sequence	Description	Ground Truth	Sensors
Forest01	Forest(Winter,Square)	MOCAP	V,H,A,O_0,O_1
Forest02	Forest(Winter,Straight)	MOCAP	V,H,A,O_0,O_1
Forest03	Forest (long path)	SLAM	V,H
Road01	Open road(short)	SLAM	V,H,A,O_0,O_1
Road02	Open road(long)	SLAM	V,H,A,O_0,O_1
Road03	Open road	GNSS/RTK	V,H,A,O_0,O_1,L,T
Indoor01	Office room(easy)	MOCAP	V,H,A,O_0,O_1
Indoor02	Office room(middle)	MOCAP	V,H,A,O_0,O_1
Indoor03	Office room(hard)	MOCAP	V,H,A,O_0,O_1
Indoor04	Large Hall	SLAM	V,H,A,O_0,O_1
Indoor05	Long Corridor	SLAM	V,H,A,O_0,O_1
Indoor06	Lab space (easy)	MOCAP	V,H,A,O_0,O_1,L,T
Indoor07	Lab space (hard)	MOCAP	V,H,A,O_0,O_1,L,T
Indoor08	Classroom space	SLAM+ICP	V,H,A,O_0,O_1,L,T
Indoor09	Corridor (short)	SLAM+ICP	V,H,A,O_0,O_1,L,T
Indoor10	Corridor (long)	SLAM+ICP	V,H,A,O_0,O_1,L,T
Indoor11	Hall (large)	SLAM+ICP	V,H,A,O_0,O_1,L,T

3.1.3 Dataset Evaluation and Benchmarking

SLAM-Assisted Ground Truth Evaluation

The evaluation of the accuracy of the proposed ground truth prior map method is challenging for some scenes in the dataset, as both GNSS and MOCAP systems are not available in indoor environments such as long corridors. To evaluate the generated ground truth, we adhere to the methodological approach described in the referenced study [31]. Figure 16 (a),(b),(c) shows the standard deviations of the ground truth generated by the proposed method during the first 10 seconds when the device is stationary from sequence *Indoors09*. The standard deviations along the X, Y, and Z axes are 2.2 cm, 4.1 cm, and 2.5 cm, respectively, or about 4.8 cm overall. However, evaluating localization performance when the device is in motion is more difficult. To better understand the order of magnitude of the accuracy, we compare the NDT-based ground truth Z values with the MOCAP-based ground truth Z values in the sequence *Indoor06* environment. The results in Figure 16 (d) show that the maximum difference does not exceed 5 cm.

LiDAR Odometry Benchmarking

Different types of SLAM algorithms are selected and tested in our experiment. LiDAR-only algorithms LeGo-LOAM (LEGO) [1] and Livox-Mapping (LVXM) [2] are applied on data from the VLP-16 and Horizon separately; Tightly-coupled iterated extended Kalman filter-based methods, FAST-LIO (FLIO) [3] [55], are applied on both spinning LiDAR and solid-state LiDAR with built-in IMUs; A tightly coupled LiDAR inertial SLAM system based on sliding window optimization, LiLi-OM [4] [7] is tested with OS1 and Horizon. Furthermore, a tightly coupled method featuring sliding window optimization developed for Horizon LiDAR, LIO-LIVOX (LIOL)[5] has also been tested on Horizon LiDAR data. When IMU is required, we use the Avia's IMU for Velodyne LiDAR results.

We provide a quantitative analysis of the odometry error based on the ground truth in Table 5. To compare the trajectories in the same coordinate, we treat the coordinate of OS0 as a reference coordinate and transformed all trajectories generated by selected SLAM methods to reference coordinate. The Absolute Pose Error (APE) [193] is employed as the core evaluation metric. We calculated the error of

[1] https://github.com/RobustFieldAutonomyLab/LeGO-LOAM

[2] https://github.com/Livox-SDK/livox_mapping

[3] https://github.com/hku-mars/FAST_LIO

[4] https://github.com/KIT-ISAS/lili-om

[5] https://github.com/Livox-SDK/LIO-Livox

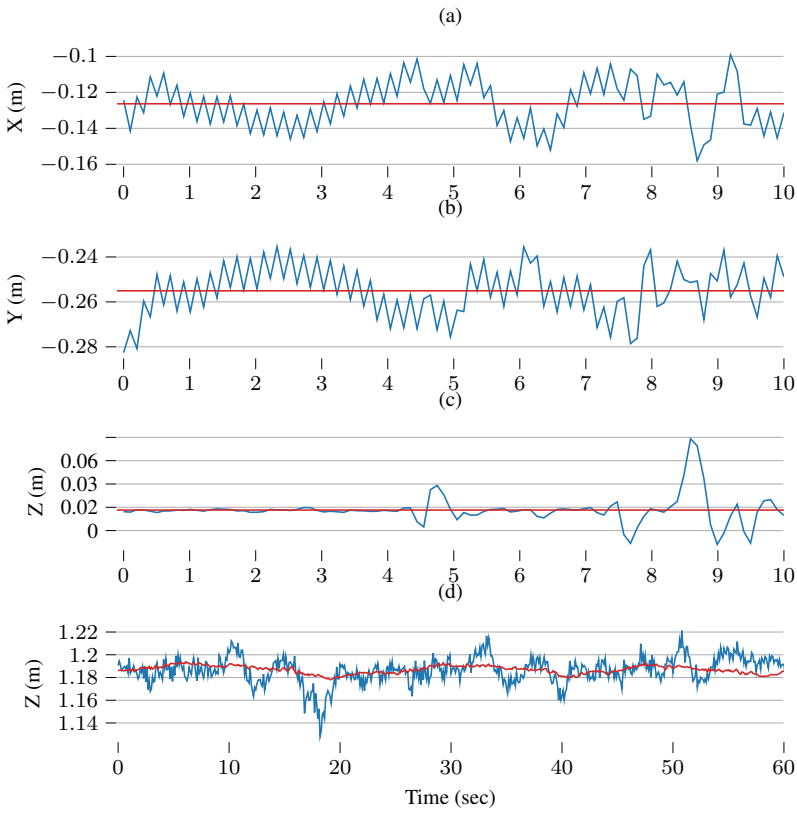


Figure 16: **(a) (b) (c):** Ground truth position values for the first 10 seconds of the dataset when the device was stationary. Red lines show the mean values over this period of time. **(d):** Comparison of NDT-based ground-truth z-values (blue) to MOCAP-based z-values (red) over the course of 60 seconds of the dataset while the device was in motion.

each trajectory with the open-source EVO toolset [6].

From the result, we can conclude that FAST_LIO with high resolution spinning LiDAR OS0 and OS1 has the most robust performance that can complete all the trajectories on different sequences with promising accuracy. Especially for sequence *Indoor09* showcasing a long corridor, all other methods failed and Fast_LIO with high resolution LiDAR remain effective.

Solid-state LiDAR-based SLAM systems such as LIOL_Hori perform as well or even better in outdoor environments than rotating LiDARs with appropriate algorithms, but perform significantly more poorly in the indoor environments. For the open road sequences *Road03*, all SLAM methods perform well, and the trajecto-

[6] https://github.com/MichaelGrupp/evo.git

Table 5: Absolute Pose Error (APE) (μ/σ) in *cm* of the selected methods (N/A when odometry estimations diverge). Best results in bold.

Sequence	FLIO_OS0	FLIO_OS1	FLIO_Velo	FLIO_Avia	FLIO_Hori	LLOM_Hori	LLOMR_OS1	LIOL_Hori	LVXM_Hori	LEGO_Velo
Indoor06	**0.015 / 0.006**	0.032 / 0.011	N/A	0.205 / 0.093	0.895 / 0.447	N/A	0.882 / 0.326	N/A	N/A	0.312 / 0.048
Indoor07	**0.022 / 0.007**	0.025 / 0.013	0.072 / 0.031	N/A	N/A	N/A	N/A	N/A	N/A	0.301/0.081
Indoor08	0.048 / 0.030	**0.042 / 0.018**	0.093 / 0.043	N/A	N/A	N/A	N/A	N/A	N/A	0.361 / 0.100
Indoor09	**0.188 / 0.099**	N/A	0.472 / 0.220	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Indoor10	0.197 / 0.072	**0.189 / 0.074**	0.698 / 0.474	0.968 / 0.685	0.322 / 0.172	1.122 / 0.404	1.713 / 0.300	0.641 / 0.469	N/A	0.930 / 0.901
Indoor11	0.584 / 0.080	**0.105 / 0.041**	0.911 / 0.565	0.196 / 0.098	0.854 / 0.916	0.1.097 / 0.0.45	1.509 / 0.379	N/A	N/A	N/A
Road03	0.123 / 0.032	**0.095 / 0.037**	1.001 / 0.512	0.211 / 0.033	0.351 / 0.043	0.603 / 0.195	N/A	0.103 / 0.058	0.706 / 0.396	0.2464 / 0.063
Forest01	0.138 / 0.054	0.146 / 0.087	N/A	0.142 / 0.074	0.125 / 0.062	0.116 / 0.053	0.218 / 0.110	**0.054 / 0.033**	0.083 / 0.041	0.064 / 0.032
Forest02	0.127 / 0.065	**0.121 / 0.069**	N/A	0.211 / 0.077	0.348 / 0.077	0.612 / 0.198	N/A	0.125 / 0.073	0.727 / 0.414	0.275 / 0.077

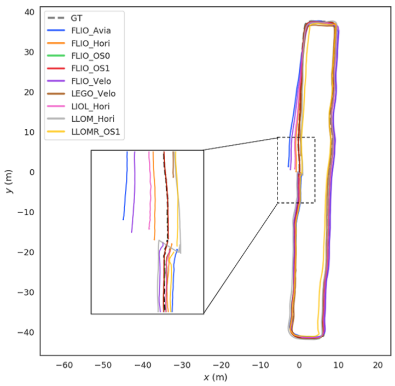
ries are completed without major disruptions. For the indoor sequence *Indoor06*, Avia-based and Horizon-based FLIO are able to reconstruct the sensor trajectory but significant drift accumulates. In all of these sequences, all the methods applied to spinning LiDARs perform satisfactorily. This result can be expected as they have full view of the environment, which has a clear geometry. For the sequence *Indoor10* showcasing a long corridor, almost all methods can reconstruct the complete trajectory again. The best performance comes from OS0-FLIO and OS1-FLIO with correct alignment between the first and last positions. We hypothesize that this occurs because OS0 has more channels than OS1, leading to lower accumulated cumulative angular drift.

In addition to the quantitative trajectory analysis, we visualize trajectories generated by selected methods in 3 representative environments in Figure 17. Within this illustration, Figure 17a signifies the trajectory within an indoor setting, Figure 17b depicts the trajectory within an open road environment, and Figure 17c demonstrates the trajectory within a forest environment. Full reconstructed paths are available in the dataset repository.

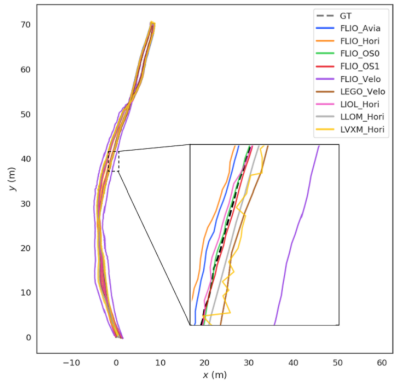
Runtime Evaluation

We conducted this experiment on 4 different platforms. The first platform (1) was a Lenovo Legion Y7000P with 16 GB RAM, a 6-core Intel i5-9300H (2.40 GHz) and an Nvidia GTX 1660Ti (1536 CUDA cores, 6 GB VRAM). The second (2) platform was the Jetson Xavier AGX, a popular computing platform for mobile robots, which has an 8-core ARMv8.2 64-bit CPU (2.25 GHz), 16 GB RAM and 512-core Volta GPU. From its 7 power modes, we chose MAX and 30 W (6 core only) modes. The third (3) platform was the Nvidia Xavier NX which is a common embedded computing platform with a 6-core ARM v8.2 64-bit CPU, 8 GB RAM, and 384-core Volta GPU with 48 Tensor cores. We chose the 15 W power mode (all 6 cores) for the NX. The fourth (4) platform was the UP Xtreme board featuring an 8-core Intel i7-8665UE (1.70 GHz) and 16 GB RAM.

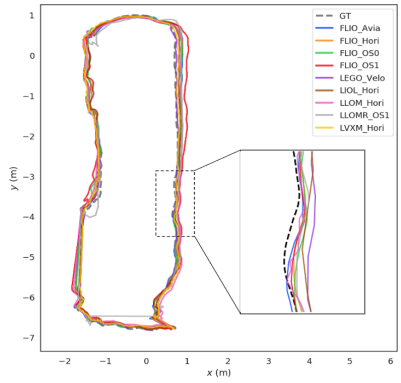
These platforms all run ROS Melodic on Ubuntu 18.04. The CPU and mem-



(a) trajectory comparison of sequence *Indoor10*



(b) trajectory comparison of sequence *Road03*



(c) trajectory comparison of sequence *Forest01*

Figure 17: Demos of trajectories generated by multiple 3D LiDAR SLAM based on data from indoor, road, and wild environments

ory utilization is measured with a ROS resource monitor tool [7]. Additionally, for minimizing the difference of the operating environment, we unified the dependencies used in each SLAM system into same version, and each hyperparameter in the SLAM system is configured with the default values. The results are shown in Table 6.

The memory utilization of each selected SLAM approach among the two processor architectures platforms is roughly equivalent. However, the CPU utilization of the same SLAM algorithm running on Intel processors is generally higher than the other algorithms, and also the highest publishing frequency is obtained. LeGO_LOAM has the lowest CPU utilization but its accuracy is towards the low end (see Table 6), and has a very low pose publishing frequency. Fast-LIO performs well, especially on em-

[7] https://github.com/alspitz/cpu_monitor

Table 6: Average run-time resource (CPU/RAM) utilization and performance (pose calculation speed) comparison of selected SLAM methods across multiple platforms. The data is played at 15 times the real speed for the pose publishing frequency. CPU utilization of 100% equals one full processor core.

	(CPU utilization (%), RAM utilization (MB), Pose publication rate (Hz))				
	Intel PC	**AGX MAX**	**AGX 30 W**	**UP Xtreme**	**NX 15 W**
FLIO_OS0	(79.4, 384.5, 74.0)	(40.9, 385.3, 13.6)	(55.1, 398.8, 13.2)	(90.9, 401.8, 47.3)	(53.7, 371.1, 14.3)
FLIO_OS1	(73.7, 437.4, 67.5)	(54.5, 397.5, 21.2)	(73.9, 409.2, 15.4)	(125.9, 416.2, 58.0)	(73.3, 360.4, 14.2)
FLIO_Velo	(69.9, 385.2, 98.6)	(44.4, 369.7, 29.1)	(58.3, 367.6, 21.4)	(110.5, 380.5, 89.6)	(57, 331.5, 19.5)
FLIO_Avia	(65.0, 423.8, 98.3)	(40.8, 391.5, 32.3)	(47.4, 413.4, 24.5)	(113.2, 401.2, 90.7)	(51.2, 344.8, 21.9)
FLIO_Hori	(65.7, 423.8, 103.7)	(37.6, 408.4, 34.7)	(50.5, 387.9, 26.8)	(109.7, 422.8, 91.0)	(47.5, 370.7, 23.4)
LLOM_Hori	(126.2, 461.6, 14.5)	(128.5, 545.4, 9.1)	(168.5, 658.5, 1.5)	(130.1, 461.1, 12.8)	(N / A)
LLOMR_OS1	(112.3, 281.5, 25.8)	(70.8, 282.3, 9.6)	(107.1, 272.2, 6.5)	(109.0, 253.5, 13.6)	(N / A)
LIOL_Hori	(186.1, 508.7, 19.1)	(247.2, 590.3, 9.6)	(188.1, 846.0, 4.1)	(298.2, 571.8, 14.0)	(239.0, 750.5, 4.54)
LVXM_Hori	(135.4, 713.7, 14.7)	(162.3, 619.0, 10.5)	(185.86, 555.81, 5.0)	(189.6, 610.4, 7.9)	(198.0, 456.7, 5.5)
LEGO_Velo	(28.7, 455.4, 9.8)	(42.4, 227.8, 7.0)	(62.8, 233.4, 3.5)	(39.7, 256.6, 9.1)	(36.9, 331.4, 3.7)

bedded computing platforms, with good accuracy, low resource utilization, and high pose publishing frequency. In contrast, LIO_LIVOX has the highest CPU utilization due to the computational complexity of the frame-to-model registration method applied to estimate the pose.

A final takeaway is in the generalization of the studied methods. Many state-of-the-art methods are only applicable to a single LiDAR modality. In addition, those that have higher flexibility (e.g., FLIO) still lack the ability to support a point cloud resulting from the fusion of both types of LiDARs.

Mapping Quality Evaluation

We qualitatively compare the mapping result generated from different LiDARs in indoor environments as shown in Figure 18.

From Figure 18, we can observe that the LIOL method applied to solid-state LiDAR presents the most detailed and clear map. It is worth noting that these maps have been generated with the default configuration of the methods and without changing parameters such as the map update frequency. This result matches the quantitative results obtained with the same sensors and algorithms in the forest.

As shown in Figure 18, Horizon-based LIOL has the best mapping ability, but

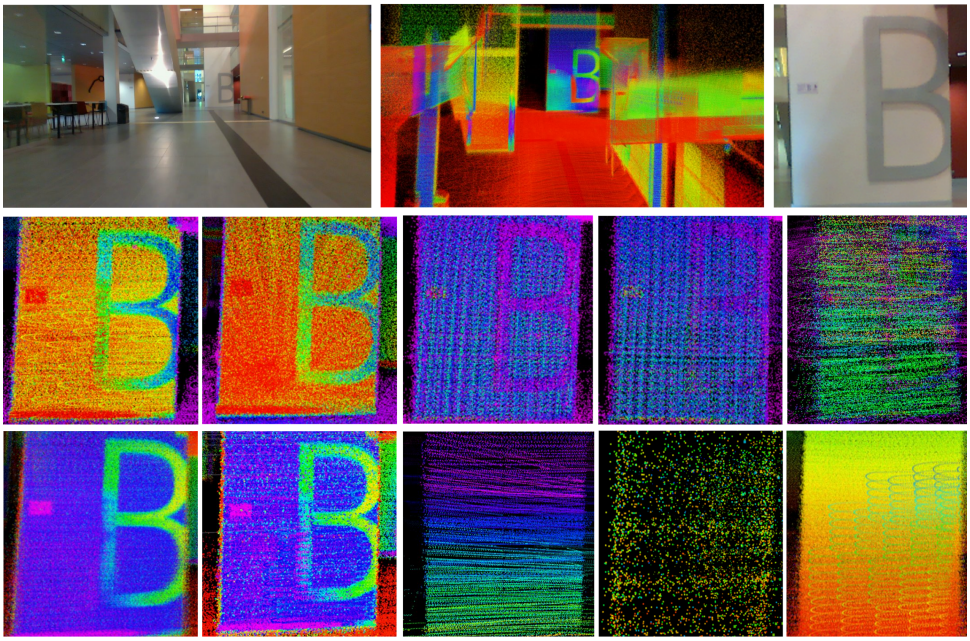


Figure 18: Qualitative comparison of the mapping quality. The first row from left to right shows RGB full view image, full view Horizon-based LIOL and close view RGB image. The second row from left to right shows OS0, OS1, Velodyne, Avia and Horizon-based FLIO. The bottom row from left to right shows the Horizon-based LIOL, Horizon, OS1-based LLOM and LLOMR, Velodyne's LeGo-LOAM maps and Horizon-based LVXM, respectively.

if the environment (such as sequence *indoors 06-09*) is complex, LIOL will fail to map due to drift. In addition, OS0 and OS1-Based FLIO also have good mapping ability, thanks to the wide FOV and excellent resolution of OS0 and OS1. Compared to OS0 and OS1, Velodyne has poorer mapping ability due to its larger resolution, and it has almost failed to reconstruct the letter B sign in Figure 18. LVMX, LLOM, and LLOMR focus on calculating the mobile platform's pose estimation rather than point cloud mapping ability, so the point cloud maps they reconstructed are relatively poor.

3.2 UAV Tracking Dataset with Multi-Modal LiDARs

The integration of UAVs into multi-robot systems underscores the critical importance of inter-robot tracking for both relative and global state estimation methods [194; 195]. Tracking UAVs from an UGV within these multi-robot configurations enhances miniaturization and flexibility while reducing the dependency on high-accuracy onboard localization systems [127; 126]. The UGV frequently serves as a base station, providing essential data to support UAV operations in areas with limited GNSS availability.

One of our primary objectives is to address the paucity of data from solid-state LiDARs. These advanced sensors, a recent innovation in long-range scanning technology, generate high-density point clouds, making them particularly well-suited for tracking objects in three-dimensional space, including UAVs [7]. Their non-repetitive scan patterns enable the production of dense point clouds with adjustable frequencies and variable FoV coverage. Recognizing the necessity for more data to advance research in the development of general-purpose, sensor-agnostic LiDAR data processing algorithms, we have initiated efforts to bridge this gap. Consequently, we introduce our novel multi-LiDAR UAV tracking dataset, which includes a spinning LiDAR, two solid-state LiDARs with distinct FoV and scan patterns, and an RGB-D camera combination. This dataset is designed to facilitate advancements in UAV tracking and support the development of robust algorithms capable of processing a diverse range of LiDAR data.

The main contributions of this work and the presented dataset are the following:

1. A dataset with data from 3 different LiDAR sensors and an RGB-D camera in both indoor and outdoor environments. To our knowledge, this presents the first diverse dataset in terms of LiDAR sensors specifically for UAV tracking. The dataset features a spinning LiDARs with 64 channels (Ouster OS1-64) , two different solid-state LiDARs with different scanning patterns and FoVs (Livox Mid-360 and Avia), and an RGB-D camera (RealSense D435). Due to the shorter range of the point cloud generated by the RGB-D camera compared to the LiDAR sensors, only RGB images were extracted from it. The dataset is further enriched with low-resolution images containing depth, near-infrared, and laser reflectivity data from the Ouster sensor. These components are illustrated in Figure 19.

2. The dataset includes sequences with MOCAP ground truth in both indoor and outdoor environments. The indoor trajectories exhibit more intricate patterns than the outdoors, while the outdoor sequences were deliberately selected to simulate potential docking and infrastructure inspection scenarios [196], with an emphasis on their proximity to a building.

3. Based on the presented dataset, we provide a baseline comparison with re-

Figure 19: Illustration of the hardware used in the experiments. At the bottom, the tracking sensors including Ouster OS1-64, Livox Mid-360, Livox Avia and Intel RealSense D435.

cent LiDAR-based UAV tracking algorithms, benchmarking the performance with different sensors, UAVs of different sizes (from MAVs to more standard commercial platforms), and algorithms.

3.2.1 Provided Ground Truth

Accurately generating ground truth data in complex environments is a challenging task, as evidenced by various existing datasets. In indoor settings, MOCAP systems have been widely adopted due to their ability to provide positioning data with millimeter-level accuracy. However, for outdoor datasets near buildings, obtaining a reliable GNSS signal is often not feasible.

To address the need for reliable ground truth data across diverse environments, our dataset uniquely includes MOCAP-based ground truth data for both indoor and outdoor scenarios. This inclusion ensures comprehensive coverage and facilitates robust evaluation in various real-world conditions.

3.2.2 Configuration of Hardware and Software

Hardware Information

The primary objective of our sensor system is to gather data from a diverse range of LiDAR sensors, each offering distinct characteristics. These sensors include two innovative, low-cost solid-state LiDARs and a 3D spinning LiDAR, complemented

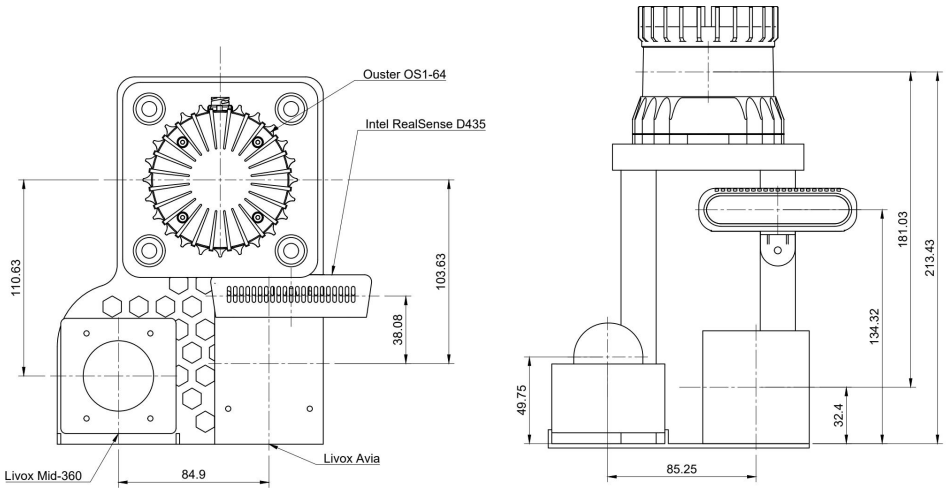


Figure 20: Data collecting platform, top view (left) and front view (right)

by an integrated RGB-D camera.

Specifically, our data collecting platform consists of a 64-channel Ouster spinning LiDAR (OS1), two Livox solid state LiDAR sensors: Mid-360, featuring a nearly 360° FoV, and Avia, with an almost-circular FoV. The setup is completed with an Intel RealSense D435 RGB-D camera. The top and front views depicted in Figure 20 provide a comprehensive understanding of the distances, positions, and orientations of these sensors.

The LiDARs are linked to a Gigabit Ethernet router, as well as to an onboard computer on the platform. This computer is equipped with an Intel i7-10750h processor, 16 GB of DDR4 RAM memory, and 1 TB SSD storage. To maintain a distinct connection from the LiDARs, the Optitrack system is physically attached to the same computer via a separate Ethernet interface. Furthermore, the RealSense D435 camera is connected to a USB 3.2 port for seamless integration and operation.

Software Information

Our software system is exclusively built upon ROS Noetic, operating on Ubuntu 20.04. The ROS drivers and the publishing frequency of the various sensors are illustrated in Figure 21. To address the absence of hardware signals for sensor data synchronization, as noted in other datasets in the literature [29], we adopt an approach aimed at minimizing the data synchronization challenge. This involves running all sensor drivers and data recording programs locally on a high-performance computer. By doing so, and in conjunction with the networking equipment, we effectively reduce data transmission latency at both the hardware and software levels. Timestamps are applied at the ROS drivers to ensure synchronization. Additionally,

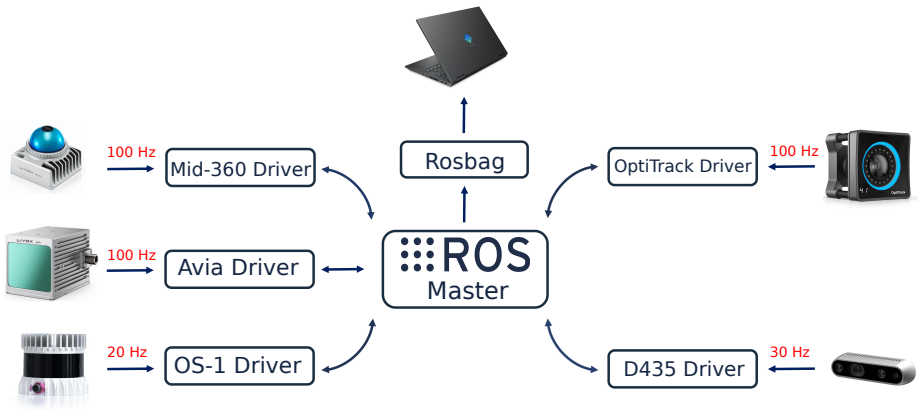


Figure 21: ROS drivers and data gathering frequency for the different LiDAR sensors used in our platform.

we maintain timestamp consistency across all sensors by utilizing the PTP, which is specifically designed for high-precision time synchronization within local networks.

Sensor Calibration

The extrinsic parameters for the LiDAR sensors were determined using optimization methods similar to those presented in Jeong et al. [190]. The calibration process was conducted in an indoor office environment with the sensor platform stationary. During calibration, the coordinate system of the Ouster LiDAR sensor was designated as the reference frame. To enhance the level of detail in the environment, ten consecutive frames of point cloud data were integrated from the solid-state LiDARs.

To align the point cloud data from each LiDAR to the reference frame, we manually measured a set of features in the environment. Subsequently, we employed the GICP method to iteratively optimize the relative transformation between the reference frame and the other LiDARs [191]. This iterative optimization process ensures accurate and precise calibration, thereby enhancing the overall performance of the LiDAR-based system.

Similarly, the extrinsic parameters between the Ouster sensor and the Intel RealSense D435 camera were determined using the depth cloud produced by the latter. Additionally, we provide supplementary stationary data for extrinsic, if a custom calibration is preferred. The intrinsic parameters of the sensors are provided based on factory settings and manufacturer information.

Dataset Sequences and Format

Our dataset is organized into three distinct categories based on the environment and trajectory structure: structured indoor, unstructured indoor, and unstructured outdoor. Each category captures specific movement patterns and characteristics, as follows:

(i) Structured Indoor: This subset (HolybroStdn) comprises simple trajectories represented by predefined, systematic patterns, including a circle, a cube, a spiral, and an up and down movement. These structured trajectories are intentionally included to provide standardized, reproducible, and easily interpretable movement patterns. By employing these basic trajectories, ablation studies can be performed on isolated and specific aspects of different methods. This naturally allows for evaluating scenarios where different elements are *decoupled*. The structured indoor trajectories act as a reference point for understanding how well a method performs under well-defined and controlled conditions.

(ii) Unstructured: In this subset, trajectories exhibit a more irregular nature, simulating movements that occur both indoors and outdoors without strict adherence to predefined patterns. These trajectories aim to capture the spontaneous and less structured nature of real-world flight scenarios, where a UAV's movements can vary significantly based on the environment and other influencing factors.

A comprehensive list of sequences in each category is shown in Table 7.

Figure 22 displays a subset of the dataset, represented in three dimensions to enhance the understanding of spatial distances in each direction. Complete visualizations of all recorded trajectories are available on the project's GitHub page.

The data collection for the indoor and outdoor trajectories was conducted in distinct locations to capture a diverse range of environments. The indoor data was gathered in the open area of our lab, providing a controlled and confined setting for UAV movements. On the other hand, the outdoor data was acquired in an open area adjacent to the building, offering a more estensive space for capturing trajectories with greater distances and varied environmental conditions. While the indoor trajectories exhibit both greater complexity and, in some, regularity, the outdoor trajectories emphasize their proximity to the building, simulating potential docking scenarios.

We qualitatively assessed the complexity of each trajectory accommodating for different UAV capabilities, classifying the trajectories into three levels: easy, medium, and difficult. This categorization is based on the size and speed of the UAV used during data collection. Specifically, we utilized three different UAVs - the Holybro, Autel II, and Tello. As expected, larger UAVs tended to result in easier

Table 7: List of data sequences in our dataset recorded indoor and outdoor

Sequence	Description	Ground Truth	Difficulty
HolybroStnd01	Structured(Up/Down)	MOCAP	Easy
HolybroStnd02	Structured(Square)	MOCAP	Easy
HolybroStnd03	Structured(Circle)	MOCAP	Easy
HolybroStnd04	Structured(Spiral)	MOCAP	Easy
Holybro01	Unstructured, Indoor	MOCAP	Easy
Holybro02	Unstructured, Indoor	MOCAP	Easy
Holybro03	Unstructured, Indoor	MOCAP	Easy
Holybro04	Unstructured, Indoor	MOCAP	Medium
Holybro05	Unstructured, Indoor	MOCAP	Medium
HolybroOut01	Unstructured, Outdoor	MOCAP	Medium
HolybroOut02	Unstructured, Outdoor	MOCAP	Medium
Autel01	Unstructured, Indoor	MOCAP	Easy
Autel02	Unstructured, Indoor	MOCAP	Easy
Autel03	Unstructured, Indoor	MOCAP	Easy
Autel04	Unstructured, Indoor	MOCAP	Medium
Autel05	Unstructured, Indoor	MOCAP	Hard
AutelOut01	Unstructured, Outdoor	MOCAP	Hard
AutelOut02	Unstructured, Outdoor	MOCAP	Hard
Tello01	Unstructured, Indoor	MOCAP	Medium
Tello02	Unstructured, Indoor	MOCAP	Medium
Tello03	Unstructured, Indoor	MOCAP	Hard
Tello04	Unstructured, Indoor	MOCAP	Hard
Tello05	Unstructured, Indoor	MOCAP	Hard
TelloOut01	Unstructured, Outdoor	MOCAP	Hard
TelloOut02	Unstructured, Outdoor	MOCAP	Hard

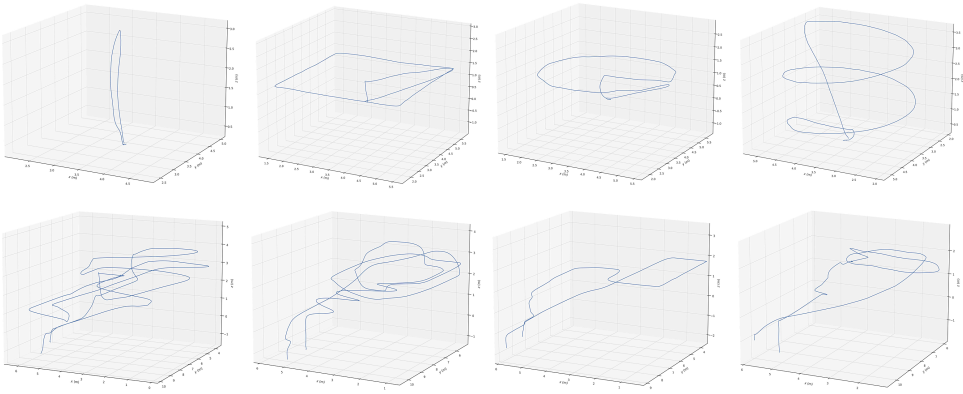


Figure 22: Sample of recorded trajectories. Top row: *HolybroStnd01*, *HolybroStnd02*, *HolybroStnd03*, *HolybroStnd04*. Bottom row: *Holybro03*, *Autel02*, *Autel05*, *Tello04*.

trajectories, while faster speeds led to higher levels of difficulty. The difficulty levels are chosen with the aim of enabling researchers to evaluate their methods across a spectrum of challenges, ranging from simple to more intricate flight paths, following similar standards in previous datasets [197].

To introduce additional challenges, we designed the fourth indoor track for each UAV as a circular path around an obstacle and the fifth indoor track as a loop between two obstacles. These obstacle tracks were specifically included to test the UAVs' maneuvering abilities in constrained spaces, further diversifying the difficulty levels and encouraging the evaluation of methods under more intricate flight conditions.

In our dataset, the indoor trajectories typically span approximately 16 m, whereas the outdoor trajectories cover larger distances, extending up to 30 m. This contrast in distances allows researchers to analyze trajectory characteristics in diverse spatial contexts, emphasizing the importance of robust analysis techniques that can adapt to varying scales and complexities.

Data collection within the ROS environment makes use of the rosbag format, which has emerged as a standard within the robotics research community. Figure 23 showcases sampled data frames from a subset of the sensors. Detailed data formats for each type of data included in the dataset are as follows:

(i) **Point cloud data from spinning LiDAR (OS1-64)** is recorded as *sensor_msgs::PointCloud*. Each point in the point cloud contains four values (x, y, z, I), representing local Cartesian coordinates (x, y, z), and the measured laser reflectance (I).

(ii) **Point cloud data from solid-state LiDARs, Avia, and Mid-360**, employs Livox's custom data format named *livox_ros_driver/CustomMsg* in the ros-

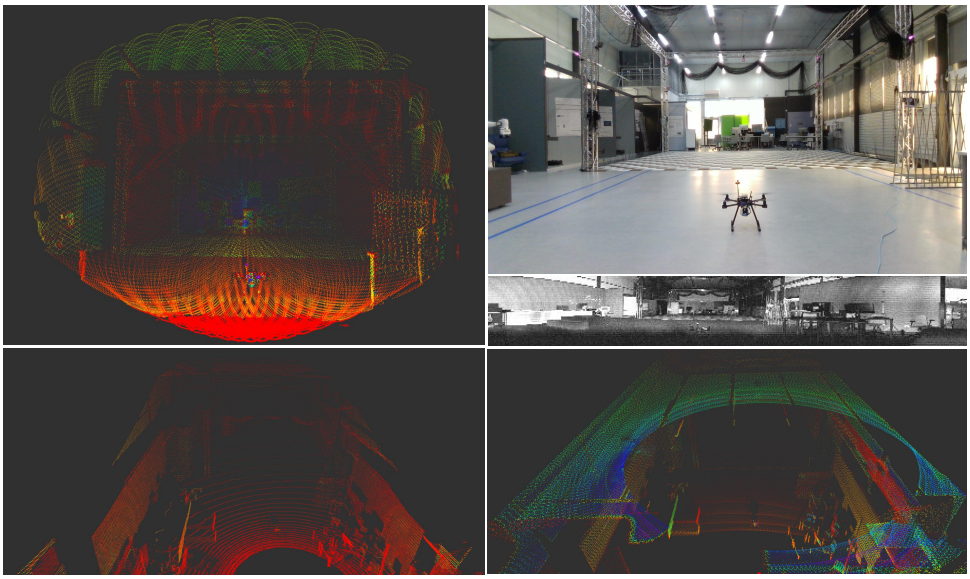


Figure 23: Subsets of data from the *Holybro01* sequence. The left column shows the LiDAR point cloud from the Avia and the OS-1 as the right column displays the point cloud from the Mid-360 as well as the RGB image from the D435 and the signal image from the OS-1.

bags. This custom message includes a base time and an offset time relative to the base time for each point. This approach compensates for the non-repetitive pattern inherent to solid-state LiDARs and allows for de-skewing of the point cloud data, addressing distortions caused by the sensor's egomotion. We have retained this message type for algorithms involving point cloud deskewing and related research. However, to facilitate visualization in tools like Rviz and compatibility with standard LiDAR processing algorithms relying on ROS messages, we provide format conversion tools to transform the Livox custom message data to the standard ROS message type, *sensor_msgs::PointCloud*. The converted points now hold five values (x, y, z, I, C), where x, y, z represent local Cartesian coordinates, I is the intensity of the point, and C incorporates the line (integer part) and point timestamp (decimal part).

(iii) **Images from RGB camera**. The RealSense D435 camera publishes RGB images at 1920×1080 resolution and a 30 Hz frequency. The message type is $sensor\_msgs :: Image$.

(iv) **Images from the high-resolution spinning LiDAR (OS1-64)** consist of fixed-resolution range images, near-infrared images captured by the laser sensor, and signal images. Each pixel in these images represents the distance

Table 8: Position error (RMSE) for the dynamic scan tracking I^{EKF} and I^{KF} (N/A when the error diverges because the estimated trajectory is incomplete. Unit: meter)

Method	HolybroStnd01	HolybroStnd02	HolybroStnd03	HolybroStnd04	Holybro01	Holybro02	Autel02	Autel03	Tello01	Tello02
$I^{\mathrm{EKF}}_{\mathrm{Avia}}$	0.0431	0.037	0.0425	0.0526	**0.0592**	**0.0649**	**0.0984**	0.183	**0.125**	**0.1182**
$I^{\mathrm{KF}}_{\mathrm{Avia}}$	**0.0415**	**0.0272**	**0.0389**	**0.042**	0.1155	N/A	0.112	**0.0395**	N/A	0.1321
$I^{\mathrm{EKF}}_{\mathrm{Mid\text{-}360}}$	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
$I^{\mathrm{KF}}_{\mathrm{Mid\text{-}360}}$	0.1042	N/A	N/A	N/A	0.0673	N/A	N/A	N/A	N/A	N/A

from the sensor origin to the point, the captured light's strength, and the object's reflectivity, respectively. These images are published at a frequency of 10 Hz and have 16 bits per pixel with a linear photo response. The standard ROS message type, *sensor_msgs/Image*, is used for these image data.

(v) **Inertial data** is available from both spinning and solid-state LiDARs, featuring three built-in 6-axis IMU sensors with a 3-axis gyroscope and a 3-axis accelerometer. The IMU data is published at a frequency of 100 Hz for the spinning LiDAR and 200 Hz for the solid-state LiDAR. The standard ROS message type, *sensor_msgs::Imu*, is employed for IMU data in the rosbags.

(vi) **Ground truth data** is derived from the MOCAP system and is included in the rosbags as *geometry_msgs::PoseStamped* messages. These data are obtained from the computer connected to the OptiTrack cameras via a VRPN connection, providing precise ground truth information.

3.2.3 Dataset Evaluation

This section covers the characterization of the different tracking approaches for the dataset sequences.

Evaluation Metrics

First, to quantify the disparity between the different LiDAR sensors and the external position system estimates, we computed the error by taking the difference between the position estimates obtained from both systems for two distinct positions and orientations of the target. This analysis revealed a Root Mean Squared Error (RMSE) of 0.0143 m.

To quantitatively evaluate the tracking performance, we employed the RMSE metric, and the summarized outcomes are presented in Table 8. Additionally, we provide an overall assessment of each method's performance based on the percentage of successfully estimated trajectories, as shown in Table 9.

Table 9: Percentage of successfully estimated trajectories for the dynamic scan tracking I^{EKF} and I^{KF} on the selected data subset (N/A when the method was not designed for that type of LiDAR)

LiDAR	I^{EKF}	I^{KF}
Livox Avia	100%	80%
Livox Mid-360	0%	20%
Ouster	N/A	N/A

Experimental Evaluation

As part of our dataset, we provide an evaluation of current UAV tracking methods on several sequences. The objective is to compare the performance of different methods to provide a baseline for future research. Throughout this section, we discuss the best methods for different types of LiDAR sensors and environments.

From our analysis, one of the key findings is the pressing need for methods that can enhance UAV tracking with sparse LiDAR data, regardless of the scanning pattern.

Regarding the dynamic tracking method [198] (I^{EKF} and I^{KF}), initially designed for the Livox Horizon LiDAR, we observed that it exhibits strong generalization capabilities and performs effectively on the Livox Avia.

Among the tested methods, we observed that tracking using the dense solid-state Livox Avia LiDAR yields superior results compared to the sparser Mid-360. Specifically, the I^{EKF} method successfully estimated 100% of the selected trajectories, while the I^{KF} method achieved better results on more standard patterns, such as *HolybroStdn*. However, it's worth noting that the dynamic tracking method, being designed for dense solid-state LiDARs, exhibited poor performance on the Mid-360, regardless of the trajectory type and UAV used.

In summary, our evaluation highlights the need for improved UAV tracking methods, especially for sparse LiDAR data, and demonstrates the varying performance of different methods based on the type of LiDAR sensor and scanning environment. These findings lay the groundwork for future research and development in UAV tracking techniques.

3.3 UWB Relative Localization Dataset

This section aims to provide a dataset for calculating relative localization using moving UWB transceivers mounted on top of each robot (in the center). The dataset collected is derived from the work discussed in Section 4.2. To our knowledge, this is the first dataset with only one UWB transceiver on each robot. In the dataset, we have the the data from UWB and also robot odometery.

Each Turtlebot, as illustrated in Figure 34 (b) on page 79 is equipped with a UWB transceiver. In total, four robots are employed, each following different movement patterns as depicted in Figure 34 (a) on page 79. The ground truth positions of the robots are provided by a MOCAP system.

The UWB ranges are recorded as a format of *sensor_msgs/Range* while the odometry data is in *nav_msgs/Odometry*.

3.4 Summary

3.4.1 Multi-Modal LiDAR Dataset for General Purposes

First, we have presented a novel dataset collected with a multi-modal LiDAR sensor system in diverse environments including indoor, outdoor, and forest. The dataset includes data from LiDARs of different types (spinning and solid-state), resolution (16, 64 and 128 channels for spinning LiDARs) and scan patterns (for two different solid-state LiDARs), in addition to a LiDAR camera.This opens the door to future research in general-purpose algorithms, as our analysis shows that different algorithms clearly perform better in one or another type of LiDAR, if they are able to process the data at all. There is therefore a significant gap to be filled in more robust LiDAR odometry, localization and mapping algorithms.

After the date collection, we systematically evaluate multiple open source SLAM algorithms in terms of LiDAR Odometry, and power consumption. The experiments have covered different data sequences across 4 computing platforms. Including the Nvidia Jetson Xavier platform provides further references for the application of various SLAM algorithms on computationally resource-constrained devices such as drones.

3.4.2 Dataset for Specific Applications

UAV Tracking

The aforementioned multi-modal LiDAR dataset is designed for general purposes. In addition, we have developed datasets tailored to specific applications.

We introduce a novel dataset collected using a multi-LiDAR sensor system, covering both indoor and outdoor environments. The dataset includes a variety of LiDAR types with differing resolutions and scan patterns, as well as an RGB camera. It also features UAVs of varying sizes, typical of urban areas. This diverse range of sensors and UAVs offers a unique opportunity for future research on general-purpose algorithms, as our analysis reveals significant performance variations among different algorithms depending on the type of LiDAR used. Consequently, there is a substantial opportunity to develop more robust LiDAR-based UAV tracking algorithms.

To facilitate algorithm analysis, we have included ground truth data for both indoor and outdoor settings. The dataset's distinctive characteristics, encompassing a wide array of data and environmental conditions, set it apart from existing literature. This paper aims to establish a solid foundation for benchmarking and conducting quantitative comparisons between current and future LiDAR-based UAV tracking algorithms. This dataset holds promise for advancing the field of UAV tracking, opening new avenues for cutting-edge research and development.

UWB Based Infrastructure Free Relative Localization

Apart from UAV tracking, we provide a dataset for calculating relative localization based on moving UWB transeivers mounted on top of each robot. It is worth noting that in this dataset, there is one UWB transeiver mounted on a each robot. In total, four robots moves in different patterns included in the dataset with ground truth position giving by a MOCAP system.

4 UWB Based Multi-Robot Relative Localization

4.1 Cooperative Localization Between UAVs and UGVs

Multiple industrial use cases benefit from the deployment of UAVs [199]. When accurate localization is needed, GNSS/RTK is the de-facto standard for gathering aerial data with UAVs [200]. For example, high-accuracy photogrammetry [201], civil infrastructure monitoring [202], or in urban environments where GNSS signals suffer more degradation [200]. As UAVs become ubiquitous across different domains and application areas [203], having access to more flexible and lower-cost solutions to precise UAV navigation can aid in accelerating adoption and widespread use. In this paper, we consider the problem of UAV navigation through relative localization to a companion UGV. We consider a ground robot as a more flexible platform from the point of view of deployment, but in simulations, we also consider localization based on fixed beacons in the environment, closer to how GNSS/RTK systems are deployed.

Within the different approaches that can be used for cooperative relative localization, from visual sensors [204] to cooperative SLAM [205], wireless ranging technologies offer high performance with low system complexity [65]. In particular, ultra-wideband (UWB) wireless ranging offers unparalleled localization performance within the different radio technologies in unlicensed bands [61]. Other benefits of UWB include resilience to multipath, high time resolution, and low interference with other radio technologies [206].

The system that we analyze in this chapter consists of a UGV equipped with four UWB transceivers and a UAV equipped with two transceivers. The UAV transceivers act as initiators, taking turns in sending signals to each of the UGV transceivers. When these respond, the time of flight of the signal is calculated and the distance between each pair of transceivers is calculated. This process is illustrated in Figure 24. The main contribution of this part is thus on evaluating how UWB-based relative localization can improve the positioning of UAVs when supported by ground robots. We simulate different trajectories to evaluate the performance of the system and compare the accuracy of the GNSS, UWB, and VIO approach to localization with field tests in an urban environment. In the simulations, we consider different

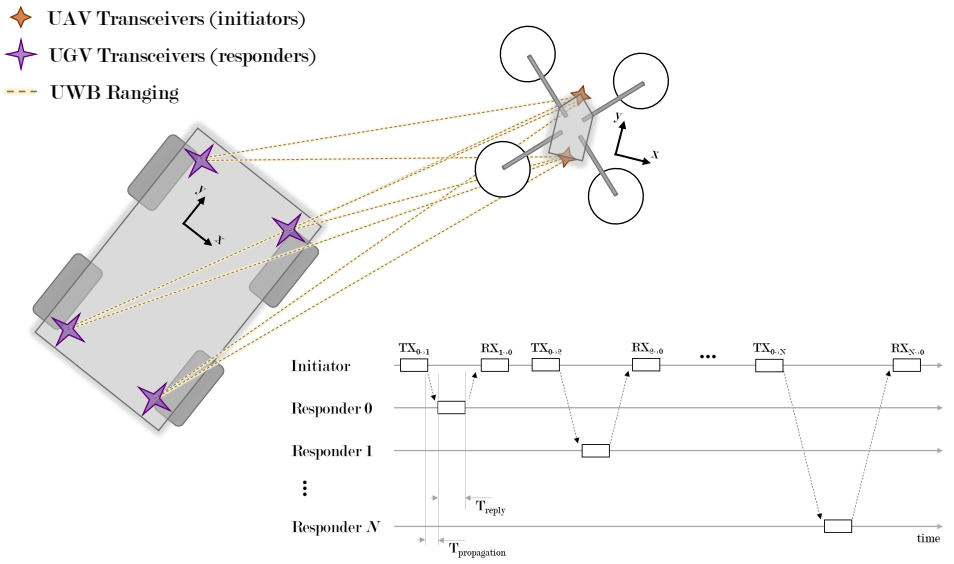


Figure 24: Cooperative localization approach based on UWB ranging measurements from multiple transceivers in different robots

configurations of transceivers in the ground to compare the localization and navigation performance.

4.1.1 Problem Definition

We consider the problem of relative localization between a UAV and a UGV based on UWB ranging between transceivers installed onboard both robots. The objective is to leverage this relative localization to improve the accuracy of the UAV navigation outdoors. We are especially interested in improving the navigation performance in urban areas where the accuracy of GNSS sensors is degraded due to the signal being reflected at or occluded by nearby buildings.

Let us denote by $I = \{I_i\}_{i=0,\dots,N-1}$ the set of N transceivers onboard the UAV. These will act as initiators, i.e., will actively transmit messages to initiate ranging measurements between them and the responder transceivers on the ground. We denote the latter ones by the set $R = \{R_i\}_{i=0,\dots,M-1}$. An initial approach, which we implement, is to iteratively range between each initiator and the set of responders. If the number of nodes increases significantly, more scalable approaches can be used where, for example, a single initiator message is answered by several or all responders with different delays [108].

We model the UWB ranges between an initiator i and a responder j with

$$\mathbf{z}_{(i,j)}^{UWB} = \|\mathbf{p}_i(t) - \mathbf{q}_j(t)\| + \mathcal{N}\left(0,\, \sigma_{UWB}\right) \tag{1}$$

where $\mathbf{p}_i$ and $\mathbf{q}_j$ represent the positions of the initiator and responder transceivers, respectively. Based on the ranges, different approaches to localization include, e.g., multilateration or a least squares estimator (LE). We implement the latter, and hence the position of each tag can be calculated based on the known anchor positions by

$$\mathbf{p}_i = \underset{\mathbf{p}\in\mathbb{R}^3}{\arg\min} \sum_{j=0}^{M} \left(\mathbf{z}_{(i,j)}^{UWB} - \|\mathbf{p} - \mathbf{q}_j\| \right)^2 \tag{2}$$

Alternatively, assuming that the position of initiators in the UAV ($\{\mathbf{p}_i\}$) is given based on the UAV's position and orientation ($\mathbf{p}$ and θ, respectively) by a set of rigid body transformations f_i, i.e., $\mathbf{p}_i = f_i(\mathbf{p}, \theta)$, then the estimator can be used to obtain the full pose of the UAV directly with

$$\mathbf{p},\ \theta = \underset{\substack{\mathbf{p}\in\mathbb{R}^3 \\ \theta\in(-\pi,\pi]}}{\arg\min} \sum_{i=0}^{N} \sum_{j=0}^{M} \left(\mathbf{z}_{(i,j)}^{UWB} - \|f_i(\mathbf{p}, \theta) - \mathbf{q}_j\| \right)^2 \tag{3}$$

4.1.2 Experimental Setup

This section describes the simulation settings and robotic platforms utilized in the field experiments.

Simulation Environment

The first tests are carried out in a simulation environment using ROS and Gazebo. We simulate the UWB ranging with a standard deviation of the Gaussian noise set to $\sigma_{UWB} = 10\,cm$. This is a conservative value based on the literature [65]. We simulate a single transceiver on the UAV and four transceivers on the ground. The latter ones are set are variable distances simulating deployment in small UGVs (0.6 separation), large UGVs (1.2 m separation) and different settings based, e.g., on tripods (with separations at 3 m, 4 m, 12 m and 16 m).

In the simulation experiments, we perform two types of flights. First, a vertical flight where the UAV is set to follow a straight vertical line up to an altitude of 30 m. Second, a flight following a square pattern with a fixed size of 8 by 8 m but at different altitudes (5 m, 10 m and 20 m). For each of these flights, we evaluate the UWB positioning performance with flights based on ground truth positioning. Then, we perform the flight using the UWB position estimation as control input and evaluate how well the UAV follows the predefined trajectory (we refer to this as navigation error).

Algorithm 2: Ground truth extraction

Input:
3D lidar point cloud: $\mathcal{P}$
Last known MAV state: $\mathbf{p}_{MAV}^{k-1}, \dot{\mathbf{p}}_{MAV}^{k-1}$

Output:
MAV state: $\{\mathbf{p}_{MAV}^{k},\ \dot{\mathbf{p}}_{MAV}^{k}\}$

1 **while** *new* $\mathcal{P}_k$ **do**

2
- Generate KD Tree: $kdtree \leftarrow \mathcal{P}$;
- MAV pos estimation: $\hat{\mathbf{p}}_{MAV}^{k} \leftarrow \mathbf{p}_{MAV}^{k-1} + \frac{\dot{\mathbf{p}}_{MAV}^{k-1}}{I}$;
- MAV points: $\mathcal{P}_{MAV}^{k} = KNN(kdtree,\ \hat{\mathbf{p}}_{MAV}^{k})$;
- MAV state estimation: $\mathbf{p}_{MAV}^{k} = \frac{1}{|\mathcal{P}_{MAV}^{k}|} \sum_{p \in \mathcal{P}_{MAV}^{k}} p$;

Multi-Robot System

The multi-robot system employed consists of a single ground robot and a UAV. The ground robot is a ClearPath Husky outdoor platform equipped with four UWB responder transceivers for cooperative positioning and a Livox Avia LiDAR utilized to obtain ground truth. Owing to the lack of a reference system such as a GNSS-RTK receiver, we extract the UAV position from the LiDAR's point cloud and utilize this as a reference. The point cloud is automatically processed following the steps described in Algorithm 2, and manually validated. We refer the reader to [85] for further details on this method. Based on indoor testing with a reference anchor-based UWB system, we have evaluated the ground truth accuracy to be in the order of 10 cm. The UGV and the custom UAV are shown in Figure 25. The UAV is equipped with two UWB transceivers and an Intel RealSense T265 that performs VIO estimation.

Real-World Settings

The UAV runs the PX4 autopilot firmware, which is unable to obtain a stable GNSS lock in the test location. The field experiments are carried out in Turku, Finland (precise location is 60.4557389° N, 22.2843384° E), between a short line of trees and a large building that presumably blocks and reflects GNSS signals. This location is chosen as an example of an urban location where GNSS receivers operate in suboptimal mode.

4.1.3 Cooperative Localization Results

In this sections, we study the performance of the UWB-based cooperative localization system both in simulation and field experiments.

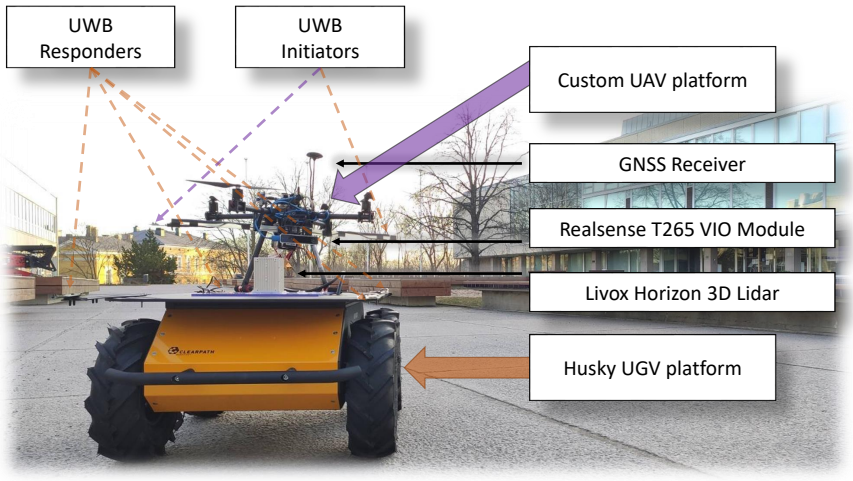


Figure 25: UAV and companion ground robot utilized in the experiments.

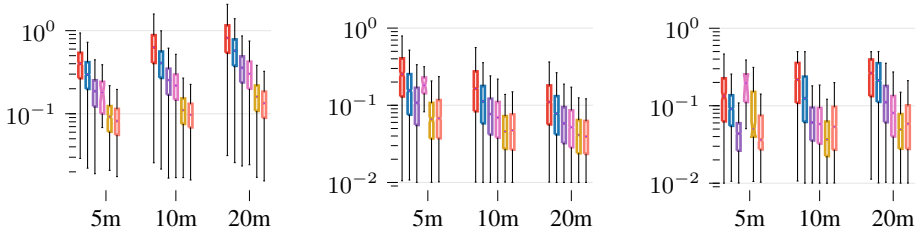
Simulation Results

The positioning and navigation errors for vertical flights are shown in Figure 27. We observe that the positioning error consistently decreases as the anchors become more separated. For the small UGV setting, the error goes over 1 m almost 20% of the time, being highly unstable. It is worth noticing that the navigation error becomes relatively stable with the large UGV anchor distribution (1.2 m separation). Navigation errors are in general lower than their positioning counterparts as the control of the drone is less affected by individual ranging errors, and these tend to average to zero as time passes. It is also worth noticing that the altitude error is significantly lower in all cases when compared to the planar xy error.

Figure 26 then shows the results of flights following a square pattern. We can see that if UWB systems based on fixed anchors separated more than 10 m are utilized, then the navigation error can be consistently maintained below 10 cm. In the case of relying on small or large UGVs, the error is in the tens of centimeters, providing a competitive alternative to GNSS/RTK systems with higher deployment flexibility and lower system complexity.

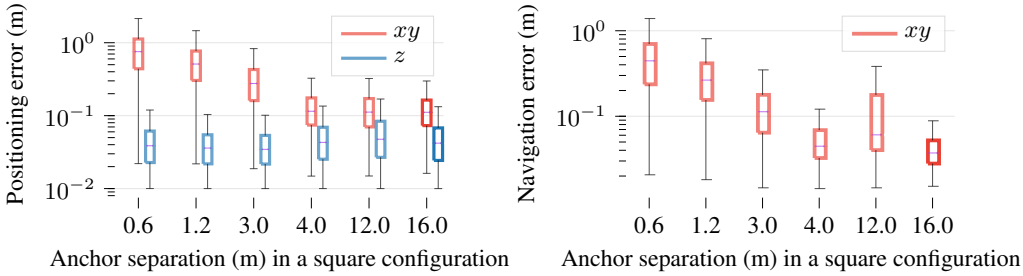
Real-world results

Results from outdoors experiments with real robots are reported in Figure 28 and Figure 29. The former shows a partial extract from the trajectory in 3D, where we can observe that the UWB error is significantly smaller even when the altitude reaches 30 m. In the latter plot we can see that the overall error more than 5 min flight time.



(a) Positioning xy error. (b) Positioning z error. (c) Navigation xy error.

Figure 26: Positioning and navigation errors over a flight following a squared shape of 8 by 8 m, at three different altitudes (5, 10 and 20 m). The altitude is set to a constant so only the XY error is calculated for the UWB-based navigation. The legend has been omitted due to limited space, with the colors representing, from left to right in each group, anchors separated by 0.6 m, 1.2 m, 3 m, 4 m, 12 m and 16 m.



(a) Positioning error based on different anchor distribution settings when doing up and down navigation during a vertical flight.

(b) Navigation error based on different anchor distribution settings when doing up and down navigation during a vertical flight.

Figure 27: Positioning and navigation errors over a vertical flight to an altitude of 30 m. The navigation error includes only the planar distance to the vertical line the drone is set to follow.

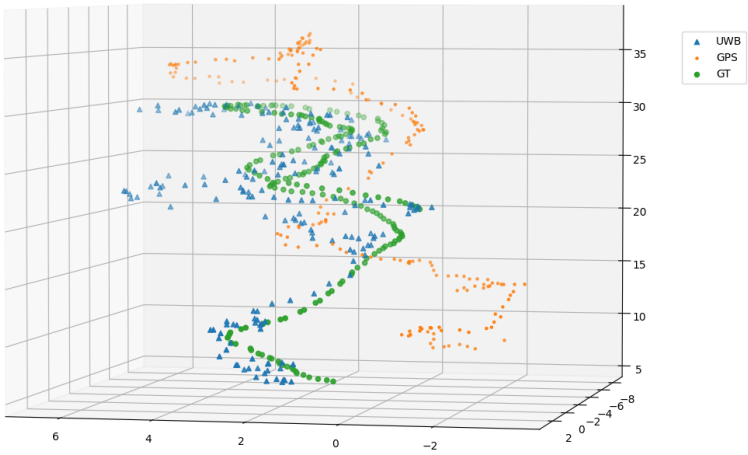


Figure 28: Partial trajectory of the UAV during the outdoors experiment. VIO is not included because it becomes unusable once the UAV reaches 8 m of altitude.

The cooperative UWB approach particularly outperforms both VIO and GNSS estimations in terms of vertical accuracy. In terms of planar xy error, VIO is more accurate but only during the first few seconds of flight, before it rapidly loses accuracy and diverges when the UAV altitude increases. In any case, the cooperative UWB-based localization provides consistent accuracy throughout the flight and therefore has potential for better collection of aerial data through autonomous flights.

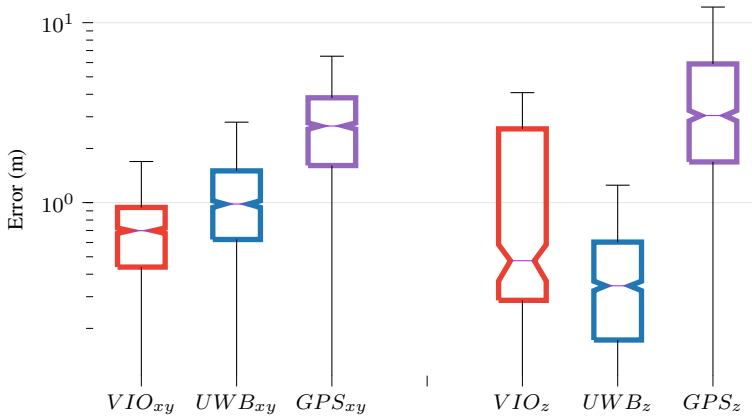


Figure 29: Planar and vertical errors of the different methods during the outdoors flight. The VIO has low error but was only valid for the first few seconds of flight.

4.2 Particle Filter Based Fusion Approach

Since UWB ranging sensors offer low-cost and centimeter-level out-of-the-box accuracy, they have gradually gained attention in autonomous systems applications, including UWB-based state estimation with and without fixed infrastructure, and UWB mesh sensor networks [13; 14]. This technology holds significant potential for relative state estimation in multi-robot systems [15; 16], a crucial yet still challenging research topic in GNSS-degraded environments and outdoors [17; 18]. Moreover, it can serve as the basis for collaborative tasks such as search and rescue, and terrain inspection [19], or extended to transitions of multi-robot systems between indoor and outdoor scenarios [20].

State estimation based on the fusion of UWB and other sensors or estimators, such as LO [42] and VIO [207], has been the subject of numerous research efforts [15; 19; 52]. LO and VIO, which focus on ego-state estimation, are among the most popular and reliable state estimation techniques. However, porting these approaches to multi-robot systems is often either computationally expensive or complex to achieve in realistic scenarios [208; 209]. Either independently or as part of collaborative SLAM processes, multi-robot localization methods often use environment features to adjust for potential drift or error in the relative state estimation [210], or the optimization of map matching or pose graphs [211].

In our approach, we aim to provide a flexible and cost-effective relative multi-robot localization method. Instead of using additional sensors or optimizing raw sensor data matching, we leverage simultaneously detected objects in the environment as measurements without the need for fixed, known landmarks. We refer to this method as cooperative spatial detections, where the 3D pose of the detected objects is identified with respect to each robot's sensor.

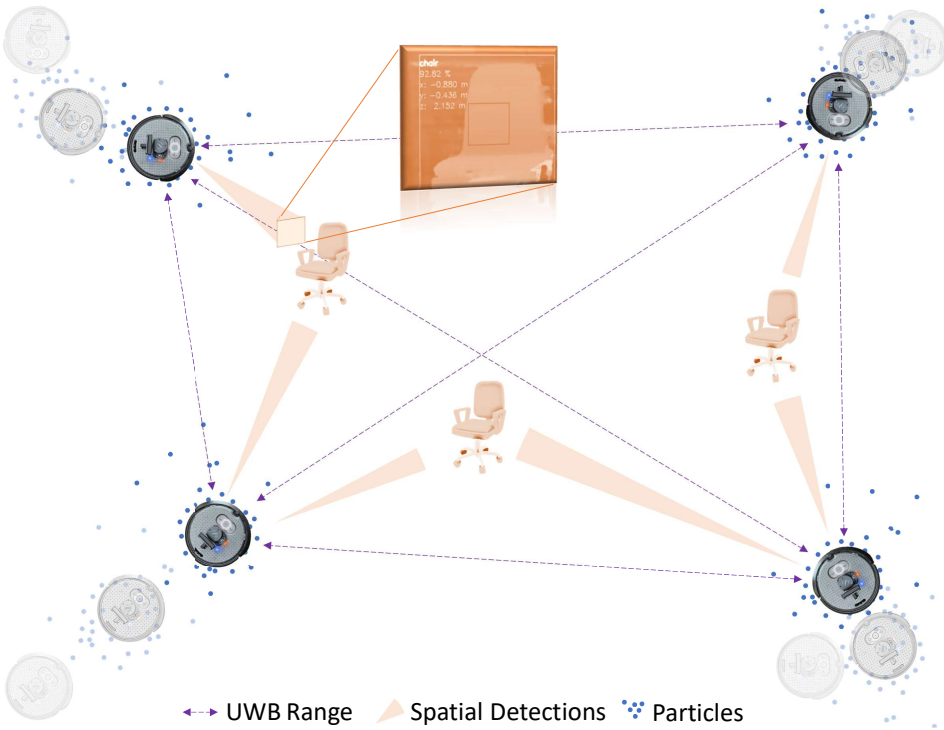


Figure 30: Conceptual diagram of the proposed particle filter-based multi-robot relative localization fusing UWB ranges, robot odometry, and cooperative spatial detections.

It is essential to underline the clear departure of our method from VIO based approaches [15]. Unlike VIO, which relies heavily and continuously on visual input for estimating the robot's pose, our approach minimizes reliance on visual data for localization. Instead, we leverage the camera solely for cooperative spatial detections, wherein simultaneous recognition of objects by multiple robots yields relative positional information.

We, therefore, propose a particle filter (PF) based optimization approach to calculate the relative position among robots. As opposed to traditional methods such as the Extended Kalman Filter (EKF) that may converge to local minima without ensuring global optimality, PF provides a more robust solution to the inherent non-Gaussian optimization challenge. Figure 30 shows a conceptual illustration of the working mechanism and sensing modalities.

Owing to various sources of UWB ranging error, multiple studies have shown the importance of mitigating the UWB ranging error as part of the workflow of UWB data processing [212]. To achieve this, we use a Long Short-Term Memory (LSTM) network to calibrate the UWB range data prior to the PF deployment. In contrast

to previous studies, we evaluate multiple types of LSTM networks, considering not only UWB measurements but also robot orientation.

Another key element concerning existing works in infrastructure-free UWB-based relative positioning is that we assume that only a single UWB transceiver is available for each robot. However, the same methods can be applied with multiple transceivers per robot. This is, to the best of our knowledge, the first infrastructure-free UWB localization system that is generalizable to an arbitrary number of nodes and robots, and that integrates odometry and cooperative spatial detections. We adopt a shared two-dimensional yaw-orientation reference for all robots, where the orientation is not computed relatively between robot pairs but rather with respect to a unified global reference frame. Consequently, at the beginning of the experiment, all robots are initialized with identical orientations.

In summary, the main contributions of this work are the following:

i) a novel and computationally efficient particle filter-based relative localization method fusing odometry and ranging measurements, effective even with only a single range input from UWB;

ii) the addition of an LSTM network for reducing individual range errors calibrated for individual pairs of UWB transceivers (all moving), taking not only the UWB measurements but also the orientation of the robot as input;

iii) the integration of cooperative spatial detections to further increase the accuracy of the relative state estimation;

iv) the demonstration of a practical multi-robot deployment utilizing ROS2 and Zenoh. The seamless integration of Zenoh addresses challenges associated with data flooding in the network, ensuring efficient communication even with numerous robots.

4.2.1 Methodological Overview

The relative positioning framework is conceptually depicted in Figure 30 and the workflow is represented in Figure 31. We use the following notation for the remainder of this part: we consider a group of N robots, or agents, each represented by a position vector $\mathbf{p}_i(t) \in \mathbb{R}^3$ within a shared reference frame, where $i \in \mathcal{V} = \{1, \ldots, N\}$. These agents can measure their relative distances to a subset of the other agents bidirectionally, meaning they can simultaneously calculate a common ranging estimation. Although each robot's position is defined in three-dimensional space, for the purpose of collaborative localization and without loss of generality, we assume that the position of each agent can be represented in a two-dimensional plane as $\mathbf{p}_i(t) \in \mathbb{R}^2$.

We model UWB range measurements with Gaussian noise as follows:

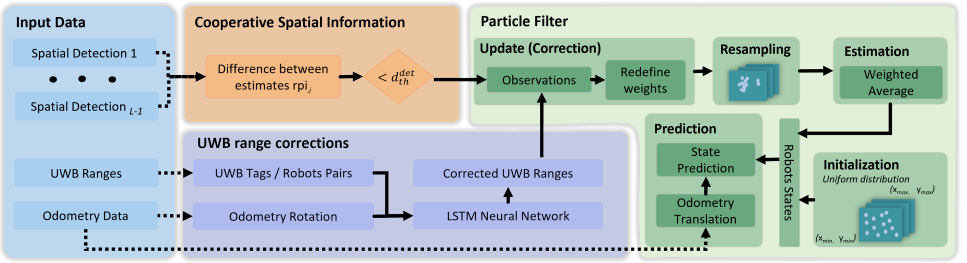


Figure 31: Pipeline (from left to right) of the particle filter-based multi-robot relative position system. d_{th}^{det} is the threshold determining if detections from different robots correspond to the same object.

$$\mathbf{z}_{(i,j),i,j\in\mathcal{V}}^{UWB} = \|\mathbf{p}_i(t) - \mathbf{p}_j(t)\| + \mathcal{N}(0, \sigma_{UWB}) \tag{4}$$

Odometry estimations for each robot are modeled as:

$$\mathbf{z}_{i,\, i\in\mathcal{V}}^{odom} = \begin{bmatrix} \mathbf{R}_i(t-\delta t)\hat{\mathbf{R}}_i(t) & \|\mathbf{p}_i(t) - \mathbf{p}_i(t-\delta t)\| \\ 0 & 1 \end{bmatrix} + \mathcal{N}(0,\ \sigma_{odom}) \tag{5}$$

where δt is the output frequency of the Turtlebot's odometry, $\mathbf{R}_i(t)$ is the orientation matrix for agent i and $\hat{\mathbf{R}}_i(t)$ the relative egomotion estimation in the interval $(t-\delta t, t]$.

Cooperative Spatial Information

Finally, when more than one robot detects the same objects simultaneously as shown in Figure 32, we also include in the PF estimation the cooperative spatial detection measurements modeled as:

$$\begin{aligned} \mathbf{z}_{(i,j),\,\in\mathcal{V}}^{det} &= \mathbf{rp}_{i,j} + \mathcal{N}(0, \sigma_{det}) \\ &\text{where} \\ \mathbf{rp}_{i,j} &= \left(\mathbf{p}_{obj_{l_i}}(t) + \mathbf{p}_i(t)\right) - \left(\mathbf{p}_{obj_{l_j}}(t) + \mathbf{p}_j(t)\right) \end{aligned} \tag{6}$$

Here, $\mathbf{p}_{obj_{l_i}}$ and $\mathbf{p}_{obj_{l_j}}$ represent the position of the simultaneously identified object relative to the reference frame of the robot in position $\mathbf{p}_i$ and $\mathbf{p}_j$, respectively. The vector $\mathbf{rp}_{i,j} \in \mathbb{R}^2$ represents the difference between the two estimates of the position of the object made by the robot at position $\mathbf{p}_i$ and $\mathbf{p}_j$, respectively, with respect to the common reference frame. In other words, it represents the relative position of the object with respect to the robot at position $\mathbf{p}_i(t)$ as observed from the perspective of the robot at position $\mathbf{p}_j(t)$. To ensure that detections correspond to the same object, we use a distance threshold if $\|\mathbf{rp}_{i,j}\| < d_{th}^{det}$ (set to $15\,cm$ in our experiments based on the accuracy of the depth camera).

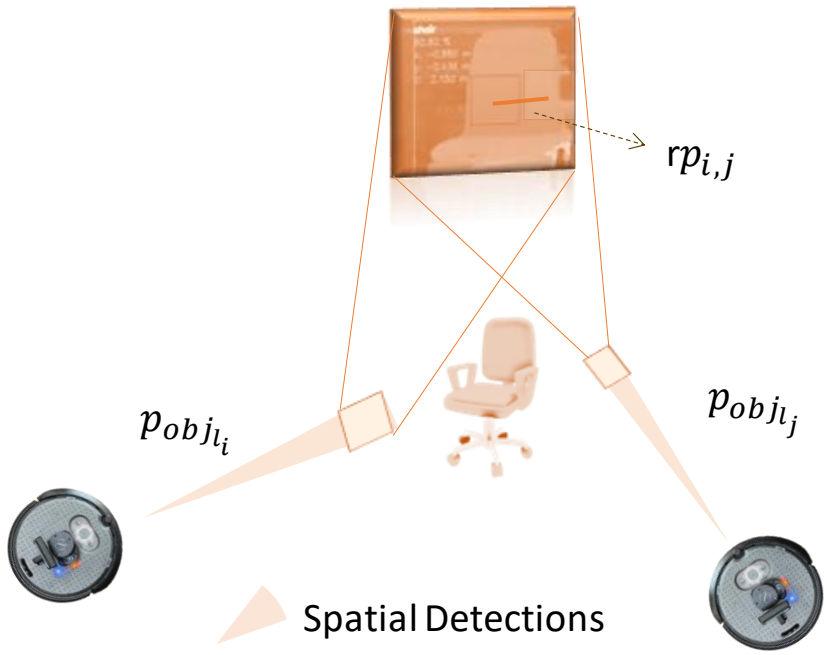


Figure 32: Cooperative spatial detection, where $\mathbf{p}_{obj_{l_i}}$ and $\mathbf{p}_{obj_{l_j}}$ represent the position of the simultaneously identified object relative to the reference frame of the robot in position $\mathbf{p}_i$ and $\mathbf{p}_j$, respectively. The vector $\mathbf{rp}_{i,j} \in \mathbb{R}^2$ represents the difference between the two estimates of the position of the object made by the robot at position $\mathbf{p}_i$ and $\mathbf{p}_j$, respectively, with respect to the common reference frame.

LSTM network for UWB ranging error estimation

In this study, we applied three primary LSTM variants, namely stacked LSTM, bidirectional LSTM, and convLSTM. Each variant was designed and trained individually for each UWB pair utilized in the experiments. Figure 33 shows the customized layer information of these LSTM neural networks.

The inputs to these networks comprise the UWB range, denoted as $\mathbf{z}_{(i,j)}^{UWB}$ between UWB pairs, and the orientations of the robots, θ_i^{odom} and θ_j^{odom}, at the UWB installation ends. To estimate the current UWB ranging error, the LSTM takes a sequence of n_steps data frames $[\mathbf{z}_{(i,j)}^{UWB}, \theta_i^{odom}, \theta_j^{odom}]$ preceding the current time as the input. The dense layer at the end of the LSTM network generates a single UWB ranging error as the output.

The individual LSTM networks are trained using data from a separate experiment that involved deploying the same robots within the same controlled environment and using the Optitrack ground truth system. The training data involved robots moving in

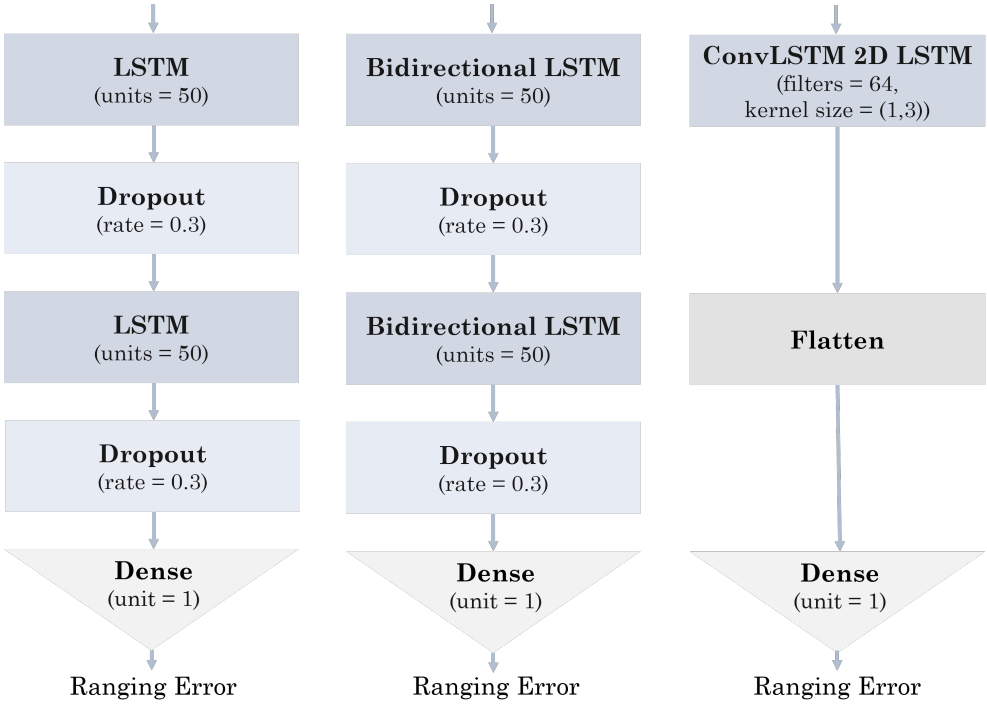


Figure 33: The layer details of customized LSTM neural network variants applied in this study with Figure 33a to 33c illustrating the implementation of stacked LSTM, bidirectional LSTM, and convLSTM, respectively.

circular trajectories, while the LSTM's predictions used in the state estimation were generated for different robot movement patterns as shown in Figure 34. Therefore, we expect the networks to be able to model antenna delays and predict potential errors based on the relative orientation of the antennas. This limits the generalization of the results, but we separate in time the training data from the final experiments reported in this dissertation.

Measurement-Adaptive Particle Filter

The Particle Filter (PF) implementation is outlined in Algorithm 3. The filter primarily consists of three key steps: sampling from UWB ranging estimations s_k (line 2), reweighting based on spatial detections (line 7), and resampling based on the updated weight of each particle (line 11).

a) Initialization: Each particle encapsulates a stack of 2D states of all robots, represented as $s_k = [x_0, y_0, x_1, y_1, ..., x_N, y_N], k \in M$. We initialize the states of the M

Algorithm 3: Multi-robot relative state estimation.

Input: U_r; S_d; O;
Output: $S_k = [x_0, y_0, x_1, y_1, ..., x_N, y_N]$

1 **if** U_r ***not*** *empty* **then**
2 Initialize M particles s_k from uniform distribution or
3 sample from $\mathcal{N}(\tilde{S}_k | S_{k-1} + O_k, \sigma_O)$
4 **for** s_k *in M samples* **do**
5 $\omega_k = \omega_{k-1} \mathcal{N}(U_r - \text{dist}(s_k))$
6 **if** $\|S_d - S_{d,k}\|_2 \leq \varepsilon$ **then**
7 $\omega_k^j = \omega_k \mathcal{N}(S_d - S_{d,j}, \sigma)$
8 **end**
9 **end**
10 Resampling s_k based on ω_k
11 $S_k = \sum_{i=1}^{M} \omega_k s_k$
12 **end**

particle samples (line 2) by drawing from a uniform distribution, leveraging a priori knowledge of the approximate area size where robots are deployed:

$$P(t=0) \sim \mathcal{U}\left((x_{min},\ x_{max}),\ (y_{min},\ y_{max})\right) \tag{7}$$

along with associated weights:

$$w_k(t=0) = \frac{1}{M}, k \in 0, \ldots, M \tag{8}$$

b) Prediction: The state of each particle at time t is predicted based on a Gaussian distribution and odometry data (line 3):

$$\mathcal{N}(\tilde{S}_k | S_{k-1} + O_k, \sigma_O) \tag{9}$$

with $S(t) \in \mathbb{R}^{2N}$ and $\mathbf{z}^{odom}(t) = [\mathbf{z}_i^{odom}(t)]$, while σ_O is derived from the odometry covariance matrix of all robots, $\sigma_{O_i}, i = 1, \ldots, N$.
c) Update (Correction): In this step, the particle filter updates its estimation based on available information, specifically UWB and spatial detection data. When both types of information are present (line 6), the input sample $\mathbf{z}(t)$ is modified to include both inter-robot ranges $\mathbf{z}^{UWB}(t)$ and spatial detections $\mathbf{z}^{det}(t)$, forming a combined observation vector: $\mathbf{z}(t) = stack[\mathbf{z}^{UWB}(t),\ \mathbf{z}^{det}(t)]$.

To refine the estimation in case of spatial detections, The weights of these new samples are determined by multiplying the parent weight with a normal distribution, which accounts for the difference between the parent particle's spatial detection and the observed detection (line 7). Conversely, particles that do not closely match

the observed spatial detection are downweighted based on the UWB measurements alone.

In essence, this process aims to maximize the probability of the distribution by aligning the expected measurements with the actual measurements.

d) Resampling: In every iteration, we randomly re-initialize the particle states by sampling the inverse weight distribution. In our experiments, the resampling proportion is set to 1%.

e) Estimation: Finally, we estimate the states based on the weighted average of the M particles. The estimated state $S(t)$ is given by:

$$S(t) = \sum_{i=1}^{M} w_k(t) \cdot S_k(t) \tag{10}$$

During our experiments, we compared this mean state calculation with an alternative state given simply by the particle with maximum likelihood but we observed more robust overall performance and convergence with the weighted sum.

4.2.2 System and Experimental Design

Multi-Robot Relative Localization System Design

In our experiments, we employed five Turtlebot 4 Lite robots, denoted as $\mathcal{TB}_{i,\, i\in\{0,1,2,3,4\}}$. These mobile robot platforms served for data collection and real-world navigation, utilizing our proposed UWB-based relative state estimation method. To enhance their capabilities, we customized the Turtlebot4 platform, integrating a Qorvo DWM1001 UWB transceiver and an OAK-D stereo camera, which replaced the default OAK-D Lite as shown in Figure 34 (b).

For all our computations, we used a Jetson Nano computer, to which the OAK-D camera is also connected, while the Turtlebot4 ROS 2 drivers run on the existing Raspberry 4.

During the experiments, the UWB transceivers were programmed to iteratively measure the time of flight (ToF) between pairs of robots, providing the necessary ranging measurements. The OAK-D stereo cameras provide poses of the detected objects relative to the cameras as the odometry measurements are given directly by the Turthebot4 ROS 2 drivers.

The experimental site, as depicted in Figure 34 (a) comprised an arena of approximately $8\,m \times 9\,m \times 5\,m$, equipped with an OptiTrack motion capture system (MOCAP) providing the ground truth to verify the estimated relative states of each robot. Within this area, we positioned a static robot $\mathcal{TB}_4$ and four moving robots $\mathcal{TB}_{i,\, i\in\{0,1,2,3\}}$. The static robot serves to align references with the ground truth, while the other four followed specific paths - a triangle, an X, a circle, and a rectangle - as shown in Figure 34 (a). We assumed a common orientation frame among the

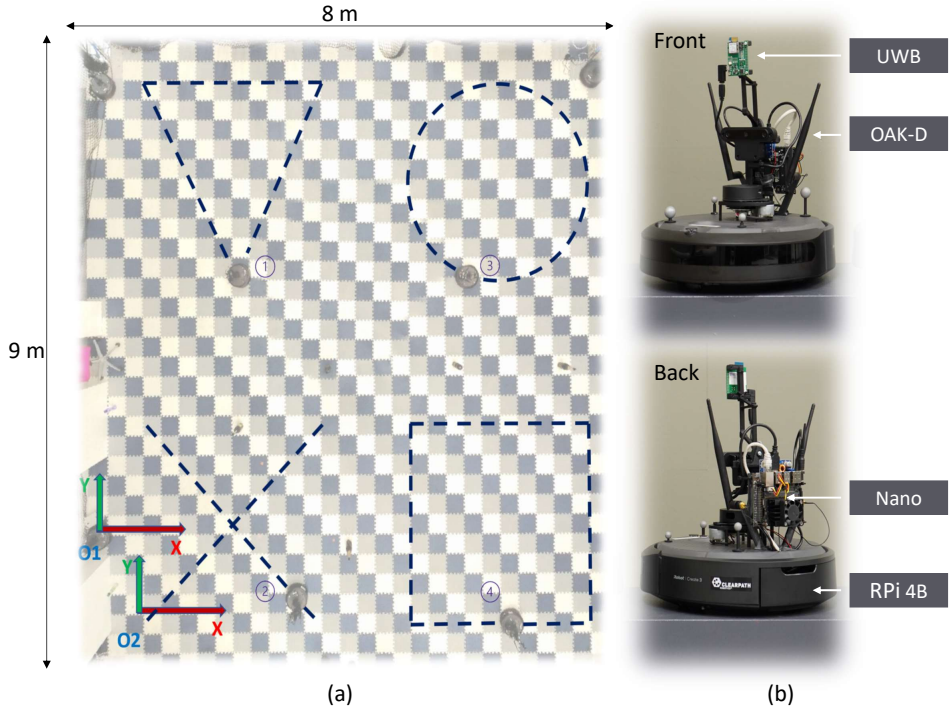


Figure 34: Experimental site and platforms. Subfigure (a) shows the motion capture (MOCAP) arena and the moving patterns of four Turtlebot4 robots. Subfigure (b) shows the customized Turtlebot4 platform mounted with a single UWB transceiver, an OAK-D stereo camera, and a Jetson Nano.

robots, facilitated by a compass, enabling single-range relative state estimation. Importantly, no fixed anchor nodes were used, and the static robot's position remained unknown and assumed mobile a priori.

Finally, to enable cooperative spatial detections, we utilized a pre-trained YOLOv4 network. This network effectively detected several objects, including bottles, chairs, and cups, which were placed arbitrarily both inside and outside the experimental site, serving as references.

Multi-robot relative localization system implementation

The system was developed using ROS 2 as the communication framework for the robots. ROS 2 employs topics as the primary means of communication, and the Turtlebot4 is originally compatible with ROS 2, offering a robust ecosystem. However, when operating in a multi-robot system, challenges arise in differentiating the robots for communication purposes. One approach is to use namespaces to distin-

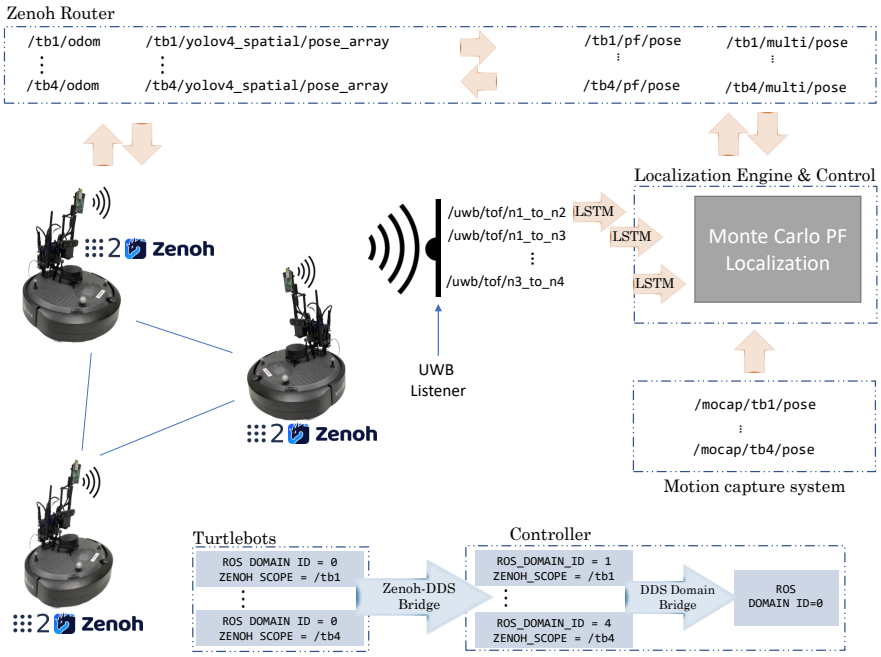


Figure 35: System Implementation with ROS 2 and Zenoh

guish them, but this necessitates significant modifications to the Turtlebot4's software, including the ROS 2 navigation stack. Striking a balance between efficient communication and maintaining the robots' originality poses a considerable challenge.

We adopted a solution that combines ROS 2 with Zenoh, while maintaining DDS as the internal middleware for each individual robot. Figure 35 provides a visual representation of this approach, which involves the robots, the Zenoh Router, and the localization and control engine. This engine can reside on various devices like a computer, Raspberry Pi, or Jetson Nano within the robots. In our implementation, we used a laptop for this purpose, but we demonstrate that the computational resources of any embedded computer are sufficient for the task.

We utilized Zenoh clients with different $ZENOH\_SCOPE$ for each robot, while preserving the common DDS domain ID and topic names. This approach required no alterations to the robot settings (common DDS domain ID and topic names across robots). A Zenoh router was hosted to facilitate message discovery and transfer through the Zenoh DDS bridge[1], enabling seamless communication with the localization and control engine. Ground truth data from the MOCAP system was also received by the engine.

[1] `https://github.com/eclipse-zenoh/zenoh-plugin-dds`

To maintain consistency and simplicity in the communication process, each scope was mapped to a distinct domain ID at the controller end. The DDS domain bridge merged these scopes with topic namespacing[2]. Notably, the topic names remained unchanged on the receiving end (i.e., the engine), allowing for streamlined localization and control procedures. For instance, within the robots, the odometry topic was referred to as $\backslash odom$. However, in the Zenoh communication network, it became $\backslash$tb$\{i\}\backslash odom$ (where $\backslash$tb$\{i\}$ is the Zenoh scope). In the localization engine, the topic names were $\backslash odom$ with different domain IDs, or again $\backslash$tb$\{i\}\backslash odom$ with the common domain ID (with $\backslash$tb$\{i\}$ now serving as a ROS topic namespace).

Regarding the specific implementations, our PF localization is based on the Python pfilter library[3], tailored to suit our program's needs. For LSTM-based UWB ranging error estimation, we built customized LSTM models using TensorFlow Keras, performing both training and real-world inference, as illustrated in Figure 33.

Evaluation on multiple mobile computing platforms

Given the multitude of prevalent computing platforms in contemporary multi-robot systems, presenting a performance evaluation of the proposed approach is valuable as a reference for further research in the field. To this end, we conducted memory consumption and CPU utilization evaluations on several platforms, including an Intel computer (i9-11900H), an NVIDIA Jetson Nano, and a Raspberry Pi 4 B integrated with Turtlebot4. The Intel computer boasts 32 GB of DDR4 RAM, while the Jetson Nano and Raspberry Pi 4 B feature 4 GB RAM. Notably, we executed the PF code without LSTM for UWB ranging error correction, as GPU support is essential for LSTM operations.

In our experiments, the Jetson Nano is needed for both spatial detections and LSTM deployment. However, the majority of the computation regarding spatial detection (both the YOLO detector and the depth data fusion) is done onboard the OAK-D camera on the robot and does not need a GPU in the companion computer.

Navigation based on proposed relative state estimation

To validate the relative state estimated by the proposed approach, we estimate both positioning and navigation errors in a real-world navigation task. For positioning error, the robots follow trajectories using the MOCAP data as feedback to the controller. In the autonomous navigation part, we set one robot to perform a navigation task in the shape of a rectangle using the aforementioned relative position method for controller feedback.

[2] `https://github.com/ros2/domain_bridge`

[3] `https://github.com/johnhw/pfilter`

Table 10: Comparison of the LSTM variants on the estimation of UWB ranging error.

	Stacked LSTM	**Bidirectional LSTM**	**ConvLSTM**
Computation Time (s)	0.0413	0.0381	0.0702
MSE (m^2)	0.0109	0.008	0.0075

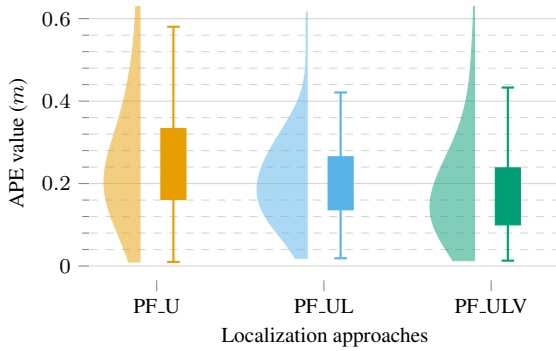


Figure 36: Absolute Positioning Error (APE) for the two-robot, single-range positioning experiment.

4.2.3 Relative Localization Performance Evaluation

UWB ranging error modeling

Table. 10 provides a comparison of Mean Square Error (MSE) and computation time across various LSTM variants, such as stacked LSTM (SL), bidirectional LSTM (BL), and convLSTM (CL). These results indicate that all estimation processes can operate at frequencies exceeding $10Hz$ while delivering notably reduced MSE values. Both SL and BL variants can achieve real-time processing rates surpassing $20Hz$. In light of these findings, we opted to utilize the stacked LSTM (SL) architecture for the subsequent experiments.

Relative State Estimation based on a Single UWB range

We assessed the proposed relative state estimation approach using a single UWB range measurement, focusing on Absolute Positioning Error (APE) and ground truth trajectory analysis. In our results, we identify three variants: PF_U, PF_UL, and PF_ULV, representing the particle filter with only UWB ranges, stacked LSTM corrected UWB ranges, and both corrected UWB ranges with dynamic cooperative spatial detections, respectively.

For experiments involving a single UWB range measurement between two robots, multilateration methods are unable to compute the relative position. Conse-

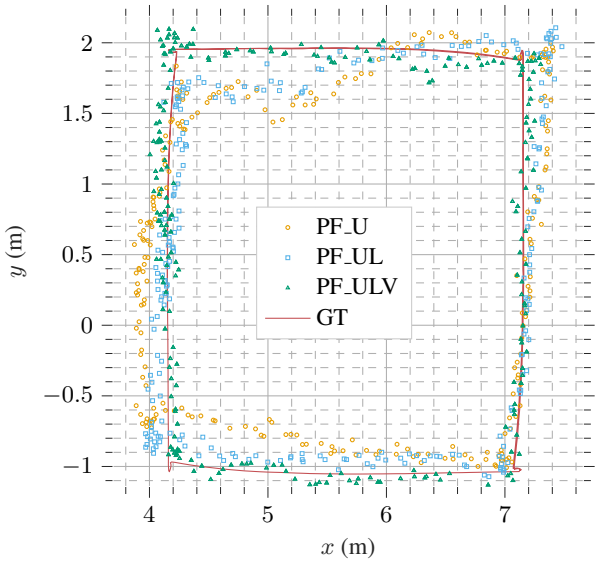


Figure 37: Trajectories of the state estimation with a single UWB range.

quently, we exclusively analyze the APE values and trajectories of PF_U, PF_UL, and PF_ULV in this context.

As depicted in Figure 36, leveraging LSTM correction improves the proposed PF by mitigating UWB ranging errors. Furthermore, integrating cooperative spatial detection enhances performance. These observations are visually depicted in Figure 37.

Relative state estimation for multiple UWB ranges

Figure 38 shows the APE values of the four approaches with *Multi* denoting multilateration applied to robots moving in four distinct patterns. Our proposed PF approach significantly outperforms multilateration. Additionally, incorporating the LSTM network and integrating cooperative spatial detection generally improves the performance of the proposed PF in most cases. For visual representation, Figure 39 illustrates the trajectories of the relative localization approaches.

Resource utilization

Table. 11 reveals that our methods yield low CPU usage (roughly 20%) on Intel PC, Nano, and Raspberry Pi, with memory usage under 500M across all devices. Notably, the program computes relative states among robots, not just individual ones.

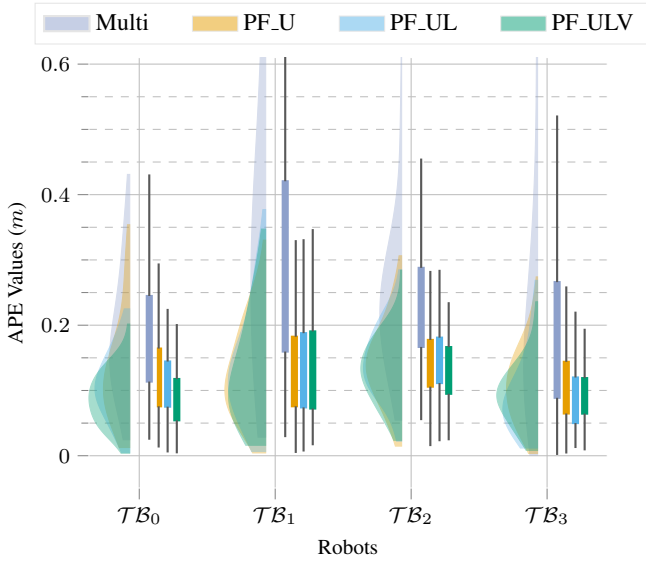


Figure 38: Absolute Positioning Error (APE) values of the four relative localization methods with four robots moving in four distinct patterns.

Table 11: CPU utilization of the different methods.

	Intel PC (i9-11900H)	**Jetson Nano 4GB**	**Raspberry Pi 4B 4GB**
	(CPU (%), RAM (MB))		
Multilateration	(2.9, 390)	(21.3, 99)	(22.4, 66)
Our Approach	(2.8, 69)	(23.2, 442)	(23.2, 83)

Navigation performance

In real-world navigation performance assessment, we compute the median and standard deviation of the APE and Absolute Trajectory Error (ATE). The APE quantifies positioning error, while the ATE assesses the robot's ability to track a prefined path using the proposed approach. Specifically, the APE for positioning error is $0.1094\,m/0.10125\,m$, and the ATE value is approximately $0.0872\,m/0.1011\,m$. The detailed results are shown in Figure 40a. Figure 40b shows the trajectories.

4.3 Summary

In this work about the UWB based infrastructure free multi-robot relative localization, we have presented an analysis on how UWB-based relative localization between a UAV and a companion ground robot can improve the accuracy of autonomous flights outdoors. In particular, we have simulated different scenarios to assess the

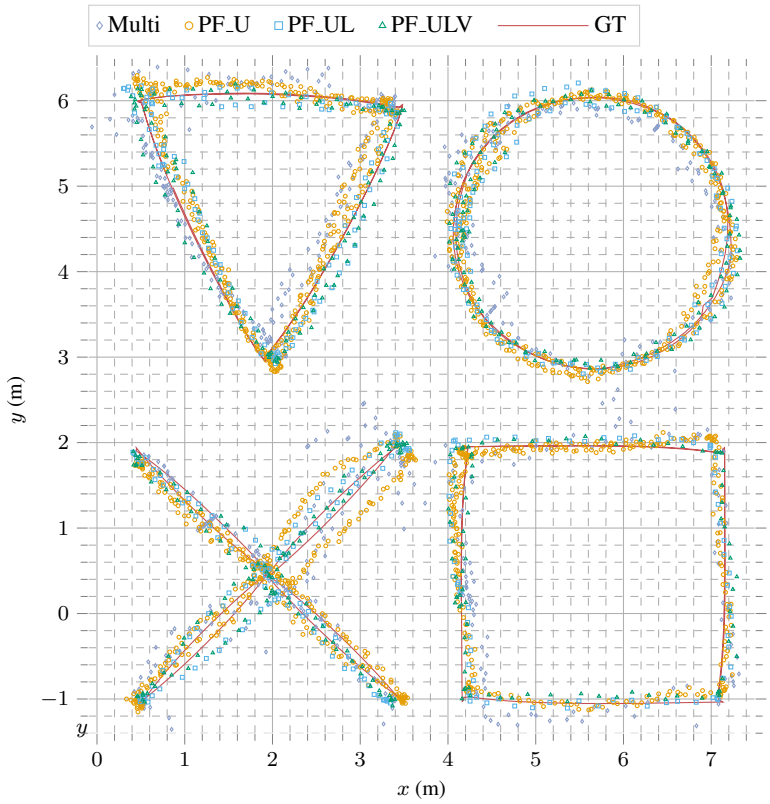
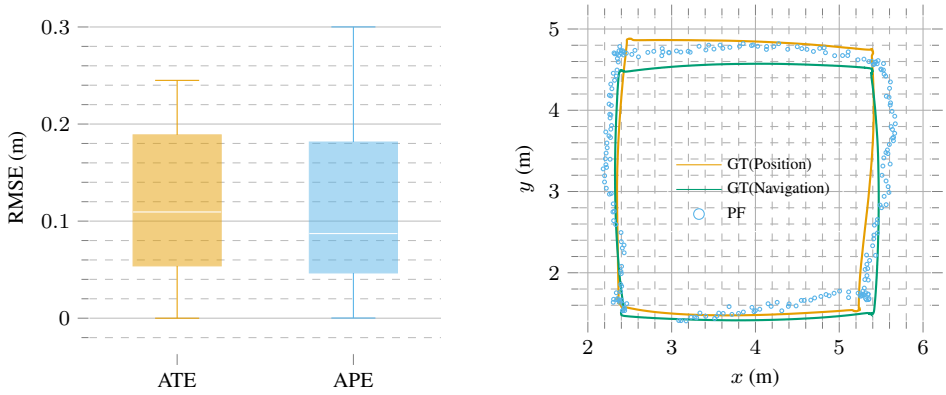


Figure 39: Trajectories of robots moving in distinct patterns with different relative state estimation approaches.

accuracy of the UWB-based relative positioning method. We have then validated this with robots in outdoor experiments, in an urban area where GNSS receivers do not perform optimally. Our analysis includes VIO estimation, which is more accurate at first but loses the reference when the UAV starts gaining altitude, presumably due to the lack of reference points. In summary, we can conclude that UWB-based positioning systems can provide an alternative to RTK-GNSS when the accuracy of standalone GNSS is not enough for gathering aerial data. Moreover, we have proved that even when the transceivers are placed near each other in the ground, mounted on a mobile platform, the accuracy is enough to enable autonomous flight.

Then, we introduce a particle-filter-based approach for relative multi-robot localization, integrating inter-robot UWB ranges, robot odometry, and cooperative spatial detections. Unlike conventional methods such as multilateration, our method can estimate relative positions even with just a single UWB range measurement. We employ LSTM networks trained for each UWB pair to predict ranging errors, ensuring accuracy, reliability, and real-time performance. The application of LSTM to every



(a) (b)

Figure 40: Figure 40a shows Absolute Trajectory Error (ATE) and Absolute Pose Error (APE) for the real-time Turtlebot4 navigation. Figure 40b shows the trajectory of it

UWB-ranging measurement before inputting them into the particle filter enhances accuracy and real-time processing.

Furthermore, in contrast to VIO-based approaches, our method utilizes a camera only for cooperative spatial detections, integrating them when available. Specifically, when two or more robots detect identical external objects, our particle filter dynamically incorporates the extracted spatial information between them as inputs. Experimental results demonstrate that our approach surpasses multilateration for relative state estimation.

Our findings indicate that LSTM error estimation and cooperative spatial information enhance performance compared to using solely UWB measurements. Finally, we evaluate our approach using ROS 2 and Zenoh on various mobile computing platforms, revealing low CPU usage and memory consumption.

5 LiDAR as a Camera

As we discussed in the previous chapter, LiDAR point cloud data nowadays features not only 360° three-dimensional high spatial resolution data but also low-resolution images with 360° FoV obtained with LiDAR sensors by encoding either depth, reflectivity, or near-infrared light in the image pixels. A sample of the data used in this work is shown in Figure 15 on page 42. Since LiDARs measure the Time of Flight (ToF) of a laser signal to objects in the environment, they are not influenced by changes in light such as darkness and daylight.

Albeit the higher cost of LiDAR at the moment, compared with passive visual sensors, LiDARs are inherently more robust to adverse weather conditions and low-visibility environments. They are also a standard part of most of today's self-driving autonomy stacks. Therefore, it comes at no extra cost to leverage their vision-like capabilities in addition to processing the three-dimensional point cloud data.

This work explores the possibilities of applying conventional computer vision technologies on the LiDAR generated images (LiDAR as a camera) by giving a comprehensive analysis on general purposes of DL models, enhancing LO and UAV tracking by fusing LiDAR generated images and point cloud.

In the section, we mainly use the images generated by Ouster LiDAR. The Ouster LiDAR system generates four distinct types of images: ambient, signal, reflectivity, and range images [213]. Each pixel in these images corresponds to a different type of measurement. In range images, each pixel denotes the distance from the sensor origin to a specific point. Signal images display the intensity of light returned to the sensor from a given point. Ambient images capture the intensity of sunlight collected for each point. Reflectivity images represent the reflectivity of the surface or object detected by the sensor.

1. The analysis of the performance of a variety of DL-based visual perception models in LiDAR generated image data. We assess the viability of applying object detection and instance segmentation models to low-resolution, 360° images from two different Ouster LiDARs with different FoVs and range. On the object detection side, we utilize both one-stage detectors (YOLOv5 and YOLOx) and two-stage detectors (Faster R-CNN and Mask R-CNN). For semantic instance segmentation, we study the performance of HRNet, PointRend and Mask R-CNN. (in Section 5.1 on page 88)

2. Keypoint detectors and descriptor based on LiDAR generated images to improve point cloud matching with LiDAR odometry. (in Section 5.2 on page 96)
   (a) We investigate the efficacy of the existing keypoint detectors and descriptors on LiDAR-generated images with multiple specialized metrics providing a quantitative evaluation.
   (b) We conduct an extensive study of the optimal resolution and interpolation approaches for enhancing the low-resolution LiDAR-generated data to extract key points more effectively.
   (c) We propose a novel approach by leverages the detected key points and their neighbors to extract a reliable point cloud (downsampling) for the purpose of point cloud registration with reduced computational overhead and fewer deficiencies in valuable point acquisition.
3. A UAV tracking approach based on the integration of images and 3D point clouds generated by an Ouster LiDAR sensor. The diagram of the approach we follow is illustrated in Figure 56 on page 114. The UAV can be detected in signal images instead of manually giving its initial position as it is needed in other point-cloud-only approaches. The detection also yields an approximate region of interest (ROI) in the point cloud, which will be expanded if no detection occurs. This approach reduces computation overhead by avoiding the need for an overall point cloud search. UAV identification is achieved by clustering points within the ROI, followed by continuous position estimation using the Kalman Filter (KF). (in Section 5.3 on page 114)

5.1 General Purpose Vision Based DL Model Evaluation on LiDAR generated Images

This part of the work focuses on the analysis of the performance of a variety of DL-based visual perception models on LiDAR camera data.

5.1.1 Model Evaluation Readiness

This section covers the hardware and methods utilized in our study. We describe the sensors utilized for data acquisition as well as the different DL model architectures.

Hardware

In the experiment, we use the dataset collected from the Section 3.1 as a preliminary analysis. Subsequently, we acquire more data for specific categories, such as cars, bikes, and persons, required for this study The equipment for data acquisition consists of two spinning LiDARs, the Ouster OS1-64 and the Ouster OS0-128. Table 3

shows the key specifications of these LiDARs, including the resolution of the images that they generate. It is worth noting that the vertical resolution of the images matches the number of channels in the LiDAR.

Figure 41 depicts the data collection platform that can be mounted on different mobile platforms. The two LiDARs are installed on the sides, while an Intel RealSense L515 LiDAR camera captures RGB images.

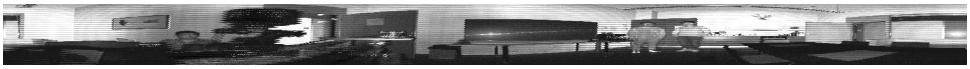
Figure 41: Equipment utilized for data acquisition.

We gathered data in various settings, including indoors and outdoors, day and night. For this initial assessment of the performance of DL models on images generated by the LiDAR sensors, we concentrate on a selection of object categories. These categories have been chosen based on the typical needs of autonomous systems as well as on objects that appear more often in the collected data. Outdoors, we analyze the detection of cars, bicycles and persons. Indoors, persons and chairs. Table 12 shows the number of object instances in the collected data. Samples of the data generated by the sensors are shown in Figure 2 on page 3. In these examples, the resolution of a LiDAR-generated image is 2048×128 with $360°$ FoV of a scene around while an RGB image of L515 is 1920×1080. In our experiment, we utilize signal images due to their superior clarity.

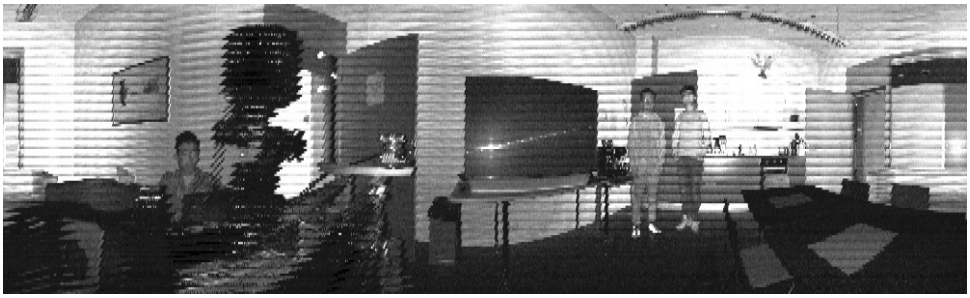
Table 12: Instances of the different objects in the analyzed dataset

	Indoors		Outdoors		
	Person	Chair	Person	Car	Bike
Instances	43	42	103	37	14

While we do not study invariability of object detection or segmentation of the same class of objects across different environments (e.g., indoors and outdoors) or environmental conditions (e.g., day or night), we can assume, based on the works in the literature [12], that the data characteristics do not change significantly. Indeed, one of the key benefits of LiDAR-generated images is that they are not affected by



(a) Original ouster-128 signal images



(b) Ouster-128 signal images after preprocessing

Figure 42: Ouster signal images before and after preprocessing

environmental conditions. Therefore, a person is detected in an almost invariant manner both indoors and outdoors, as long as it is at the same distance and relative position to the sensor. The same applies to daylight or nighttime LiDAR-generated images.

Data Preprocessing

One of the main drawbacks of LiDAR-generated images (signal images) is the low vertical resolution, which is only up to 128 pixels in the highest-performance sensors. Our early experiments showed low performance of the different detection and segmentation models due to the high distortion in the untraditional image ratio. To address this issue with data preprocessing, we performed two main steps: denoising and interpolation using the OpenCV libraries with Python. We considered different denoising and interpolation approaches and repeatedly ran object detection and segmentation on a set of test images. In our experiments, we applied a box filter to denoise the images and linear interpolation methods to properly resize the images to the dimension of 1000×300. Figure 42 shows the original signal image in Figure 42a and the one after the preprocessing in Figure 42b.

Object Detection Approaches

Over the last decade, deep neural network models have achieved significant advances in computer vision, especially object detection. Object detection, which includes both object recognition and localization, is generally divided into two types: one-stage and two-stage detectors [214]. In this study, some of the most commonly used models from both frameworks are utilized for object detection.

Two-Stage Object Detection

A two-stage detector divided the detection process into region proposal and classification phases. At the region proposal phase, several object candidates are proposed as ROI, classified and localized in the second phase. Object localization and detection are typically more accurate in models with a two-stage architecture than in others. Two popular two-stage detectors were used in this study: FasterR-CNN [215] and MaskR-CNN [216]. These models are implemented based on Pytorch, and ResNet-50 is used as the pre-trained backbone for object detection.

One-Stage Object Detection

In contrast to two-stage models, one-stage detectors utilize a single feed-forward fully-convolutional network for object feature extraction, bounding-box regression, and classification. In the one-stage approach, feature maps are detected and classified simultaneously. In addition to their excellent accuracy, the one-stage detector models are popular in real-time applications due to their high detection speed. One of the first widely adopted one-stage detectors in the deep learning field was YOLO, which was introduced in [217]. Two variations of the YOLO model were applied in this study: YOLOx [218] and YOLOv5 [219]. In the YOLOx toolset, there are different types of networks, including the YOLOx-s, YOLOx-m, YOLOx-l, and YOLOx-x models. We use the YOLOx-m model in this dissertation due to its high detection speed and performance.

Image Segmentation Approaches

Object segmentation is the process of assigning each pixel value of an image to a specific class and is generally divided into two main types (at the time of this analysis): semantic segmentation and instance segmentation. The semantic segmentation method considers objects that belong to the same class as a single group [220], while the instance segmentation method combines semantic segmentation and object detection approaches and identifies multiple objects of a single class as distinct instances [221].

For semantic segmentation, HRNet + OCR + SegFix (High-Resolution Network) which placed 1st in the Cityscapes competition at ECCV 2020, is used [222]. HRNet + OCR + SegFix is the integration of HRNet, OCR, and SegFix to provide a powerful tool for precise localization of text or objects in images that require high-resolution feature extraction. HRNet is a DL architecture designed for high-resolution images that capture fine-grained details and global context through a parallel multi-resolution pyramid structure. OCR is an optical character recognition technology that allows computers to recognize and interpret a text in images. SegFix is a post-processing technique for image segmentation that corrects errors by using context from neighboring pixels.

Additionally, Pointrend [223] and Mask R-CNN, both with ResNet-50 as their

		Faster R-CNN	Mask R-CNN	YOLOv5	YOLOx
Indoors	Person	0.837	0.837	0.924	**0.953**
	Chair	0.357	0.333	0.398	**0.515**
Outdoors	Person	0.524	0.485	0.630	**0.633**
	Car	0.865	0.811	**0.893**	0.866
	Bike	0.357	**0.643**	0.143	**0.571**

Table 13: Proportion of objects successfully detected by each of the models studied in this work. This metric does not include false negatives or false positives.

		Faster R-CNN		Mask R-CNN		YOLOv5		YOLOx	
		Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
In	Person	0.72	0.837	0.95	0.905	0.976	0.930	**1.0**	**0.953**
	Chair	1.0	0.115	0.57	**0.826**	1.0	0.115	**1.0**	0.315
Out	Person	0.912	0.505	0.957	0.464	0.872	0.854	0.969	0.653
	Car	0.943	0.688	0.712	0.627	0.919	0.829	0.825	0.618
	Bike	0.357	1.00	0.643	1.00	0.143	1.00	0.571	1.00

Table 14: Detection accuracy of multiple representative object detection networks in various scenarios

backbone, are employed, for the instance segmentation. Particularly, PointRend is a cutting-edge technique for instance segmentation, which predicts point-wise predictions for each pixel in an image and selectively refines them based on context using a context-adaptive CNN. This selective refinement approach achieves state-of-the-art results with fewer computational resources than traditional instance segmentation techniques. PointRend is flexible and easily integrated into existing pipelines, making it a popular technique in computer vision. It has demonstrated impressive results on various datasets.

5.1.2 Model Evaluation Results

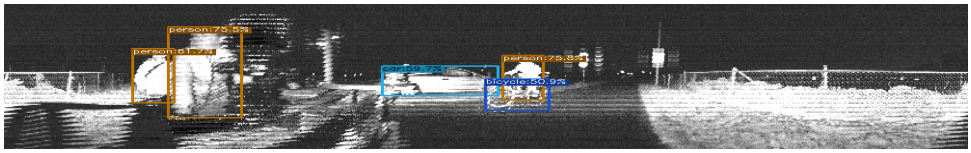
Through this section, we cover the results of applying the different object detection and instance segmentation models to the data gathered in the different environments. We collected and manually annotated the LiDAR-generated signal images. We used RGB images from a separate camera and the LiDAR point cloud data to validate the annotations through visual observation.

Detection Results

The first part of the analysis delves into the performance of the different objectors. Table 13 shows the proportion of objects successfully detected by Faster R-CNN,

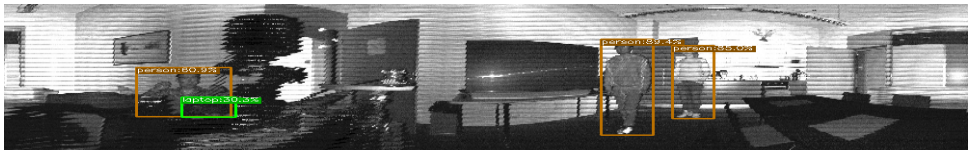
Mask R-CNN, YOLOv5, and YOLOx. Among them, YOLOx has a higher proportion of detected objects indoors and outdoors. It is worth noting that all four models were able to detect over 80% persons indoors and over 80% of cars outdoors. In general, the performance of all the models is good enough to consider the adoption of this type of object detection in systems where the LiDARs are already present. For more specific metrics, in Table 14, we show the precision and recall of the detectors. YOLOx has the most robust overall performance of the four different tested models.

Some other categories, including stop signs, handbags, and fire hydrants, are considered in our initial evaluation. However, they are not listed in Table 13 and Table 14 as we have focused on better analyzing a specific subset. In general, we have observed that both YOLOv5 and YOLOx can achieve comparable accuracy in these other classes as well.



(a) YOLOx detections in an outdoor scene

(b) YOLOv5 detections in an outdoor scene



(c) YOLOx detections in an indoor scene

(d) FasterR-CNN detections in an outdoor scene

(e) MaskR-CNN detections in an outdoor scene

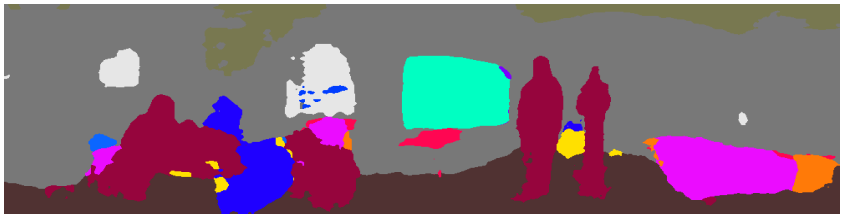
Figure 43: Detection examples in indoor and outdoor scenarios

In Figure 43, we show a sample of detection examples from YOLOv5 in Figure 43b, YOLOx in Figure 43a, Figure 43c, FasterR-CNN in Figure 43d, and Mask R-CNN in Figure 43e for both indoor and outdoor scenes.

In our experiments, single-stage object detectors outperform two-stage methods. The literature in the area points to the better overall performance of two-stage models. However, for the data studied in this dissertation, this does not hold. In any case, the limited amount of data for our tests is not enough to conclude that single-stage detectors are always better for LiDAR-generated data.

Segmentation Results

Regarding the performance of instance segmentation models, we show in Figure 44 shows examples of HRNet semantic segmentation in both indoor and outdoor scenes included. In Figure 45, we also show examples of instance segmentation results with PointRend and Mask R-CNN. In this case, the analysis is qualitative, and further results are available in the project's repository[1]. Nonetheless, our tests show good performance for the most typical object classes based on analyzing a broad series of images.



(a) HRNet: indoor example.

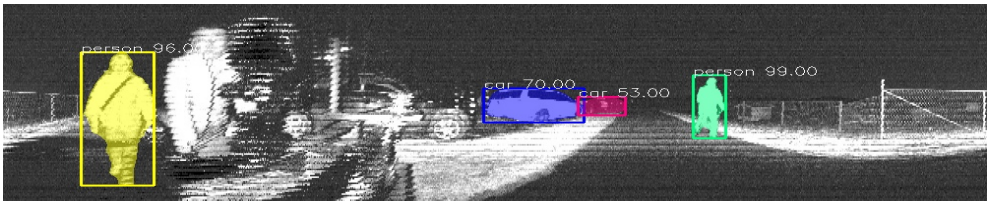
(b) HRNet: outdoor example.

Figure 44: Indoor and outdoor semantic segmentation examples based on HRNet.

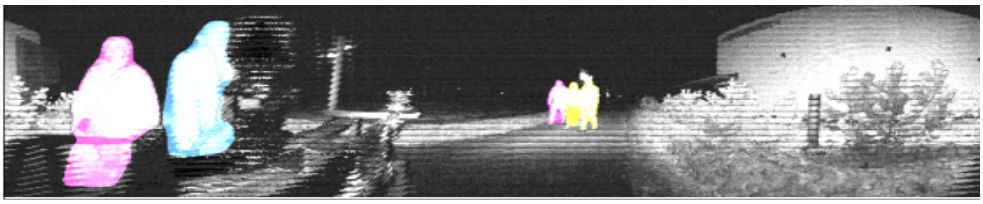
[1] https://github.com/TIERS/lidar-as-a-camera

Real-time performance evaluation

We evaluated the real-time performance of multiple representatives from the above approaches including YOLOv5, Faster R-CNN from detection tasks, and PointRend from the segmentation tasks. The computing platform utilized is an Nvidia GeForce RTX 3080 GPU with 16GB GDDR6 VRAM. YOLOv5 with YOLOv5s model has an average inference frequency of 24 HZ. Faster R-CNN with ResNet50 FPN model averages to 15 HZ. Additionally, the PointRend with ResNet50 as the backbone averages 15 HZ.



(a) PointRend



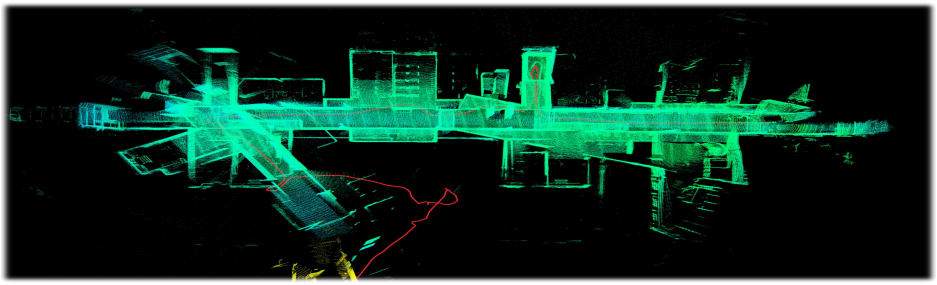
(b) Mask R-CNN

Figure 45: Indoor and outdoor instance segmentation examples based on PointRend and Mask R-CNN

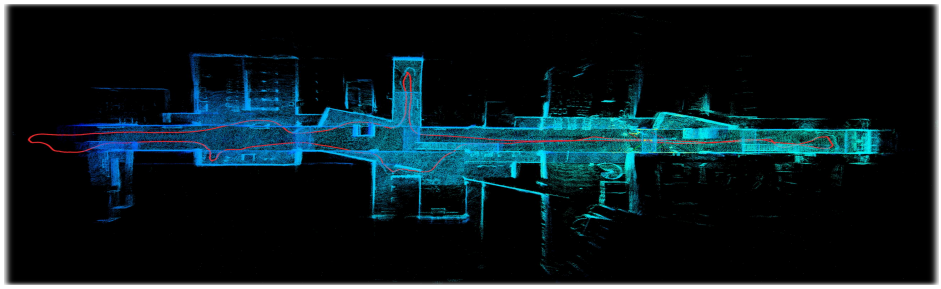
5.2 Assisting Point Cloud Registration by Extracting Keypoints from LiDAR Generated Images

LiDAR Odometry (LO), as a fundamental component in robotics has significantly drawn our attention. Point cloud matching or registration constitutes the key component in LO. There are multiple algorithms can facilitate this process. Since its inception approximately three decades ago, the Iterative Closest Point (ICP) algorithm, as introduced by Besl and McKay [224], has spawned numerous variants. These include notable adaptations such as voxelized Generalized ICP (GICP) [225], CT-ICP [226], and KISS-ICP [227]. Among these ICP iterations, KISS-ICP, denoting "keep it small and simple", distinguishes itself by providing a point-to-point ICP approach characterized by robustness and accuracy in pose estimation. Furthermore, the Normal Distributions Transform (NDT) [189] represents another prominent point cloud registration technique frequently employed in LO research.

Extensive research efforts have focused on the integration of diverse sensors, including IMUs, to bolster LO performance. However, in scenarios where LiDAR data lacks geometric distinctness or even contains misleading information, the process of point cloud registration continues to present challenges in achieving precise estimations and even causing drift in certain cases (Figure 46a).



(a) Raw point cloud with point cloud matching approach (KISS-ICP)



(b) Our proposed LiDAR-generated keypoint extraction-based approach

Figure 46: Samples of LiDAR odometry results run in our experiment

LiDAR generated images as we mentioned, are low-resolution but possibly panoramic and exhibit heightened resilience and robustness in challenging environments, such as those characterized by fog and rain, compared to conventional camera images. Additionally, these images can potentially provide crucial information for point cloud registration when there is a deficiency of geometric data or the raw point cloud lacks useful information so as to avoid drift (Figure 46b from our proposed approach).

Keypoint detectors and descriptors have found extensive utility across diverse domains within visual tasks such as place recognition, scene reconstruction, VO, VSLAM, and VIO. Nevertheless, there remains a lack of investigation into the performance of extant keypoint detectors and descriptors when applied to LiDAR-generated imagery. Contemporary methodologies for VO or VIO rely significantly on the operability of visual sensors, necessitating knowledge of camera intrinsics to facilitate Structure from Motion (SfM) – a requisite not met by LiDAR-generated images. This poses the difficulty of extracting key points from LiDAR-generated images in a certain way to further apply in the odometry estimation.

To address these challenges, this work investigates the efficacy of the existing keypoint detectors and descriptors on LiDAR-generated images with multiple specialized metrics providing a quantitative evaluation. We conduct an extensive study of the optimal resolution and interpolation approaches for enhancing the low-resolution LiDAR-generated data to extract key points more effectively. Additionally, we propose a novel approach by leverages the detected key points and their neighbors to extract a reliable point cloud (downsampling) for the purpose of point cloud registration with reduced computational overhead and fewer deficiencies in valuable point acquisition. As the latest ICP approach, KISS-ICP is the designated methodology for the point cloud registration we adopted in this study.

5.2.1 Overview of Keypoint Detector and Descriptor

In recent years, there have been multiple widely applied detectors and descriptors in the field of computer vision. As illustrated in Table 15, we've captured the essential characteristics of different detectors and descriptors.

Harris detector [228] can be seen as an enhanced version of Moravec's corner detector [229] [230]. It's used to identify corners in an image, which are the regions with large intensity variations in multiple directions. The Shi-Tomasi Corner Detector [231], is an improvement upon the Harris Detector with a slight modification in the corner response function that makes it more robust and reliable in certain scenarios. The Features from Accelerated Segment Test (FAST) [232] algorithm operates by examining a circle of pixels surrounding a candidate pixel and testing for a contiguous segment of pixels that are either significantly brighter or darker than the central pixel.

Table 15: Keypoint detectors and descriptors

Method	Detecto	Descriptor	Description
Harris	✓		Corner detection method focusing on local image variations.
Shi-Tomasi	✓		Variation of Harris with modification in the response function to be more robust.
FAST	✓		Efficient corner detection for real-time applications.
FREAK		✓	Robust to transformations, based on human retina's structure.
BRIEF		✓	Efficient short binary descriptor for key points.
SIFT	✓	✓	Invariant to scale, orientation, and partial illumination changes.
SURF	✓	✓	Addresses the computational complexity of SIFT while maintaining robustness.
BRISK	✓	✓	Faster binary descriptor method, efficient compared to SIFT/SURF.
ORB	✓	✓	Combines FAST detection and BRIEF descriptor, commonly used now.
AKAZE	✓	✓	Builds on KAZE but faster, good for wide baseline stereo correspondence.
Superpoint	✓	✓	A state-of-the-art AI approach that exhibits superior performance when applied to traditional camera images.

For descriptor-only algorithms, Binary Robust Independent Elementary Features (BRIEF) [233] utilizes a set of binary tests on pairs of pixels within a patch surrounding one key point. Fast Retina Keypoint (FREAK) [234] is inspired by the human visual system, which constructs a retinal sampling pattern that is more densely sampled towards the center and sparser towards the periphery. Then it compares pairs of pixels within this pattern to generate a robust binary descriptor.

With respect to the combined detector-descriptor algorithms, the Scale-Invariant Feature Transform (SIFT) [235; 236] detects key points by identifying local extrema in the Difference of Gaussian scale-space pyramid, then computes a gradient-based descriptor for each keypoint. Speeded-Up Robust Features (SURF) [237] is designed to address the computational complexity of SIFT while maintaining robustness to various transformations. Binary Robust Invariant Scalable Keypoints (BRISK) [238] uses a scale-space FAST [232] detector to identify key points and computes binary

descriptors based on a sampling pattern of concentric circles. Oriented FAST and Rotated BRIEF (ORB) [239] extends the FAST detector with a multi-scale pyramid and computes a rotation-invariant version of the BRIEF [233] descriptor, aiming to provide a fast and robust alternative to SIFT and SURF. Accelerated-KAZE (AKAZE) [240] employs a Fast Explicit Diffusion scheme to accelerate the detection process and computes a Modified Local Difference Binary (M-LDB) descriptor [241] for robust matching.

The emergence of DL techniques, particularly CNN, has revolutionized computer vision, over the last decade. SuperPoint [242] detector employs a fully CNN to predict a set of keypoint heatmaps, where each heatmap corresponds to an interest point's probability at a given pixel location. Then, the descriptor part generates a dense descriptor map for the input image by predicting a descriptor vector at each pixel location.

To sum up, while numerous detector and descriptor algorithms have gained popularity, it is imperative to note that they have primarily been designed for traditional camera images, not LiDAR-based images. Consequently, it's of paramount importance for this study to identify the algorithms that maintain efficacy for LiDAR-based images.

Table 16: Metrics for evaluating keypoint detectors and descriptors

Metrics	**Description**
Number of Keypoints	A high number of key points can always lead to more detailed image analysis and better performance in subsequent tasks like object recognition.
Computational Efficiency	Computational efficiency remains paramount in any computer vision algorithm. We gauge this efficiency by timing the complete detection, description, and matching process.
Robustness of Detector	An effective detector should recognize identical key points under varying conditions such as scale, rotation, and Gaussian noise interference.
Match Ratio	The ratio of successfully matched points to the total number of detected points, offers insights into the algorithm's capability in identifying and relating unique keypoints.
Match Score	A homography matrix is estimated from two point sets, to distinguish spurious matches, then the algorithm precision is quantified by the inlier ratio.
Distinctiveness	Distinctiveness entails that the key points isolated by a detection algorithm should exhibit sufficient uniqueness for differentiation among various key points.

5.2.2 Evaluation Metrics for Keypoint Detectors and Descriptors

The efficacy of detector and descriptor algorithms is typically assessed through some specific evaluation metrics. It is worth noting we are interested in the metrics which do not require ground truth data labelling in this study. As illustrated in Table 16, the first three metrics, Number of Keypoints, Computational Efficiency, and Robustness of Detector are straightforward to comprehend and implement, and also widely adopted in numerous studies [243; 237; 244]. For instance, the Robustness of the Detector [245] is implemented by contrasting key points before and after the transformations like *scaling*, *rotation*, and *Gaussian noise interference*.

When assessing the precision of the entire algorithmic procedure, which is prioritized by the majority of tasks, the prevalent metrics often necessitate benchmark datasets, such as KITTI [246], HPatches [247]. These datasets either provide the transformation matrix between images or directly contain the key point ground truth. For example, in Mukherjee et al.'s study [248], one crucial metric: Precision, is defined as *correct matches/all detected matches*, where correct matches are ascertained through the geometric verification based on a known camera position provided by dataset [249]. Similarly, in this recent work [250], the evaluation tasks including "keypoint verification", "image matching", and "keypoint retrieval", all rely on the homography matrix between images in the benchmark dataset [247].

Nevertheless, given that research predicated on LiDAR images is at a nascent stage, there exists no benchmark dataset in the field of LiDAR-based images. And the effort required for data labeling [251; 252] to produce such a dataset is considerable and challenging. To bridge this gap, we select multiple key evaluation metrics: Match ratio, Match score, and Distinctiveness, as shown in Table 16 from previous studies. Match Ratio [248] is quantitatively defined as *number of matches/number of key points*. A high Match Ratio can suggest that the algorithm is adept at identifying and correlating distinct features; While the exact homography matrix between images remains unknown when lacking benchmark datasets, it can be approximated using mathematical methodologies from two point sets. This computed homography can subsequently be utilized to find correct matches. And *number of estimated correct matches/number of matches* is denoted as Match Score in our work; And Distinctiveness is computed as follows: For every image, the k-nearest neighbors algorithm, with k=2, is employed to identify the two best matches [235]. If the descriptor distance of the primary match is notably lower than that of the secondary match, it demonstrates the algorithm's competence in recognizing and describing highly distinctive key points. Consequently, this defines the metric: Distinctiveness.

5.2.3 Keypoint Extractor Evaluation

Utilized Data

For the evaluation of keypoint detectors and descriptors and our proposed approach, we utilized the collected dataset for multi-modal LiDAR sensing in Chapter 3. The dataset consists of various LiDARs and among them, Ouster LiDAR provides not only point cloud but also its generated images. The Ouster LiDAR applied in the dataset is OS0-128 with its detailed specifications shown in Table 3.The images generated by Ouster OS0-128 shown in Figure 2 include signal images, reflectivity, ambient, and range images with its expansive $360^\circ \times 90^\circ$ FoV as we mentioned in the previous sections.

As indicated by the findings of our previous research, signal images have exhibited superior performance in the execution of conventional DL tasks within the domain of computer vision [136]. In light of this, for the first two parts of our experiment, we opt to employ signal images from the *indoor_01_square* scene provided by the dataset, which is a scene that spans 114 seconds and comprises 1146 images.

Optimal Preprocessing Configuration Searching for Signal Images

LiDAR-generated images at hand are typically panoramic but low-resolution. Moreover, these images often exhibit a substantial degree of noise. This prompts a concern of utilizing the original images for facilitating the functionality evaluation of the keypoint detector and descriptor algorithms. And our preliminary experiments have evinced unsatisfactory performance across an array of detectors and descriptors when employing the unaltered original LiDAR-generated images. To identify the optimal resolution and interpolation methodology for augmenting image resolution, an extensive comparative experiment was conducted.

In this part, we implement an array of interpolation techniques on the original images, employing an extensive spectrum of image resolution combinations. The interpolation methodologies encompass bicubic interpolation (CUBIC), Lanczos interpolation over 8×8 neighborhood (LANCZOS4), resampling using pixel area relation (AREA), nearest neighbor interpolation (NEAREST), and bilinear interpolation (LINEAR). The primary procedure of the preprocessing is displayed in Algorithm 4.

More specifically, we iterate a range of image dimensions and interpolation methods in conjunction with the suite of detector and descriptor algorithms designated for evaluation. Each iteration involves a rigorous evaluation of a comprehensive metrics set detailed in Table 16. Following a quantitative analysis, we compute mean values for these metrics. This extensive assessment aims to identify the optimal preprocessing configuration that offers a balanced performance for different keypoint detectors and descriptors.

Algorithm 4: Preprocessing configuration evaluation

Input:
N number of signal images: $\{S_i\}, i \sim N$;
Interpolation methods:
$IA = \{CUBIC, LANCZOS4, AREA, NEAREST, LINEAR\}$;
Targeted Width: $TW = \{min : 512; max : 4096; step : 128\}$;
Targeted Height: $TH = \{min : 32; max : 256; step : 32\}$;
Detectors and descriptors:
$DET = \{SURF, SIFT, SHITOMASI, HARRIS, BRISK, FAST, AKAZE, ORB\}$;
$DES = \{FREAK, SIFT, BRISK, SURF, BRIEF, AKAZE, ORB\}$;
Output: Metrics

```
foreach interplation approach in IA do
    foreach width in TW do
        foreach height in TH do
            foreach det in DET and des in DES do
                foreach S_i do
                    Calculate the value of aforementioned metrics;
                    Save the calculated value;
Analyze the metric values.
```

Keypoint Detectors and Descriptors for LiDAR-Generated Images

The evaluation workflow of detector-descriptor algorithms typically comprises three stages including feature extraction, keypoint description, and keypoint matching between successive image frames. In this section, the specific procedures for executing these stages in our experimental setup will be elaborated upon.

Designated Keypoint Detector and Descriptor
An extensive array of keypoint detectors and descriptors, as detailed in Table 15 from Section 5.2.1, were investigated. The employed keypoint detectors include SHITOMASI, HARRIS, FAST, BRISK, SIFT, SURF, AKAZE, and ORB. Additionally, we integrated Superpoint, a DL-based keypoint detector, into our methodology. The keypoint descriptors implemented in our experiment are BRISK, SIFT, SURF, BRIEF, FREAK, AKAZE, ORB.

Key Points Matching between Images
Keypoint matching, the final stage of the detector-descriptor workflow, focuses on correlating key points between two images, which is essential for establishing spatial relationships and forming a coherent scene understanding. The smaller the distance of the descriptors between two points, the more likely it is that they are the same point or object between two images. In our implementation, we employ a technique termed "brute-force match with cross check", which means for a given descriptor $\mathcal{D}_A$

in image A and another descriptor $\mathcal{D}_B$ in image B, a valid correspondence requires that both descriptors recognize each other as their closest descriptors.

Selected Evaluation Metrics

As explained in Section 5.2.2, we have opted not to rely on ground truth-based evaluation methodologies due to the lack of benchmark datasets and the substantial labor involved in data labeling. Instead, we combined some specially-designed metrics that are independent of ground truth, together with several intuitive metrics, to form the complete indicators listed in Table 16. To our best understanding, this represents the most extensive set of evaluation metrics currently available in the absence of a benchmark dataset.

Evaluation Process

The flowchart shown in Algorithm 5 below provides an outline of the steps carried out by the program. Two nested loops are employed to iterate over different detector-descriptor pairs. For each image, the algorithm detects and describes its keypoints. If more than one image has been processed, keypoints from the current image are matched to the previous one. And metrics are placed in corresponding positions to assess the algorithm's performance.

5.2.4 Keypoints Assisted Point Cloud Registration

Selected Data

The selected data for the evaluation from the dataset mentioned in Section 5.2.3 includes indoor and outdoor environments. The outdoor environment are from the normal road, denoted as *Open road*, and a forest, denoted as *Forest*. The indoor data include a hall in a building, denoted as *Hall (large)*, and two rooms, denoted as *Lab space (hard)*, and *Lab space (easy)*.

Point Cloud Matching Approach

We applied KISS-ICP[2] as our point cloud matching approach. It provides also the odometry information, affording us the means to assess the efficacy of our point cloud downsampling approach through an examination of a positioning error, namely translation error and rotation error. To generalize our proposed approach, we tested an NDT-based simple SLAM program[3] as well.

[2] `https://github.com/PRBonn/kiss-icp.git`

[3] `https://github.com/Kin-Zhang/simple_ndt_slam.git`

Algorithm 5: Overall evaluation pipeline of keypoint detectors and descriptors

Input:
N number of signal images: $\{\mathcal{S}^i\}, i \sim N$;
$DET =$
$\{SURF, SIFT, SHITOMASI, HARRIS, BRISK, FAST, AKAZE, ORB\}$;
$DES = \{FREAK, SIFT, BRISK, SURF, BRIEF, AKAZE, ORB\}$;

Output: Metrics: Number of keypoints, Robustness of Detector, Computational Efficiency, Match Ratio, Match Score, Distinctiveness

foreach $Detector \in DET$ **do**
  foreach $Descriptor \in DES$ **do**
    foreach $\mathcal{S}^i$ **do**
      $\mathcal{S}^i \longleftarrow Preprocess\left(\mathcal{S}^i\right)$;
      $\mathcal{KP}^i, \mathcal{D}^i \longleftarrow detect\_and\_compute\left(\mathcal{S}^i, Detector, Descriptor\right)$;
      $Running\ time\ of\ detection\ and\ descripton : \mathcal{T}_1^i$;
      $Number\ of\ Keypoints : \mathcal{N}_{kp}^i$;
      Apply different transformations to image S_i, then calculate robustness: $\mathcal{R}_{rot}^i$, $\mathcal{R}_{scale}^i, \mathcal{R}_{blur}^i$;
      if $i > 1$ **then**
        $\mathcal{V}_{match}^i \longleftarrow Match\left(D^i, D^{i-1},\right)$;
        $Running\ time\ of matching : \mathcal{T}_2^i$;
        $Computational\ Efficiency : \mathcal{T}_1^i + \mathcal{T}_2^i$;
        if $\mathcal{N}_{match}^i > 0$ **then**
          $Match\ Ratio = \mathcal{N}_{match}^i / \mathcal{N}_{kp}^i$;
          $\mathcal{KP}_{Ref}^i \longleftarrow find\_matched\_points\left(\mathcal{KP}^i, \mathcal{V}_{match}^i\right)$;
          $\mathcal{KP}_{Source}^i \longleftarrow find\_matched\_points\left(\mathcal{KP}^{i-1}, \mathcal{V}_{match}^i\right)$;
          $\mathcal{H}_{Estimated\_homo}^i \longleftarrow$
           $estimate\_homo\_matrix\left(\mathcal{KP}_{Source}^i, \mathcal{KP}_{Ref}^i\right)$;
          foreach $\mathcal{P}_{Ref} \in \mathcal{KP}_{Ref}^i$, $\mathcal{P}_{Source} \in \mathcal{KP}_{Source}^i$ **do**
            if $\|\mathcal{P}_{Source} - \mathcal{P}_{Ref} * \mathcal{H}_{Estimated\_homo}^i\| < 3$ **then**
              $\mathcal{N}_{good\_match}^i = +1$;
          $Match\ Score = \mathcal{N}_{good\_match}^i / \mathcal{N}_{match}^i$;
        else
          $Match\ Ratio = 0$;
          $Match\ Score = 0$;
        $\mathcal{V}_{Primary\_match}^i, \mathcal{V}_{Secondary\_match}^i \longleftarrow KnnMatch\left(D^i, D^{i-1}, k = 2\right)$;
        foreach $Match1 \in \mathcal{V}_{Primary\_match}^i$, $Match2 \in \mathcal{V}_{Secondary\_match}^i$ **do**
          if $Disctance_{match1} / Disctance_{match2} < 0.8$ **then**
            $\mathcal{N}_{Distinctive\_match}^i = +1$;
        $Distinctiveness = \mathcal{N}_{Distinctive\_match}^i / \mathcal{N}_{kp}^i$;
Analyze the metric values.

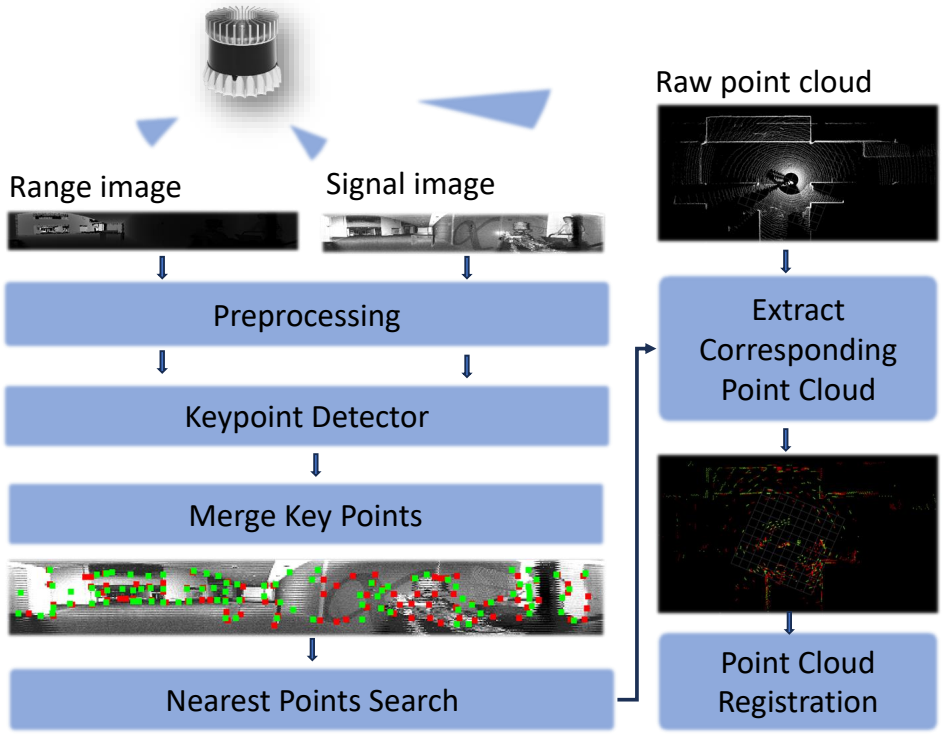


Figure 47: The process of the proposed LiDAR-generated images assisted point cloud registration

Proposed Method for Point Cloud Downsampling

Following the preprocessing of LiDAR-generated images outlined in Section 5.2.3, we derive optimal configurations for the keypoint detectors and descriptors. Utilizing these configurations as a foundation, we establish the workflow of our proposed methodology, illustrated in Figure 47. Within this process, we conduct distinct preprocessing procedures for both the range and signal images, employing them individually for keypoint detection and descriptor extraction. Subsequently, we combine the key points obtained from both images and search the K nearest points to each of these key points. We systematically varied K from 3 to 7, adhering to a maximum threshold of 7 to align with our primary objective of downsampling the point cloud. Consequently, we find the corresponding point cloud of the key points and their neighbors within the raw point cloud, thereby downsampling a point cloud.

In our analysis, we examined not only the positional error but also the rotational error, computational resource utilization, downsampling-induced alterations in point cloud density, and the publishing rate of LO.

5.2.5 Hardware and Software Information

Our experiments are run on the ROS Noetic on the Ubuntu 20.04 system. The platform is equipped with an i7 8-core 1.6 GHz CPU and an Nvidia GeForce MX150 graphics card. Primarily, we used libraries like OpenCV and PCL. Note, that we have used some non-free copyright-protected algorithms from OpenCV, such as SURF, just for research.

The assessment of keypoint-based point cloud downsampling was conducted on a Lenovo Legion notebook equipped with the following specifications: 16 GB RAM, a 6-core Intel i5-9300H processor (2.40 GHz), and an Nvidia GTX 1660Ti graphics card (boasting 1536 CUDA cores and 6 GB VRAM). Within this study, our primary focus was on the evaluation of the two open-source algorithms delineated in subsection 5.2.4, namely, KISS-ICP and Simple-NDT-SLAM. It is imperative to highlight that a consistent voxel size of 0.2 m was employed for both algorithms. Our project is primarily written in C++ (including the DL approach, Superpoint), publicly available in GitHub[4].

5.2.6 Evaluation Results

Through this section, we first cover the final results of our exploration of the preprocessing workflow of LiDAR-based images. Subsequently, an in-depth analysis of keypoint detectors and descriptors for LiDAR-based images is conducted. Then, a detailed quantitative assessment of the performance of LO facilitated by LiDAR-generated image keypoints is presented.

Results of preprocessing methods for LiDAR-generated image

Table 17: Evaluation metrics under different interpolation approaches.

Interpolation	Robustness of (rotation, scaling, noise)	Distinctiveness	Matching Score
AREA	(0.81,0.106,0.574)	0.309	0.415
CUBIC	(**0.82**,0.121,0.569)	0.292	0.408
LANCZOS4	(0.819,**0.127**,0.559)	0.286	0.405
NEAREST	(0.818,0.128,0.573)	0.275	0.401
LINEAR	(0.815,0.1,**0.583**)	**0.314**	**0.415**

As elucidated in Section 5.2.2, Distinctiveness and Match Score are considered as paramount measures for the overall accuracy of the entire algorithm pipeline. Consequently, in scenarios where different sizes and interpolation methods show peak performance on different metrics, these two metrics are our primary concern. Based on such an criteria, the size **1024×64** demonstrated better performance across all

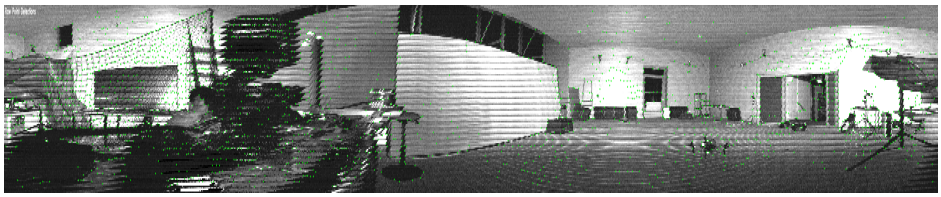
[4] https://github.com/TIERS/ws-lidar-as-camera-odom

dectectors and descriptors methods. Then in Table 17, our evaluation also revealed that the **linear** interpolation method yielded the optimal results among the various interpolation techniques.

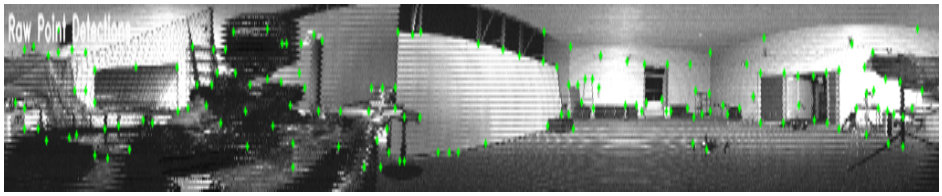
Table 18: Evaluation metrics under different resized resolutions.

Size	Robustness of (rotation, scaling, noise)	Distinctiveness	Matching Score
512×32	(**0.856**,**0.157**,0.551)	0.267	0.366
896×128	(0.827,0.156,0.591)	0.37	0.515
896×256	(0.843,0.146,**0.663**)	0.309	0.486
1024×64	(0.809,0.124,0.54)	**0.427**	**0.53**
1024×128	(0.832,0.147,0.584)	0.372	0.504
1024×256	(0.851,0.134,0.659)	0.32	0.483
1280×64	(0.798,0.116,0.527)	0.41	0.5
1280×128	(0.823,0.138,0.575)	0.353	0.479
1280×256	(0.849,0.127,0.652)	0.301	0.464
1920×128	(0.808,0.124,0.553)	0.321	0.436
1920×256	(0.844,0.114,0.644)	0.274	0.43
2048×128	(0.799,0.129,0.544)	0.309	0.425
2048×256	(0.837,0.119,0.633)	0.263	0.421
2560×128	(0.802,0.111,0.547)	0.294	0.4
2560×256	(0.842,0.106,0.644)	0.249	0.401
4096×128	(0.799,0.087,0.557)	0.257	0.339

The findings in Table 18, also suggest that there is a clear advantage in properly reducing the size of an image as opposed to enlarging it. Additionally, in the process of image downscaling, one pixel often corresponds to several pixels in original image. So overly downscaled images might lead to substantial deviations in the detected key points when re-projected to their original poistions, suggesting that extreme image size reductions should be avoided.



(a) Detect key points in an enlarged image.



(b) Detect key points in a downscaled image.

Figure 48: Keypoint detected in the resized signal images

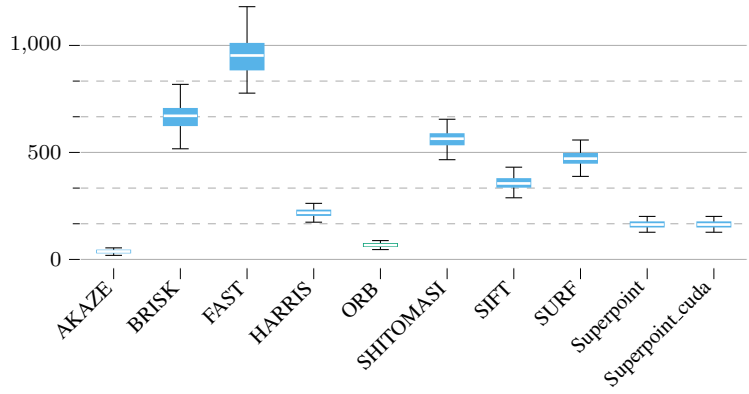


Figure 49: Number of key points

And here is a more intuitive result to show that how reducing the size of a image is far better than enlarging it. In Figure 48a and Figure 48b, Superpoint detectors identify keypoints as green dots. The enlarged image Figure 48a displays many disorganized points. Conversely, the downscaled image Figure 48a, reveals distinct keypoints, such as room corners and the points where various planes of objects meet. Note that we resized the two images for paper readability, originally, their sizes varied.

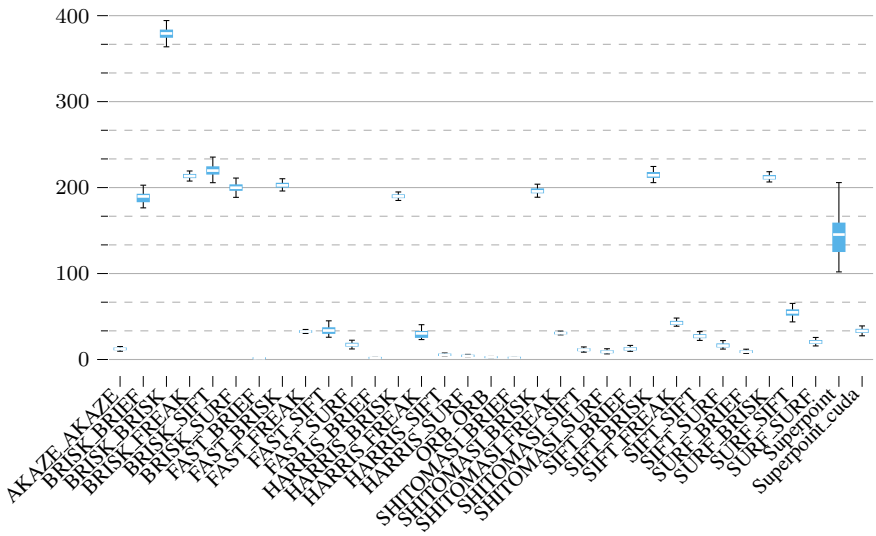
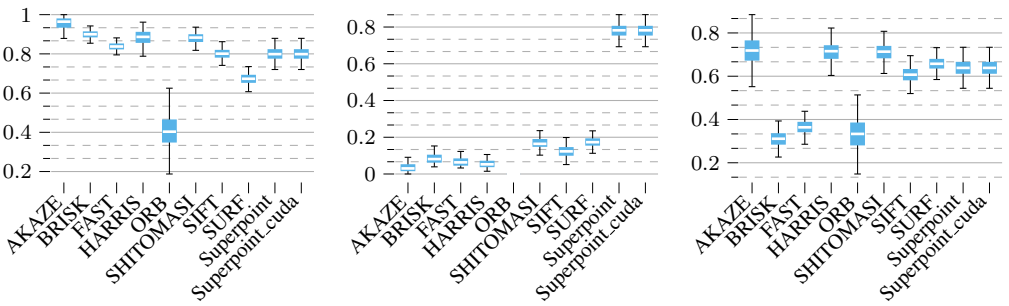


Figure 50: Computational efficiency



(a) Rotation (b) Scaling (c) Noise Interference

Figure 51: Robustness of the detector

Results of Keypoint Detectors and Descriptors For LiDAR Image

In Figure 49, which is a metric that only related to detectors, FAST and BRISK algorithms detected the highest number of keypoints, but there were significant fluctuations in the counts. Comparatively, AKAZE, ORB, and Superpoint identified a reduced number of keypoints, but the consistency was not stable.

Figure 50 depicts the Computational Efficiency, where the majority of the algorithms operate in less than 50 ms. After CUDA enabled, SuperPoint runs significantly faster with minimal variance. Among all algorithms, BRISK is the most time-consuming, just using BRISK solely as a descriptor with other detectors will hinder the overall efficiency.

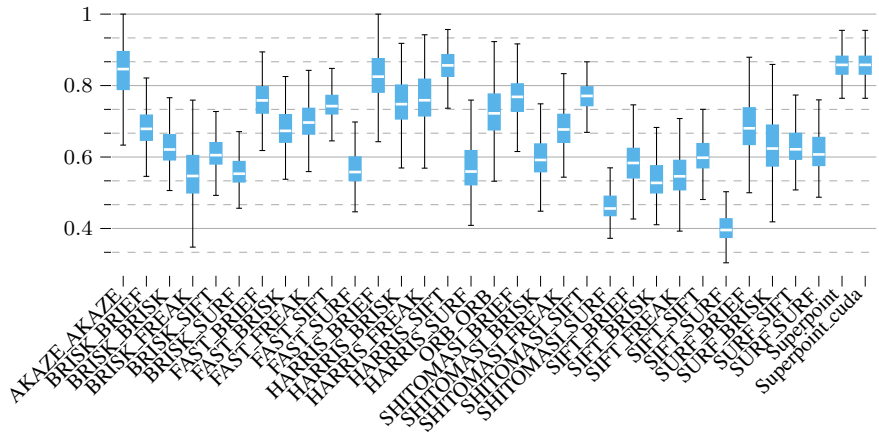


Figure 52: Match ratio

Figure 51 shows the Robustness of Detector. Superpoint consistently demonstrates robust performance across various transformations. Among conventional de-

tectors, AKAZE has proven effective, especially in handling rotated transformations and noise interference. And most detectors, exhibit marked poor performance under scale invariance. The horizontal textures inherent in LiDAR-based images might explain such weakness: when the images are enlarged, these textures can be erroneously detected as keypoints.

As emphasized in Section 5.2.2, a multitude of keypoint detections and rapid matches could be useless if their accuracy is not guaranteed. Therefore, Match Ratio, Match Score, and Dinctiveness, which pertain to algorithmic accuracy, can be regarded as the most pivotal indicators across various application perspectives. Figure 52, Figure 53, and Figure 54 present the results of these three metrics, indicating that Superpoint, when augmented with CUDA, is the most effective solution. Moreover, among traditional algorithms, AKAZE demonstrates top-tier performance across the majority of evaluated metrics, making it a commendable choice.

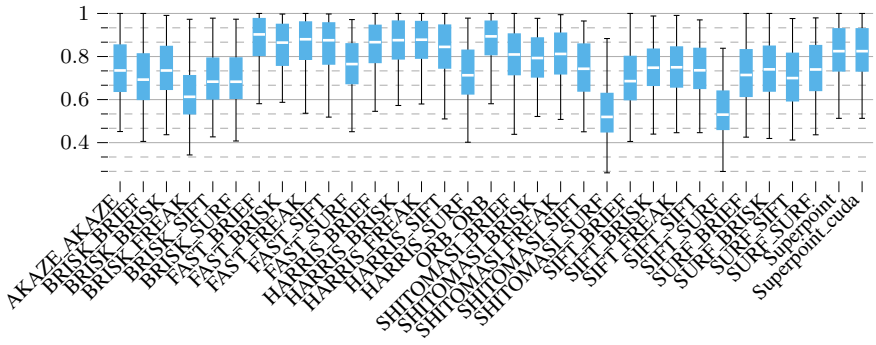


Figure 53: Match score

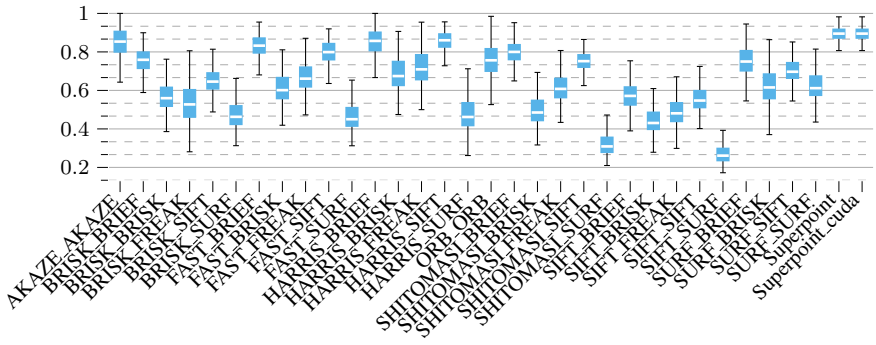
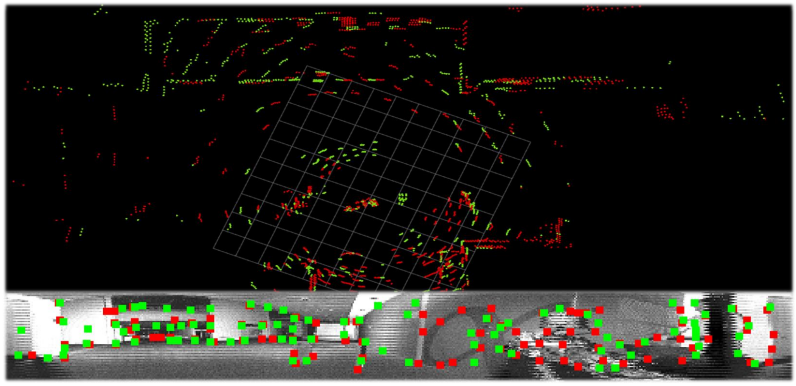


Figure 54: Distinctiveness

Results of LiDAR-generated Image Keypoints Assisted LO

Downsampled Point Cloud

In Figure 55, we demonstrate the sample result of the downsampled point cloud in Figure 55a compared with the raw point cloud in Figure 55b. Notably, in the downsampled point cloud in Figure 55a, the red points are extracted based on signal images and the green ones are from range images. We draw the key points from both images to the signal image shown in the lower part of Figure 55a. The disparity between the points extracted based on these two types of images shows the significance of different LiDAR-generated images. Additionally, in the preliminary evaluation of LO, we found the accuracy of LO is lower if we only integrated the signal images instead of both modalities. This encourages us to utilize both signal and range images in the latter part.



(a) The upper part presents the downsampled point cloud from our lidar-based method, while the bottom illustrates key point distribution in the signal image: red from the signal image and green from the distance image.

(b) Raw point cloud and signal image.

Figure 55: Samples of point cloud data

Table 19: Performance evaluation of LO (KISS-ICP) with raw point cloud and our downsampled point cloud, '*Sig*' and '*Rng*' represent the size of neighboring point areas for the signal and range images, respectively, denoted as *Sig_Rng*.

Neighbor Size (*Sig_Rng*)	**Open road**	**Forest**	**Lab space (hard)**	**Lab space (easy)**	**Hall (large)**
		(Translation error (mean/rmse)(m), rotation error(deg))			
4_4	N / A	(0.079/0.090, 6.58)	(0.052/0.062, **1.44**)	(0.027/0.031, 0.99)	(1.111/1.274, 3.37)
4_5	N / A	(0.086/0.096, 7.22)	(0.043/0.051, 1.51)	(0.031/0.035, 1.05)	(0.724/0.819, 2.95)
4_7	(**0.817/0.952**, 2.33)	(0.082/0.102, 7.78)	(0.039/0.046, 1.46)	(0.028/0.033, 0.98)	(**0.583/0.660**, 2.88)
5_4	(1.724/2.038, 2.10)	(0.085/0.100, 6.81)	(0.059/0.070, 1.71)	(0.025/0.028, 0.98)	(1.065/1.242, 2.73)
5_5	(2.176/2.410, **1.76**)	(0.108/0.203, 6.96)	(0.037/0.043, 1.35)	(0.028/0.032, 0.97)	(0.707/0.801, **2.66**)
5_7	(1.298/1.443, 2.71)	(0.076/0.084, **6.11**)	(0.064/0.075, 1.54)	(0.025/0.028, 0.94)	(0.676/0.746, 3.67)
7_4	(1.696/1.888, 2.31)	(0.082/0.094, 6.98)	(0.074/0.085, 1.64)	(0.027/0.032, 0.99)	(0.806/0.917, 3.68)
7_5	(1.784/2.006, 2.30)	(0.080/0.102, 7.72)	(**0.033/0.047**, 1.59)	(0.025/0.028, 0.97)	(0.698/0.803, 3.11)
Raw PC	N / A	(**0.057/0.073**, 8.91)	N / A	(**0.020/0.022**, **0.62**	N / A)

LO-based Evaluation

In our experiment, various numbers of neighbor points are utilized, ranging from 3 to 7 for each type of LiDAR-generated image. We selected part of them to show the result here based on the principle that more accurate but less amount points. As we found in the previous section, the Superpoint has reliable key points detected, so we utilize this DL method to extract key points in our proposed approach while KISS-ICP is the point cloud registration and LO method. Table 19 shows the performance of LO based on different sizes of neighbor point sizes in both indoor (Lab space, Hall) and outdoor (Open road and Forest) environments.

As shown in Table 19, in the scenarios of Open road, Lab space (hard), and Hall(Large), the LO from KISS-ICP applying raw point cloud can not work properly with large drift which the error can not be calculated. Meanwhile, our proposed approach works all the time. Additionally, even when applying raw point cloud to KISS-ICP works, our approach can achieve comparable translation state estimation while more robust in the rotation state estimation across most of the situations.

Table 20: The number of points left after downsampling with varied neighbor size, '*Sig*' and '*Rng*' represent the size of neighboring point areas for the signal and range images, respectively, denoted as *Sig_Rng*.

Neighbor Size (*Sig_Rng*)	**Open road**	**Forest**	**Lab space (hard)**	**Lab space (easy)**	**Hall (large)**
			Number of points(pts)		
4_4	2650	7787	6435	6360	4742
4_5	3206	7812	6416	6302	4792
4_7	4784	11447	9518	9392	7094
5_4	3182	7843	6409	6333	4776
5_5	3183	7568	6446	6292	4783
5_7	4763	11519	9513	9386	7066
7_4	4760	11631	9445	9356	7070
7_5	4756	11627	9469	9378	7078
Raw PC	131072	131072	131072	131072	131072

In outdoor settings, a neighbor size 4_7 (4×4 for signal images and 7×7 for range images) exhibits notable efficacy in both translation and rotation state estimation. Conversely, in indoor environments, a neighbor size 5_5 (5×5 for signal images and 5×5 for range images) demonstrates commendable performance in the estimation of translation and rotation states, in addition to exhibiting efficient downsampling capabilities, as delineated in Table 20.

Based on the above result, we apply the neighbor size 4_7 for outdoor settings and the neighbor size 5_5 for indoor settings to further extend the performance evaluation by including the conventional keypoint detector approach and another point cloud matching approach, NDT. It is worth noting that the purpose of applying NDT here is not to compare with KISS-ICP but to show the generalization of our proposed approach among other point cloud registration methods.

Table 21: Evaluation of LO based on conventional and DL keypoint detectors with KISS-ICP

Approaches / Evaluation Indicators	KISS-ICP					
	Outdoor			Indoor		
	AKAZE	**Superpoint**	**Raw PC**	**AKAZE**	**Superpoint**	**Raw PC**
Translation Error *(m)*	0.096/0.107	**0.082/0.102**	N / A	0.092/0.099	**0.037/0.043**	N / A
Rotation error *(deg)*	**4.27**	7.78	N / A	**1.22**	1.71	N / A
CPU *(%)*	**263.16**	457.59	544.61	**82.57**	425.93	572.50
Mem *(MB)*	247.62	308.67	**165.73**	198.27	232.85	**84.37**
Avg Pts	**4849**	11447	131072	**1176**	6446	131072
Odom Rate *(Hz)*	**10.0**	4.0	2.95	**10.0**	7.6	2.82

The result in Table 21 and Table 22 proves that the conventional keypoint extractor can achieve comparable LO translation estimation and more accurate rotation estimation to Superpoint. This performance is obtained with much less CPU and memory utilization, and fewer cloud points, but higher odometry publishing rates. Similar results are achieved by NDT based approach which validates the above result in a certain way. Notably, the memory consumption using KISS-ICP with raw point cloud in Table 21 is lower than others. As our observation indicates, the primary reason behind this is the drift, resulting in few points for the point cloud registration.

Table 22: Evaluation of LO based on conventional and DL keypoint detectors with NDT

Approaches / Evaluation Indicators	NDT					
	Outdoor			Indoor		
	AKAZE	**Superpoint**	**Raw PC**	**AKAZE**	**Superpoint**	**Raw PC**
Translation Error *(m)*	0.115/0.126	**0.090/0.098**	N / A	0.102/0.114	**0.054/0.071**	N / A
Rotation error *(deg)*	**4.84**	5.66	N / A	**1.15**	1.31	N / A
CPU *(%)*	**100.39**	325.69	571.12	**82.20**	338.54	581.30
Mem *(MB)*	**285.06**	548.16	705.43	**253.95**	290.34	645.21
Avg Pts	**4849**	11447	131072	**1176**	6446	131072
Odom Rate *(Hz)*	**10.0**	4.6	3.34	**10**	8.21	1.20

5.3 LiDAR Generated Images Enhanced UAV Tracking

LiDAR-as-camera approach can augment conventional LiDAR-based methods for object detection and tracking, as the algorithms for the latter are more computationally intensive and less mature than vision sensors. Despite the low vertical resolution and lack of color, the key motivation for considering such an approach is simply to do more with already existing data and without additional sensors.

We are interested in studying the potential of these sensors for tracking UAVs, fusing both LiDAR point cloud data and signal images. Given that the images generated by the LiDAR as a camera sensors have low resolution and the farther objects have less points (see for example in Figure 59), we are particularly interested in short-range tracking of UAVs, which finds potential applications in UAV docking, or collaboration, among others. The deployment of UAVs requires accurate localization, especially where GNSS signals are degraded or not available [200].

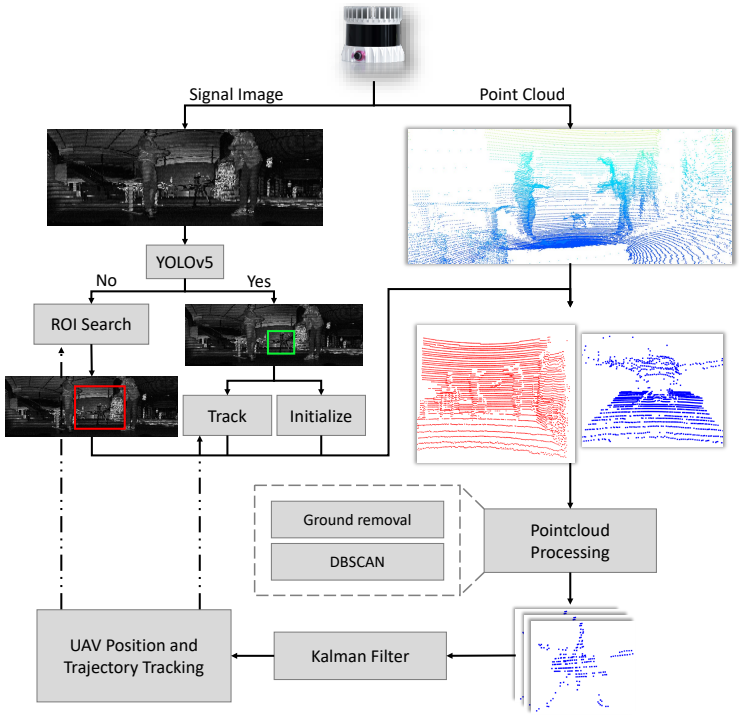


Figure 56: Diagram of proposed UAV tracking system based on the image and point cloud generated by an Ouster LiDAR.

In this dissertation, we propose a UAV tracking approach based on the integration of images and 3D point clouds generated by an Ouster LiDAR sensor. The diagram of the approach we follow is illustrated in Figure 56. The UAV can be detected in

signal images instead of manually giving its initial position as it is needed in other point-cloud-only approaches. The detection also yields an approximate region of interest (ROI) in the point cloud, which will be expanded if no detection occurs. This approach reduces computation overhead by avoiding the need for an overall point cloud search. UAV identification is achieved by clustering points within the ROI, followed by continuous position estimation using the Kalman Filter (KF).

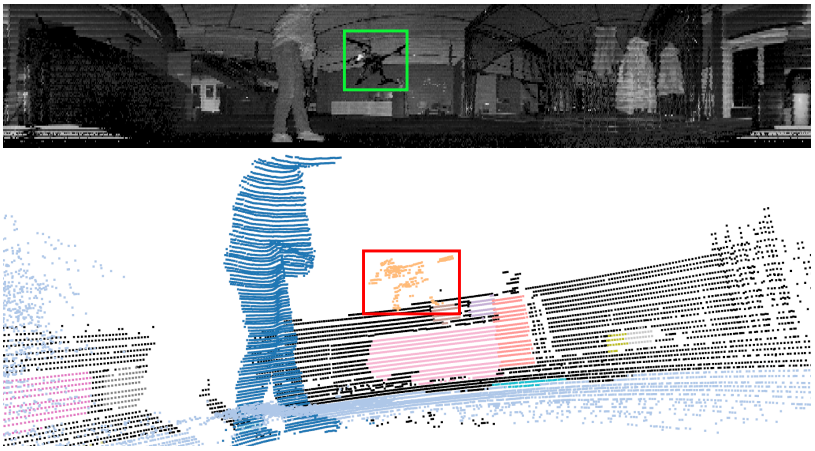


Figure 57: Example of a signal image (top) and its corresponding point cloud with background removed (bottom).

5.3.1 Initialization of UAV Position

Figure 57 shows the signal image and point cloud data of a UAV at its initial position. Notably, when the UAV approaches the Ouster LiDAR, the signal image of the UAV appears clearer. To detect the position of the UAV in the signal image and obtain the ROI in the image, the state-of-the-art object detection algorithm YOLOV5 is utilized. Given that the Ouster LiDAR signal image and the point cloud data are spatially linked, the corresponding point cloud ROI can be extracted. Subsequently, by employing ground removal and point cloud clustering techniques, the UAV point cloud can be extracted and, as such, the initial position of the UAV can be estimated.

5.3.2 Fusion of LiDAR Generated Images and Point Cloud

Our goal for fusing the signal image and point cloud data is to acquire an accurate ROI. Initially, we perform image detection on each signal image to identify the UAV. This allows us to obtain a more precise ROI. Additionally, we can use the UAV's initial position from the previous step as a reference to extract the UAV point cloud from the environment based on the number of UAV point clouds and their distance.

However, if YOLOv5 fails to detect the UAV, we select the ROI predicted by the FK and separate the UAV point cloud from it. This process is explicated in Algorithm 6.

Algorithm 6: UAV tracking fusing signal images and point clouds

Input:

Raw pointcloud: $\mathcal{P}_{raw}^{t}$

Signal image: $\mathcal{S}^{t}$

Target UAV point cloud: $\mathcal{P}_{UAV}^{t-1}$

Output:

Drone pose: $\mathbf{P}_{UAV}^{t}$;

Function `object_extraction` ($\mathcal{P}_{raw}^{t}$, $\mathcal{P}_{UAV}^{t-1}$, $\mathcal{ROI}_{YOLO}^{t}$)**:**
  if $\mathcal{ROI}_{YOLO}^{t}$ **then**
    $\mathcal{P}_{roi}^{t}$ = $\mathcal{P}_{raw}^{t}$ ($\mathcal{ROI}_{YOLO}^{t}$);
  else
    $\mathcal{ROI}^{t} \longleftarrow KF\left(get\_center\left(\mathcal{P}_{UAV}^{t-1}\right)\right)$;
  $\mathcal{P}^{t} \longleftarrow ground\_removal\left(\mathcal{P}_{ROI}^{t}\right)$;
  $\mathcal{P}_{i}^{t} \longleftarrow DBSCAN\left(\mathcal{P}^{t}\right),\ i \in (0, R)$;
  foreach $\mathcal{P} \in \mathcal{P}_{i}^{t}$ **do**
    if *Min (num($\mathcal{P}$) - num($\mathcal{P}_{UAV}^{t-1}$))* **then**
      if *Min (dis($\mathcal{P}$) - dis($\mathcal{P}_{UAV}^{t-1}$))* **then**
        $flag = 1$;
        $\mathcal{P}_{uav}^{t} \longleftarrow \mathcal{P}$;
      else
        $flag = 0$;
    else
      $flag = 0$;
  return $\mathcal{P}_{uav}^{t}$, flag ;
foreach *new* $\mathcal{S}^{t}$ **do**
  $\mathcal{ROI}_{YOLO}^{t} \longleftarrow YOLOv5\left(\mathcal{S}^{t}\right)$;
  if $\mathcal{ROI}_{YOLO}^{t} =$ ***None*** **then**
    $\mathcal{P}_{UAV}^{t}$, flag = *object_extraction* ($\mathcal{P}_{raw}^{t}$, $\mathcal{P}_{uav}^{t-1}$) ;
    if *flag = 0* **then**
      $\mathbf{P}_{UAV}^{t}$ = *KF_predict* (*get_center*($\mathcal{P}_{UAV}^{t-1}$)) ;
      KF_update ($\mathbf{P}_{UAV}^{t}$)
    else
      $\mathbf{P}_{UAV}^{t}$ = *get_center*($\mathcal{P}_{UAV}^{t}$) ;
      KF_update ($\mathbf{P}_{UAV}^{t}$) ;
  else
    $\mathcal{P}_{UAV}^{t} = object\_extraction\left(\mathcal{P}_{raw}^{t},\ \mathcal{P}_{UAV}^{t-1},\ \mathcal{ROI}^{t}\right)$;
    $\mathbf{P}_{UAV}^{t} = get\_center\left(\mathcal{P}_{UAV}^{t}\right)$;
    $KF\_update\left(\mathbf{P}_{UAV}^{t}\right)$;

5.3.3 Evaluation Setup

Hardware Information

The experimental setup consists of an Ouster OS0-128 LiDAR, an Intel computer, and a Holybro X500 V2 drone shown in Figure 58. The Ouster OS0-128 boasts a wide FoV ($360° \times 90°$) and is capable of producing both dense point cloud data and signal images at a frequency of 10 Hz. The drone is equipped with OptiTrack markers, allowing the acquisition of the drone's actual position at 100 Hz in the motion capture (MOCAP) system, which is also partially visible in Figure 58.

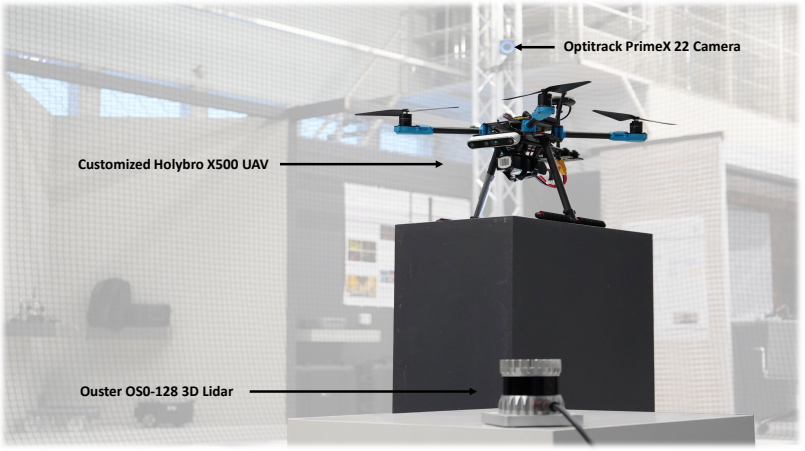


Figure 58: Experimental hardware and site.

Software Information

The system was implemented based on the ROS Noetic framework on Ubuntu 20.04 operating system. The tracking package, Ouster drivers, and OptiTrack mocap program were executed on the laptop computer connected to the Ouster LiDAR. Our tracking approach requires YOLOv5[5] for UAV detection and Open3D[6] for point cloud data processing. The algorithms and code designed and developed for these experiments are written in Python and are publicly available in a GitHub repository[7].

[5] https://github.com/ultralytics/yolov5/releases

[6] http://www.open3d.org/

[7] https://github.com/TIERS/UAV-tracking-based-on-LiDAR-as-a-camera

Data collection

Regarding the data consisting of UAV detections with the Ouster LiDAR, we collected three different data sequences ($Seq \quad i, i \in (1 \sim 3)$). in an indoor area of $10.0 \times 10.0m^2$, with distances ranging 0.5 m to 8 m between the LiDAR and the UAV. The details of the collected data can be seen in Table 23. *Seq 1* and *Seq 3* represent a helical ascension trajectory, while *Seq 2* represents an elliptical trajectory.

Table 23: Data sequences collected in our experiment.

Sequences	Time (s)	Ground Truth	Trajectory	Distance (m)
Seq 1	35.8	MOCAP	elliptical trajectory	7.0
Seq 2	26.9	MOCAP	spiral trajectory	6.3
Seq 3	32.7	MOCAP	spiral trajectory	8.0

Evaluation

To validate the precision of the UAV estimated poses and velocities by our approach, we calculate the absolute pose error (APE) and velocity error based on the ground truth from the MOCAP system. We conducted a comparative analysis of our proposed method with a UAV tracking method that solely relies on either Ouster LiDAR images or point clouds. The point cloud tracking method uses only Ouster OS0-128 LiDAR point cloud data as input, with a frame rate of 10HZ. When tracking the UAV using point cloud data, the initial position of the UAV needs to be known as the point cloud of the UAV is sparser than that of larger objects, such as cars or humans, and distinguishing the point cloud of the UAV from the environment using features is challenging. On the other hand, the image tracking method uses only Ouster OS0-128 LiDAR signal images with a frame rate of 10 Hz. Firstly, the signal image undergoes target detection processing to obtain the UAV's bounding box in the signal image. Subsequently, the image in the bounding box is converted into point cloud data. The point cloud clustering algorithm is then utilized to separate the UAV's point cloud from the environment based on the number and distance features of the point cloud clusters. This approach allows us to obtain the trajectory of the UAV. Both of these methods are estimated by the Kalman filter method to obtain the UAV's trajectory.

We conducted the experiments on two different platforms to assess real-time performance, the Lenovo Legion Y7000P equipped with 16GB RAM, 6-core Intel i5-9300H (2.40 GHz) and Nvidia GTX 1660Ti (1536 CUDA cores, 6GB VRAM), as well as the commonly used embedded computing platform Jetson Nano with 4-core ARM A57 64-bit CPU (1.43 GHz), 4 GB RAM, and 128-core Maxwell GPU.

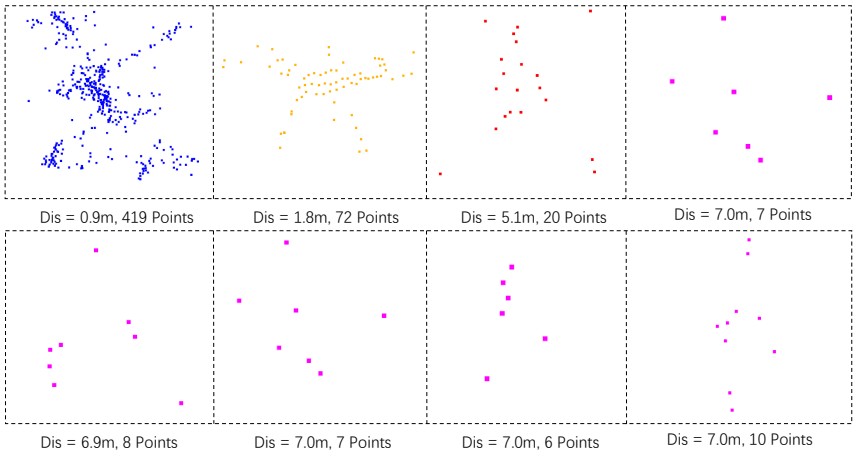


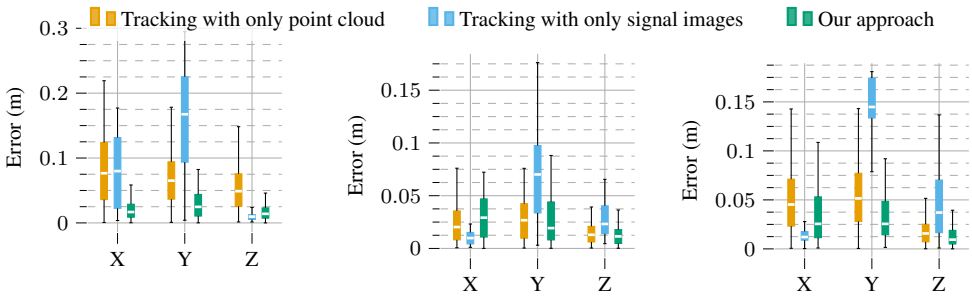
Figure 59: UAV point cloud of drones at different distances, the bottom line shows UAV point cloud of four consecutive frames at the same long distance.

5.3.4 UAV Tracking Result Evaluation

In this section, we report the experimental results, based on the three data sequences gathered in the indoor test environment.

UAV in the Ouster LiDAR point cloud

The first parameter to analyze is the number of points that reflect from the UAV at different distances. Our analysis, shown in Figure 59, reveals that the point cloud structure generated by the UAV is significantly influenced by the distance from the target. At short distances, the point cloud produced by LiDAR is abundant and presents comprehensive details. When the distance extends to a medium range, the number of UAV point clouds decreases to less than 100, but the three-dimensional structure of the UAV remains discernible. However, when the distance is at a medium range of 7 m according to our results, the number of UAV point clouds reduces to single digits, and the point cloud structure becomes highly unpredictable and unstructured. It is worth noting that in a more realistic application, additional elements such as other sensor payloads or a cargo bay would potentially increase significantly the reflective surface of the UAV.



(a) Seq 1 (b) Seq 2 (c) Seq 3

Figure 60: Absolute position error (APE) value of three data sequences.

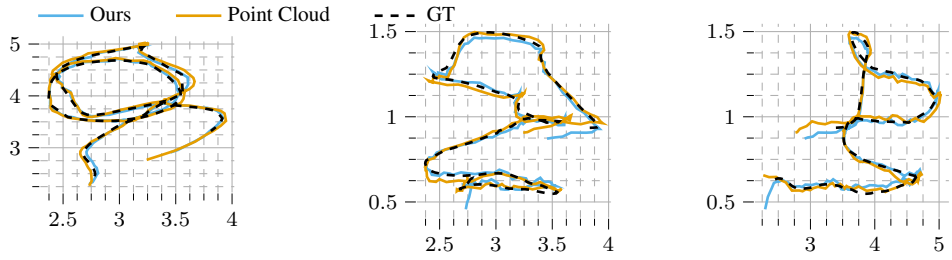
Trajectory Validation

We also show a quantitative analysis of the APE based on ground truth, with the main results summarized in Figure 60. To ensure the trajectories are compared under the same coordinates, we utilize the coordinates of the ground truth as the reference coordinates and convert all trajectories generated by the three UAV tracking methods to these coordinates. Table 24 presents a comprehensive comparison of three different UAV tracking methods in terms of detectable distance, average APE, algorithm update frequency, and need for initial conditions.

Table 24: Performance(Detectable distance, frame rate, and APE error) and initial conditions comparison of selected tracking methods.

	Distance *(m)*	**APE Error(Mean/RMSE)** (m)	**FPS** (Hz)	**Initialization**
Tracking with point cloud	**8.0**	0.104 / 0.142	8.3	Yes
Tracking with signal images	2.4	0.078 / 0.088	**10**	No
Fused(Ours)	**8.0**	**0.061 / 0.067**	**10**	No

The image-based UAV tracking method shows a relatively small average error; however, its overall error distribution is inconsistent, as the Y-axis error in the *Seq 1* sequence reaches up to 0.3 m. Conversely, the point cloud-based UAV tracking method has the largest average error, but its error distribution is more uniform. Our proposed method, on the other hand, achieves the smallest average error and minimal error fluctuation. Additionally, to supplement the quantitative trajectory analysis, we also provide a visualization of the trajectories based on the point cloud tracking method and our proposed method from three different viewpoints, as illustrated in Figure 61, with more consistent behavior.

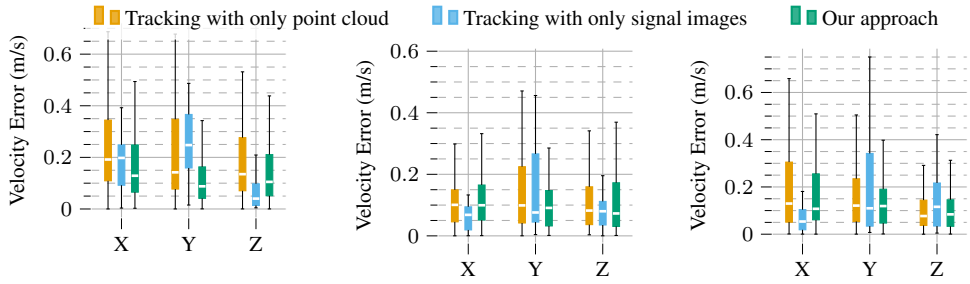


(a) XY trajectory plot (b) XZ trajectory plot (c) YZ trajectory plot

Figure 61: Comparison of estimated trajectories with the point cloud tracking method and our proposed method from three different projections.

Velocity Validation

In addition to pose estimation, we conducted a quantitative analysis of the UAV velocities based on the ground truth data and compared them with different UAV tracking methods. Figure 62 illustrates the velocity errors of each method along the X, Y, and Z axes. The experimental results reveal that all methods have similar mean values of the velocity errors, but different fluctuations. The image tracking method has a large fluctuation in the Y-axis velocity error, reaching up to *0.75 m/s* in *Seq 3*. The point cloud tracking method also has relatively large fluctuations in all dimensions. In contrast, our method achieves smaller overall velocity errors in both the mean value and fluctuation range.



(a) Seq 1 (b) Seq 2 (c) Seq 3

Figure 62: Velocity estimation error for each linear component in the three data sequences.

Resource Consumption

Both the Intel laptop and the Jetson Nano run ROS Melodic on Ubuntu 18.04. The CPU and memory utilization is measured with a ROS resource monitor tool[8]. Additionally, for minimizing the difference of the operating environment, we unified the dependencies used in each method into same version. The results are summarized in Table 25.

Table 25: Average run-time resource (CPU/RAM) utilization and performance (pose calculation speed) comparison of selected tracking methods across multiple platforms. CPU utilization of 100% equals one full processor core.

	Laptop	Jetson Nano
	(CPU (%), RAM (MB), Pose rate (Hz))	
Tracking with point cloud	(422.7, 296.1, 8.3)	(121.7, 179.7, 5.15)
Tracking with image	(209.0, 293.6, 10)	(113.8, 232.9, 6.07)
Fused (ours)	(195.5, 299.0, 10)	(114.5, 247.8, 6.04)

The memory utilization of each selected method was roughly equivalent in both processor architecture platforms. However, the same algorithm showed generally higher CPU utilization and achieved the highest publishing rate when running on the Intel processor. For the Intel processor, the point cloud tracking method had higher CPU utilization than other methods but the lowest publishing rate. The fusion method performed well on the laptop and had the smallest overall error. On the embedded computing platform, the CPU utilization of all methods did not differ significantly, and the point cloud tracking method had the lowest memory utilization but the lowest pose publication rate. The difference in CPU utilization is caused by the use of CUDA GPU acceleration in the Open3D binaries utilized for the Jetson Nano platform, while the Intel computer uses only the CPU for point cloud data processing. The image processing also leverages the embedded GPU in the Jetson Nano board. Because of the small ROI that we extract to process the point cloud, the fused method adds little overhead on top of the vision-only method.

5.4 Summary

This section is on utilizing LiDAR as a camera to enhancing various robotic perception tasks. First, we have presented an analysis of the performance of different object detection and semantic segmentation DL models on images generated by LiDAR sensors. We have collected data with two different LiDARs indoors and outdoors and in both daylight and night scenes. Our experiments show that state-of-the-art

[8] https://github.com/alspitz/cpu_monitor

DL models can process this type of data with a promising performance by interpolating the low-resolution images to adequate resolutions. Object segmentation results are particularly optimistic, therefore paving the path for further usage of LiDARs beyond the current algorithms focused on odometry, localization, mapping, and object detection from geometric methods. The main limitation of the current analysis is perhaps the lack of re-training for the models with larger datasets of LiDAR-generated images, owing to the lack of such annotated datasets.

Secondly, to mitigate computational overhead while ensuring the retention of a sufficient number of dependable key points for point cloud registration in LO, this study introduces a novel approach that incorporates LiDAR-generated images. A comprehensive analysis of keypoint detection and descriptors, originally designed for conventional images, is conducted on the LiDAR-generated image. This not only informs subsequent sections of this paper but also sets the stage for future research endeavors aimed at enhancing the robustness and resilience of LO and SLAM technology. Building upon the insights gleaned from this analysis, we propose a methodology for down-sampling the raw point cloud while preserving the integrity of salient points. Our experiments demonstrate that our proposed approach exhibits comparable performance to utilizing the complete raw point cloud and, notably, surpasses it in scenarios where the full raw point cloud proves ineffective, such as in cases of drift. Additionally, our approach exhibits commendable robustness in the face of rotational transformations. The computation overhead of our approach is lower than the LO utilizing raw point cloud but with a higher odometry publishing rate.

In the end, this work has proposed a novel approach for tracking a UAV based on the fusion of signal images and point clouds from an Ouster LiDAR. Unlike conventional LiDAR and camera fusion, this approach does not need any calibration and preprocessing with external cameras and the LiDAR data is more resistant to harsh environments. We collected three different data sequences in an indoor environment with the OptiTrack MOCAP system providing ground truth positions. We compared the proposed approach with the approaches based on either only point clouds or signal images and the results showed the effectiveness of our proposed approach. Additionally, we found that our approach can be utilized in a popular mobile computing platform, Jetson Nano according to our evaluation.

6 UAV Tracking with a Solid-State LiDAR

Even though rotating LiDARs nowadays with 64 or 128 vertical channels can provide high angular resolution in both horizontal and vertical dimensions, these high-end devices are not the most common. Moreover, the scanning pattern is in general repetitive, which has benefits from a geometric perspective in terms of data processing but does not enable a higher FoV coverage with longer exposure if the position of the sensor is fixed. In previous sections, we discussed the benefits and challenges of solid-state LiDAR, including its long-range scanning technology that produces high-density point clouds, making it ideal for tracking objects in three-dimensional space such as UAVs [7]. With non-repetitive scan patterns as that shown in Figure 63a, solid-state LiDARs can generate more dense point clouds with adjustable frequencies and varying FoV. In particular, we are interested in the possibilities of dynamically adjusting the FoV coverage and density in the point cloud to be processed for detection and tracking. Among the benefits of these new LiDARs and the possibilities of adaptive scanning rates is also higher resilience against one of the challenges in LiDAR-based perception: motion-induced distortion [254].

UAVs are increasingly used in various applications due to their mobility and ease of deployment [255; 256], and, equipped with only a flight controller and a basic sensor suite, they serve as efficient mobile sensing platforms [257; 258]. Recent research has focused on UAV navigation in GNSS-denied environments [259; 135; 260] and state estimation in both single and multi-UAV systems [203; 101].

In multi-robot systems, tracking UAVs or MAVs from UGVs enhances flexibility and reduces the need for high-accuracy onboard localization. A significant example of multi-robot system deployment in GNSS-denied environments is the DARPA Subterranean Challenge [126; 127], where localization and collaborative sensing were critical challenges. During the challenge, MAVs were dynamically deployed from UGVs. Since MAVs often rely on VIO for self and relative state estimation, external LiDAR-based tracking can extend operability to low-visibility conditions or other scenarios where VIO faces limitations [100; 261].

The work presented in Section 6.1 introduces adaptive scan integration methods for tracking a UAV from a UAV station. To the best of our knowledge, this approach has not been previously explored. Due to the unavailability of a MOCAP system to verify the tracking results at the time of this study, we have developed our own validation method. And in the later work in Section 6.2, we improve the previous

idea in both data processing (real-time adaptive dynamic integration) and a tracking algorithm using KF with ground truth provided by MOCAP system.

The main contribution of this work is as follows:

1. We first introduce a novel adaptive LiDAR scan integration method enabling more accurate and reliable UAV tracking from 3D point clouds, specifically applied to MAV detection. (in Section 6.1)

2. In addition, we define a multi-modal tracking system to process point clouds resulting in different integration times for higher accuracy and persistent tracking, while validating the trajectories using a priori known information about the MAV dimensions. (in Section 6.1)

   (It is worth noting that the aforementioned studies on UAV tracking primarily focuses on iterative detections (tracking by detection) conducted offline and in post-processing.)

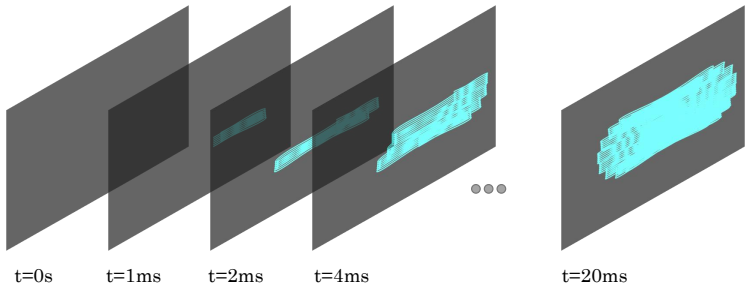
3. Unlike the above work, we propose an algorithm that adjusts the LiDAR frame integration time, or frequency, dynamically based on the UAV speed and distance from the sensor. We integrate consecutive scans in a sliding window manner to retain the most recent information about the state of the UAV. (in Section 6.2)

4. We develop a novel dual tracking approach using a Kalman filter variant that combines the two scan integration frequencies into one single state estimation using inverse covariance intersection. We evaluate the tracking performance during the experiments with ground truth data generated by a motion capture system. (in Section 6.2)

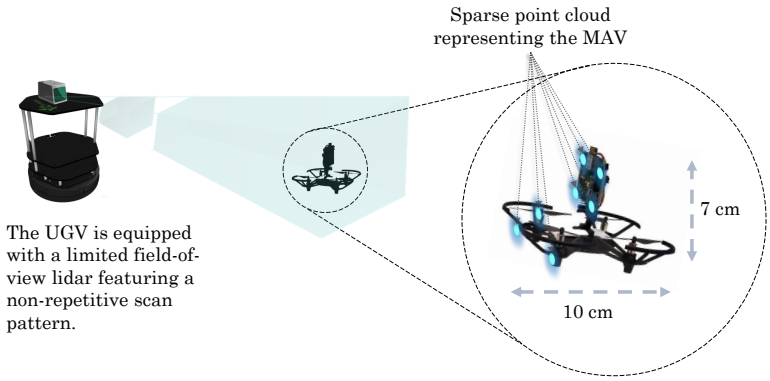
5. We provide a method to detect the initial position of the UAV using the object detection algorithm YOLOV5 over range images generated from a solid-state LiDAR point cloud. (in Section 6.2)

6.1 UAV Tracking Based on Adaptive Scan Integration

In this work, we assume that the initial position of the MAV after take-off is known. We also assume that its shape and size are known a priori. We develop a method targeting solid-state LiDARs owing to the higher density of the resulting point cloud even with more limited FoV. In these LiDARs, the concept of a frame or scan frequency changes considerably. Similarly as in rotating 3D LiDARs, a frame in solid-state LiDARs can be naturally related to a single revolution. With non-repetitive scan patterns, LiDARs can output point clouds at adjustable frequencies with varying FoV coverage, as illustrated in Figure 63a. This opens the door to a new LiDAR perception method that exploit the possibilities of adaptively adjust the frame integration



(a) Illustration of the field of view (FoV) coverage with different point cloud integration times in a non-repetitive LiDAR scanning device.



(b) Illustration of a ground robot tracking a micro-aerial vehicle (MAV) using a limited-FoV solid-state LiDARs.

Figure 63: Conceptual illustration of the field-of-view coverage with different integration times on a Livox Horizon LiDAR (top) and its application to tracking MAVs (bottom).

time to better sense the objects. We apply the proposed an adaptive LiDAR scan integration method within the problem of a UGV tracking a MAV for external state estimation, as conceptualized in Figure 63b. While our focus is on MAVs, the proposed method can also be easily adapted to detect foreign objects or intruder MAVs more accurately. We first put our focus on single and known MAV detection, but the presented generic method that can be extended to multi-MAV tracking as long as FoV limitations are accounted for.

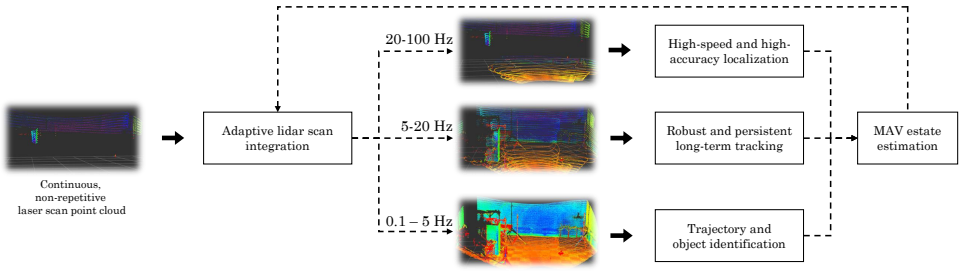


Figure 64: Overview of the proposed methods, where tracking is simultaneously performed at three different scan frequencies. Within each of these three threads, the scan frame integration is adjusted based on the distance to the target MAV and its speed.

6.1.1 Methodological Overview

We consider the problem of tracking a MAV from a ground robot. The ultimate objective is to improve the collaboration between the robots and the ability of the MAV to navigate in complex environments aided by the UGV. The rest of this paper delves into the definition, design, and implementation of a method for tracking a single MAV. Nonetheless, these can be extended to multiple MAVs. The main limitation when tracking multiple units is the FoV of the LiDAR sensors onboard the ground vehicle, and therefore assumptions have to be made to the spatial distribution of the MAVs (always within the FoV of the ground robot).

System Overview

We aim to define more optimal settings for generating point clouds based on the state (speed and distance to the sensor) of the MAV being tracked. We propose three simultaneous tracking modalities with three processes analyzing point cloud frames resulting in integration times ranging several orders of magnitude. A general view of the multi-modal tracking processes is shown in Figure 64. In more detail, the three modalities are described below:

1. Adaptive high-frequency tracking. In this first process, sparse point clouds are integrated at frequencies up to 100 Hz. The MAV is only trackable through a reduced number of points, but we are able to estimate its position and speed with high accuracy. In this process, the MAV is not necessarily recognizable in all processed frames.

2. Adaptive medium-frequency tracking. The second process operates at frequencies within the range of typical LiDAR scanners (i.e., 5 to 20 Hz). The frequency within that same range is dynamically adjusted to optimize the den-

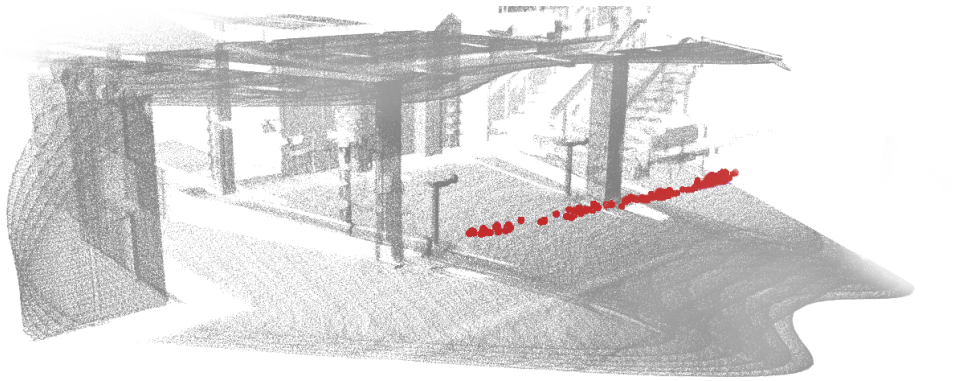


Figure 65: Integration trajectory recovery example

sity of the point cloud. At these frequencies, the extracted point cloud representing the MAV is distorted by motion, and thus the localization and speed estimation accuracy is lower. However, this process enables more robust and persistent tracking as the MAV can be recognized in most if not all frames.

3. Low-frequency trajectory and object validation. The third and last process that runs in parallel to the previous two performs long-term tracking and validates the reconstructed trajectory of the MAV based on predefined dimensional constraints. An illustration of such trajectory reconstruction is shown in Figure 65

Formulation

Let $\mathcal{P}_k(I_r^k)$ be the point cloud generated by the LiDAR with an integration time I_r^k. We also denote by $\mathbf{s}_{MAV}^k$=$\{\mathbf{p}_{MAV}^k$,$\dot{\mathbf{p}}_{MAV}^k\}$ the position and speed of the MAV. We use discrete steps represented by k owing to the discrete nature of the set of consecutive point clouds. The output of the main tracking algorithm is to extract from $\mathcal{P}_k(I_r^k)$ the set of points representing the MAV, which we denote by $\mathcal{P}_{MAV}^k$, and to adjust the integration time for the next point cloud, I_{HF}^k, I_{MF}^k.

Adaptive Scan Integration

Since we assume that the state of the MAV $(\mathbf{p}_{MAV}^{k-1}, \dot{\mathbf{p}}_{MAV}^{k-1})$ is initially known, the point cloud processing proceeds as follows: First, we perform ground removal based on the last-known altitude of the MAV. We then proceed with finding the nearest neighbor points to a predicted MAV position. This step is repeated for both the high and medium frequency scans, the former one providing a more accurate position estimation while the latter is more persistent in time. Finally, these two estimations are combined, and the results are utilized to adjust the integration rates based on

Algorithm 7: MAV tracking with adaptive scan integration

Input:

High- and medium-freq int. rates: $\{I_{HF}^{k-1}, I_{MF}^{k-1}\}$

3D lidar point clouds: $\{\mathcal{P}_k(I_{HF}^{k-1}), \mathcal{P}_k(I_{MF}^{k-1})\}$

Last known MAV state: $(\mathbf{p}_{MAV}^{k-1}, \dot{\mathbf{p}}_{MAV}^{k-1})$

Output:

MAV state: $\{\mathbf{p}_{MAV}^{k}, \dot{\mathbf{p}}_{MAV}^{k}\}$

UGV control: $\dot{\mathbf{q}}_{UGV}^{k}$

Int. rates: $\{I_{HF}^{k}, I_{MF}^{k}\}$

Function $object\_extraction\left(\mathcal{P},\ I,\ \mathbf{p}_{MAV}^{k-1},\ \dot{\mathbf{p}}_{MAV}^{k-1}\right)$**:**

Ground removal: $\mathcal{P}^{'} \leftarrow \mathcal{P}$;

Generate KD Tree: $kdtree \leftarrow \mathcal{P}^{'}$;

MAV pos estimation: $\hat{\mathbf{p}}_{MAV}^{k} \leftarrow \mathbf{p}_{MAV}^{k-1} + \frac{\dot{\mathbf{p}}_{MAV}^{k-1}}{I}$;

MAV points: $\mathcal{P}_{MAV}^{k} = KNN(kdtree,\ \hat{\mathbf{p}}_{MAV}^{k})$;

MAV state estimation: $\mathbf{p}_{MAV}^{k} = \frac{1}{|\mathcal{P}_{MAV}^{k}|} \sum_{p \in \mathcal{P}_{MAV}^{k}} p$;

return $\mathbf{p}_{MAV}^{k}$;

`// Coarse but persistent tracking`

while *new* $\mathcal{P}_k(I_{MF}^{k})$ **do**

  $\mathbf{p}_{MAV}^{k'} = object\_extraction(\mathcal{P}_k(I_{MF}^{k}), I_{MF}^{k}, \mathbf{p}_{MAV}^{k-1}, \dot{\mathbf{p}}_{MAV}^{k-1})$;

`// Fine-grained estimation`

while *new* $\mathcal{P}_k(I_{HF}^{k})$ **do**

  $\mathbf{p}_{MAV}^{k''} = object\_extraction(\mathcal{P}_k(I_{HF}^{k}), I_{HF}^{k}, \mathbf{p}_{MAV}^{k-1}, \dot{\mathbf{p}}_{MAV}^{k-1})$;

  $\mathbf{p}_{MAV}^{k},\ \dot{\mathbf{p}}_{MAV}^{k} \leftarrow estimate\left(\mathbf{p}_{MAV}^{k'},\ \mathbf{p}_{MAV}^{k''}\right)$;

$\{I_{HF}^{k}, I_{MF}^{k}\} \leftarrow adjust\_integration\_freqs\left(\mathbf{p}_{MAV}^{k}, \dot{\mathbf{p}}_{MAV}^{k}\right)$;

$\dot{\mathbf{q}}_{UGV}^{k} \leftarrow keep\_within\_FoV\left(\mathbf{p}_{MAV}^{k}, \dot{\mathbf{p}}_{MAV}^{k}\right)$;

the point cloud density expected for the given distance and speed. This process is outlined in Algorithm 7.

Trajectory Validation

The main purpose of the low-frequency scan stream is to validate the extracted MAV's trajectory. While the tracking with adaptive scan integration only takes into account the MAV size roughly in terms of distance within which nearest neighbors are looked for, the extracted point cloud is not validated against its known dimensions. This is done when enough points are accumulated into a reconstructed trajectory. As exposed in Algorithm 8, we first perform a cubic spline interpolation based on the history of estimated positions and speeds. To calculate the parameters

Algorithm 8: Trajectory validation

Input:

Low-freq int. rate: I_{LF}^{k-1}

3D lidar point cloud: $\mathcal{P}_k\left(I_{LF}^{k-1}\right)$

MAV state history: $(\mathbf{p}_{MAV},\, \dot{\mathbf{p}}_{MAV})$

Output: Trajectory validation (*bool*)

while *new* $\mathcal{P}_k\left(I_{LF}^{k-1}\right)$ **do**

    // Generate cubic splines
    // with position and speed constraints

    $\{B_i\} \leftarrow \{\mathbf{p}_{MAV},\, \dot{\mathbf{p}}_{MAV}\}$;

    // Estimate expected point cloud from
    // known density at given distance and speed

    $\hat{\mathcal{P}}_k \leftarrow \{\{B_i\},\, \mathbf{p}_{MAV},\, \dot{\mathbf{p}}_{MAV}\}$;

    // Calculate IoU

    $IoU = calc\_IoU\left(\mathcal{P}_k\left(I_{LF}^{k-1}\right), \hat{\mathcal{P}}_k\right)$;

    if $IoU > th$ **then**

        return True

    else

        return False

of the cubic spline, we utilize constraints on the first derivative based on the speed, rather than forcing the first and second derivative to be continuous. Indeed, the acceleration of the MAV can suddenly change. Based on predetermined values of point cloud density as a function of the MAV's distance to the LiDAR and its speed, we then produce an expected point cloud. We validate the original point cloud given a threshold for the IoU measure with the generated estimate.

6.1.2 Experimental Setup

Experimental platforms

The experimental platform (shown in Figure66) consists on a single ground robot and a commercially available Ryze Tello MAV. The ground robot is an EAI Dashgo platform equipped with a Livox Horizon LiDAR ($81.7° \times 25.1°$ FoV). The lidar is able to output scanned pointcloud up to 100 Hz, featuring a non-repetitive pattern.

A pair of UWB transceivers is used to obtain a single range between the robot and the MAV at frequencies ranging from 10 Hz to 100 Hz.The UWB ranging is used in aiding the manual validation of the extracted trajectory in places where there was no external positioning system. In the future work, it could be incorporated as part of the tracking algorithm as well, as is becoming increasing adopted in multi-robot

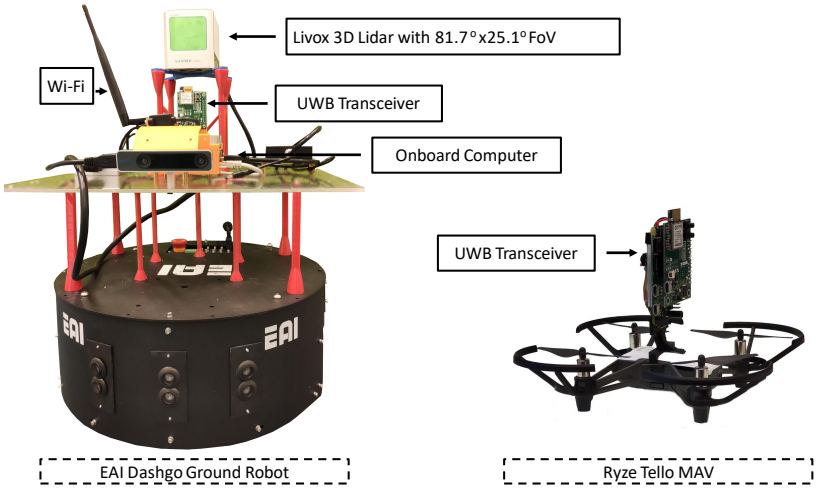


Figure 66: Ground robot and MAV utilized in the experiments.

systems [61; 73].

Software

The system has been implemented using ROS Melodic under Ubuntu 18.04. The algorithms are running in the main computer onboard the ground robot. The computer runs the Tello MAV driver[1], the Livox LiDAR driver[2], and our open-source MAV tracking package[3]. The latter is a multi-threaded node able to process the different point clouds in real time. The point cloud library (PCL) [262] is utilized to extract the position of the MAV from the LiDAR's point cloud.

Metrics

Owing to the lack of an accurate external positioning system such as a MOCAP system for a large environment. our focus is instead on measuring the performance of the tracking at different scan integration rates and manually validating the overall trajectory. The experimental flights are carried out in large indoor halls with multiple columns and objects, as shown in Figure 28. Another set of experiments is carried out in a small flying area where an external UWB positioning system was available and used to fly the MAV over a predefined trajectory. A characterization on the accuracy of such system can be found in [263].

[1] https://github.com/TIERS/tello-driver-ros

[2] https://github.com/Livox-SDK/livox_ros_driver

[3] https://github.com/TIERS/adaptive-lidar-tracking

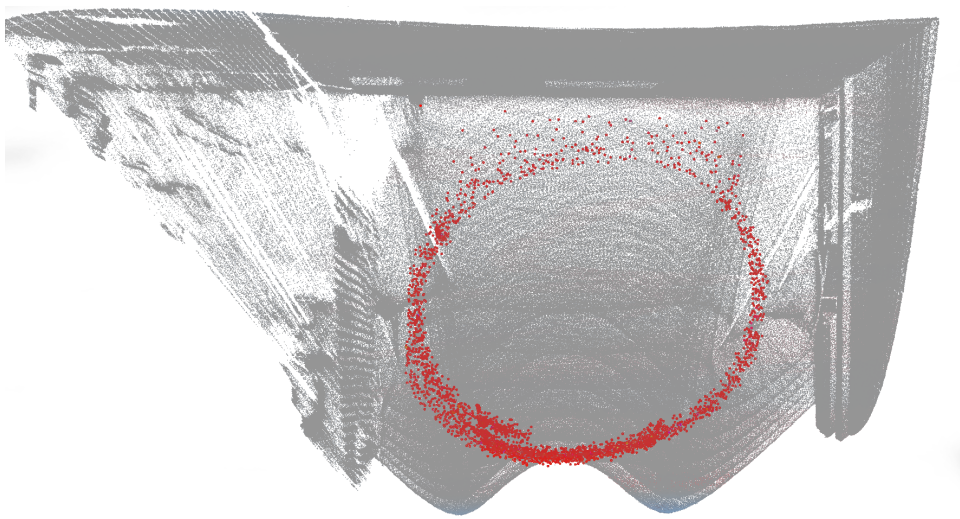


Figure 67: Accumulated point cloud for the circular trajectory.

6.1.3 UAV Tracking Results

The experimental results consist mainly of flights in two different indoor environments and different conditions.

Adaptive scan integration

The first objective of our experiments was to assess the tracking performance at different scan frequencies in order to better model the adaptiveness of our algorithm. In order to adapt the scanning frequency to optimize the tracking performance, key parameters are the point cloud density, which is the number of points representing the UAV, at different distances and the reliability of the detections at different speeds.

The point cloud density for different scanning frequencies as a function of the distance between the LiDAR and the MAV is shown in Figure 68. This measure refers only to the density of the points representing the MAV and not the overall density including the rest of the scene. The darker lines represent the average point cloud density, while the band with higher transparency represents the values within the standard deviation. The size of the Tello MAV is about 500 cubic centimeters. According to our observation in the experiments, reliable tracking at high speeds can be achieved with at least 4 points, while we require at least 20 points at medium scanning frequency. This, however, only applies in free space. As can be seen in Figure 67, significant noise appears in the point cloud between the MAV and walls in the environment when flying nearby. We discuss further this issue at the end of this section.

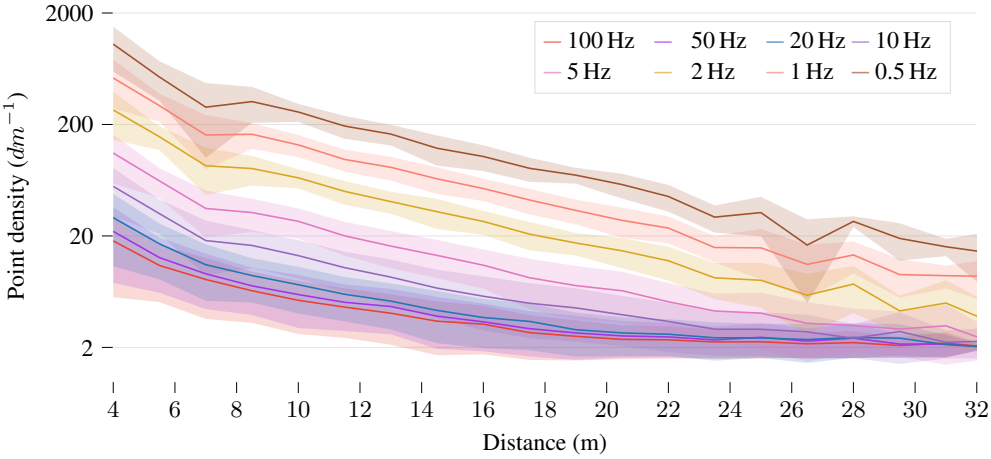


Figure 68: Density of the point cloud (number of points) representing the MAV based on the distance to the LiDAR scanner and the scanning frequency.

In terms of the tracking performance based on the speed, we plot in Figure 69 the distance between consecutive detections at different scanning frequencies. The results in this particular figure cannot be directly utilized to model the adaptive nature of our tracking Algorithm Nonetheless, they can be leveraged to better understand what are the speed limits under which given scanning frequencies do not provide the expected distance between detection that can be inferred from the MAV speed and the scan frequency.

The results included in Figure 68 and Figure 69 have been obtained flying the MAV in a long, straight corridor with a length of about 35 m. The MAV was flying mostly in straight lines and the speed was estimated using both visual odometry and the position history extracted from the LiDAR data in a partially manual manner.

Qualitative trajectory validation

In order to validate the performance of the tracking algorithm and better understand the limitations of our tracking approach at different scanning frequencies, we compare two different types of trajectories. Owing to the lack of a system to obtain ground truth (e.g., a motion capture system), we provide qualitative analysis for one of the trajectories and compare it with a UWB positioning system in the other one.

First, we test the tracking algorithm through a trajectory where the MAV flies in a large open area at distances from 2 m to over 17 m far from the LiDAR scanner and variable speeds. In this scenario, the analysis is mostly qualitative, with the trajectories shown in Figure 70. However, the UWB ranging data and the LiDAR data has been both manually checked and the maximum positioning error along the

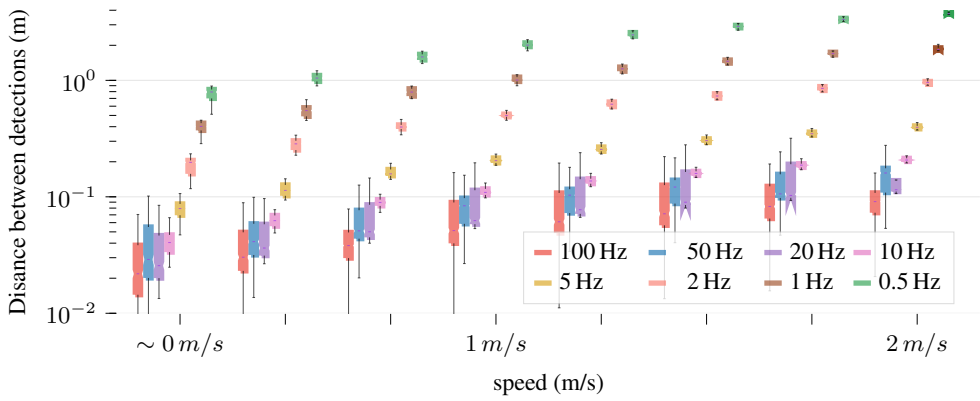


Figure 69: Distance between consecutive MAV detections based on its speed and the LiDAR's scanning frequency.

track is at worst around 20 cm. Qualitatively, the main results from this experiment are the ability of the tracking algorithm to keep track of the MAV over changes in speed, direction, and at longer distances. The figure only shows frequencies equal to or above 5 Hz because at lower scanning frequencies the speed estimation was highly inaccurate during the early stages of the flight. We can see that only at the highest frequency we are able to track the MAV along the completed trajectory, while the trajectory itself is noisier. The higher level of error when estimating the MAV position is due to a lower number of points being detected, which can correspond to different parts of the MAV in consecutive scans. The last subplot shows the overall estimated trajectory where our algorithm has combined the different scanning frequencies to obtain the smoothness of the medium frequencies and the performance of the higher frequencies. The trajectory also employs the cubic spline interpolation from the validation Algorithm

Second, we perform a continuous flight with a predefined circular trajectory in a small flying arena where the UWB positioning system is available. The results for this flight are shown in Figure 71. The leftmost plot shows the reference position. However, it is worth noticing that the accuracy of the LiDAR, of around 2 cm for distances smaller than 20 m, is higher than the average accuracy of 10 to 15 cm in the UWB positioning system. Therefore, the trajectory is mere as a reference and only a qualitative discussion is possible with these results. In any case, owing to the continuous change in the speed of the MAV, which is a prior unknown to the tracking algorithm, again only at frequencies equal or over 5 Hz are we able to track the MAV. Nonetheless, at 5 Hz the tracking stops before the fourth revolution is completed, and persistent tracking is only possible when higher frequencies are taken into account.

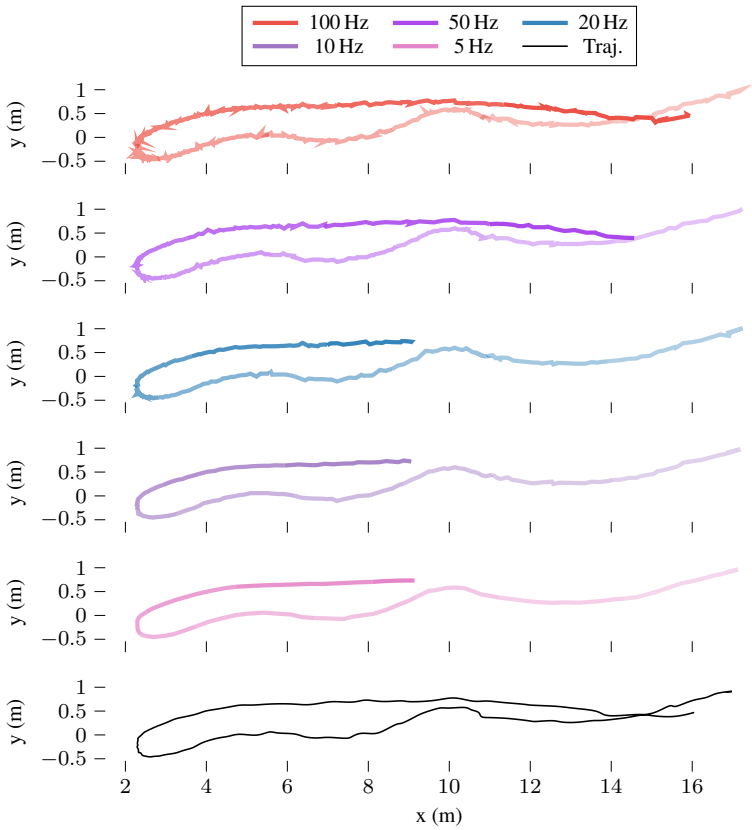


Figure 70: Estimated trajectories at different frequencies and with adaptive approach (top five plots), and trajectory estimated from our algorithm (bottom plot).

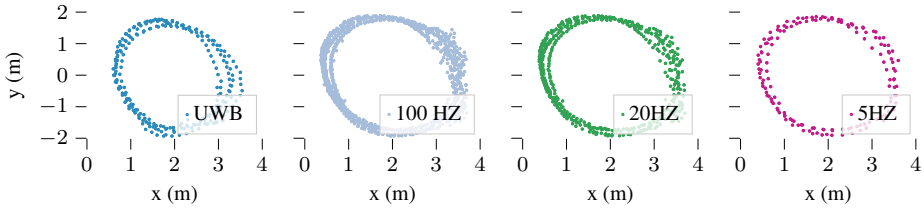


Figure 71: Reference trajectory (UWB) and estimated positions at different fixed frequencies.

Discussion

We have shown in this section qualitative results that show the performance of the adaptive tracking algorithm and the same approach applied only to specific scanning frequencies. From both sets of experiments, the main conclusion is that the adaptive approach is able to accommodate a wider variety of scenarios. We have been able to put together the flexibility of high-speed tracking with the robustness of medium frequencies, avoiding the frequent errors of the former, and the lower tracking capacity of the latter is more challenging conditions.

One key limitation when tracking MAVs, as visualized in the circular trajectory experiments, is the low density of the point cloud and the inability to tell the difference between the MAV's points and LiDAR noise. This is also due to the low reflectively of the MAV, and there is thus the potential for mitigation with more reflective surfaces that could aid in separating the sparse MAV point cloud from the LiDAR noise originated due to near objects. As we can see in Figure 67, the point cloud density near the rear wall is very sparse in some areas, therefore being unable to reconstruct a robust trajectory as there are multiple options available that would meet the dynamics and dimensional constraints of the MAV.

6.2 UAV Tracking Based on Dynamic Multi-Frequency Scan Integration using Kalman Filter

In this work, we build upon our previous works in Section 6.1 to provide a more sophisticated approach that enables real-time UAV tracking. Specifically, we design and develop an algorithm to dynamically adjust the scan integration time, or frequency, based on the tracking state (position and velocity). We do this in parallel for two different integration frequencies, both dynamically adjusted. A higher frequency, as preliminary results in [7] show, allows for more accurate tracking, while the lower frequency enables persistence over time. Importantly, we address here two main limitations in [7]. First, the lack of tracking, as the previous algorithm simply tracks a UAV by iterative detections. Second, while the concept is similar, the integration in [7] is done offline and in post-processing. We also provide a quantifiable comparison with ground truth data, extending the benchmarks in [264] with the new dynamic multi-frequency integration approach. Finally, we no longer require a manually given priori knowledge of the UAV shape but only its approximate size.

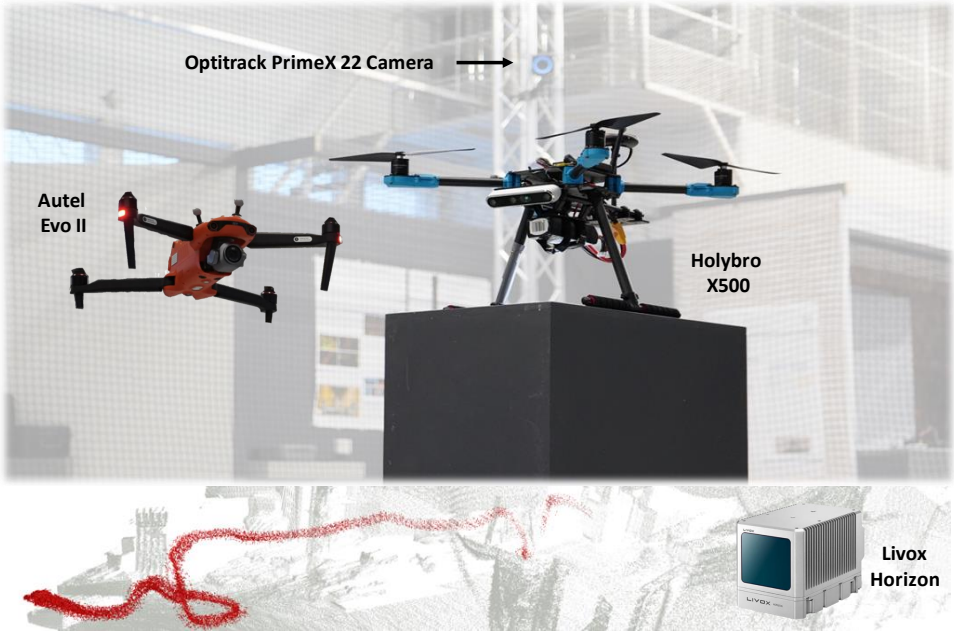


Figure 72: Illustration of hardware used in the experiments.

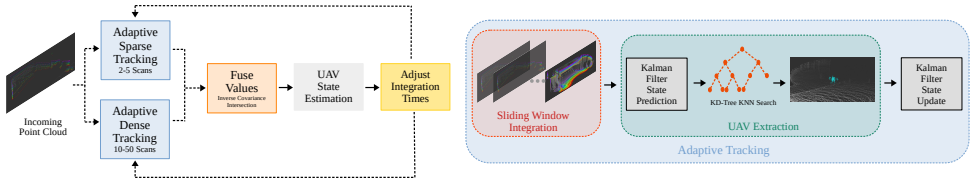


Figure 73: Overview of the proposed method: tracking is simultaneously performed at two different integration time ranges and then fused using the Inverse Covariance Intersection (ICI) method. The fused estimate is used to refine the estimations and integration times of both estimators.

6.2.1 Methodology

Baseline Method

To establish a baseline for comparison, we investigate a tracking approach that keeps the integration time constant along the UAV trajectory. Unlike our proposed method, the approach in [264] performs tracking without using a sliding window. Our baseline method uses a sliding window to integrate the LiDAR scans and capture more relevant information while keeping the integration time constant.

Overall, the baseline method is similar to our adaptive method, with the main difference being that in the adaptive method, we vary the integration time based on the UAV motion dynamics. In the following section, we provide further analysis of the results and compare the performance of the adaptive method to that of the baseline.

Formulation

We propose two simultaneous tracking estimators on two different scan frequencies running in parallel where the integration time I, defining the number of scans to accumulate, is dynamically adjusted to optimize the point cloud density as depicted in Figure 73:

(i) Adaptive Sparse Tracking (AST): Sparse point clouds are integrated up to 5 consecutive scans. This process provides high accuracy for estimating the UAV position and speed, but only tracks it through a reduced number of points and not necessarily in all frames.

(ii) Adaptive Dense Tracking (ADT): The number of scans ranges between 10 and 50. The extracted point cloud representing the UAV is distorted by motion, which reduces the accuracy of localization and speed estimation, but the tracking is more robust and persistent as the UAV can be recognized in most frames.

In the following formulation, we denote discrete steps as k due to the discrete nature of the set of consecutive point clouds.

Let $\mathcal{P}_k(I_r^k) = \{\mathbf{p}_1^k, \mathbf{p}_2^k, \ldots, \mathbf{p}_{n_k}^k\}$ be the set of n_k points in the point cloud generated by the LiDAR sensor at time step k using an integration time I_r^k, where r is the range of the interval. The objective of the tracking algorithm is to identify the subset of points in $\mathcal{P}_k(I_r^k)$ that corresponds to the UAV, denoted $\mathcal{P}_{\text{UAV}}^k$, to estimate its position and velocity, and consequently adjust the integration time for the next point cloud in the two integration time intervals, $I_{\text{AST}}^k, I_{\text{ADT}}^k$.

To initiate the tracking process, we assume the initial position of the UAV to be known. In this part of the experiment, we manually give the initial position of the UAV for the verification of our tracking algorithm. Subsequent investigations will consider the feasibility of employing intensity images from the solid-state LiDAR to automate the acquiring of this preliminary data by UAV detection. The point cloud $\mathcal{P}_k(I_r^k)$ is integrated by accumulating the number of scans defined by I_r^k in a sliding window fashion. This allows us to retain the most recent information about the state of the UAV in the point cloud. We then employ a Nearest-Neighbor Search (NNS) algorithm to identify the points in the point cloud that are closest to the predicted position of the UAV, based on its initial position. To improve the reliability and accuracy of the tracking results, we leverage a priori information about the dimensions of the tracked object. Specifically, the NNS is constrained to a search radius of r, which is set to half of the largest dimension (length, width or height) of the UAV. This allows us to constrain the NNS to a smaller volume around the estimated position, leading to faster and more accurate search results.

Next, we estimate a new position for the UAV by averaging the extracted points, which serves as the measurement in the Extended Kalman Filter (EKF) update step. To account for the large distances between scans caused by the velocity of the UAV, we have chosen to prioritize the most recent point clouds. This means that points in these more recent clouds are given greater importance than those in earlier ones. We accomplish this by assigning a weight to each point based on its timestamp t_p, which follows the formula shown in Equation (11):

$$w_{\text{p}} = \exp\left[-\gamma \times \left(t_{\text{scan}} - t_{\text{p}}\right)\right] \tag{11}$$

where t_{scan} corresponds to the time at which the latest scan is acquired.

For the prediction step of the EKF, we use a Constant Turn Rate and Velocity (CTRV) motion model, commonly used for airborne tracking systems [265]. In our case, we extended this model to the 3D scenario by incorporating a Constant Velocity (CV) motion model for the z coordinate. We opted for this model for its proven robustness in the literature [266; 267].

The state space

$$\mathbf{x} = \begin{bmatrix} x & y & z & v & \psi & \dot{\psi} \end{bmatrix}^T \tag{12}$$

can be transformed by the non-linear state transition.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \begin{bmatrix} \frac{v_k}{\dot{\psi}_k}\left(\sin\left(\psi_k + \dot{\psi}_k \Delta t\right) - \sin\left(\psi_k\right)\right) \\ \frac{v_k}{\dot{\psi}_k}\left(-\cos\left(\psi_k + \dot{\psi}_k \Delta t\right) + \cos\left(\psi_k\right)\right) \\ \dot{z}\Delta t \\ 0 \\ \dot{\psi}_k \Delta t \\ 0 \end{bmatrix} \tag{13}$$

Next, the updated predicted position is used as input to the NNS algorithm to obtain an updated measurement. This measurement is then used in the next iteration of the EKF to further improve the accuracy of the predicted position. For more details, we refer the reader to Algorithm 9.

Algorithm 9: UAV tracking with adaptive scan integration

Input:

Adaptive Sparse and Dense Tracking int. rates: $\left\{I_{\text{AST}}^{k-1}, I_{\text{ADT}}^{k-1}\right\}$

3D lidar point clouds: $\left\{\mathcal{P}_k\left(I_{\text{AST}}^{k-1}\right), \mathcal{P}_k\left(I_{\text{ADT}}^{k-1}\right)\right\}$

Last known UAV state: $\left(\mathbf{x}_{\text{UAV}}^{k-1}, \dot{\mathbf{x}}_{\text{UAV}}^{k-1}\right)$

Output:

UAV state: $\{\mathbf{x}_{\text{UAV}}^{k}, \dot{\mathbf{x}}_{\text{UAV}}^{k}\}$

Function $uav\_tracking\left(\mathcal{P},\ I,\ \boldsymbol{x}_{UAV}^{k-1},\ \dot{\boldsymbol{x}}_{UAV}^{k-1}\right)$**:**

UAV pos estimation: $\hat{\mathbf{x}}_{\text{UAV}}^{k}, \dot{\hat{\mathbf{x}}}_{\text{UAV}}^{k} = \mathcal{KF}_{\text{prediction}}\left(\mathbf{x}_{\text{UAV}}^{k-1}, \dot{\mathbf{x}}_{\text{UAV}}^{k-1}\right)$;

Generate KD Tree: $kdtree \leftarrow \mathcal{P}$;

UAV points: $\mathcal{P}_{\text{UAV}}^{k} = NNS\left(kdtree,\ \hat{\mathbf{x}}_{\text{UAV}}^{k}\right)$;

UAV measurement: $\mathbf{z}_{\text{UAV}}^{k} = \frac{1}{|\mathcal{P}_{\text{UAV}}^{k}|}\sum_{x \in \mathcal{P}_{\text{UAV}}^{k}} x$;

UAV state estimation: $\mathbf{x}_{\text{UAV}}^{k} = \mathcal{KF}_{\text{update}}\left(\mathbf{z}_{\text{UAV}}^{k}\right)$;

return $\mathbf{x}_{\text{UAV}}^{k}, \dot{\mathbf{x}}_{\text{UAV}}^{k}$;

// Adaptive Sparse Tracking

foreach *new* $\mathcal{P}_k\left(I_{AST}^{k}\right)$ **do**

  $\mathbf{x}_{\text{UAV}}^{k''} = uav\_tracking\left(\mathcal{P}_k\left(I_{\text{AST}}^{k}\right), I_{\text{AST}}^{k}, \mathbf{x}_{\text{UAV}}^{k-1}, \dot{\mathbf{x}}_{\text{UAV}}^{k-1}\right)$;

// Adaptive Dense Tracking

foreach *new* $\mathcal{P}_k\left(I_{ADT}^{k}\right)$ **do**

  $\mathbf{x}_{\text{UAV}}^{k'} = uav\_tracking\left(\mathcal{P}_k\left(I_{\text{ADT}}^{k}\right), I_{\text{ADT}}^{k}, \mathbf{x}_{\text{UAV}}^{k-1}, \dot{\mathbf{x}}_{\text{UAV}}^{k-1}\right)$;

// Inverse Covariance Intersection

$\{\mathbf{x}_{\text{UAV}}^{k}, \dot{\mathbf{x}}_{\text{UAV}}^{k}\} \leftarrow new\_state\_estimation\left(\mathbf{x}_{\text{UAV}}^{k'}, \mathbf{x}_{\text{UAV}}^{k''}\right)$;

$\{I_{\text{AST}}^{k}, I_{\text{ADT}}^{k}\} \leftarrow adjust\_integration\_rates\left(\mathbf{x}_{\text{UAV}}^{k}, \dot{\mathbf{x}}_{\text{UAV}}^{k}\right)$;

Within each integration time interval, the integration time is adjusted based on the distance and velocity of the tracked object. To avoid motion blur, shorter integration times are used for closer and faster-moving objects, while longer integration times are used for more distant and slower-moving objects. This adjustment is based on Equation (14).

$$I_r^k = I_{\text{min}}^k + \left(I_{\text{max}}^k - I_{\text{min}}^k\right)\left(\frac{d}{d_{\text{max}}}\right) \tag{14}$$

Here, I_{max}^k and I_{min}^k define the upper and lower bound for the integration time within each interval, and d represents the current distance of the UAV from the sensor. As for d_{max}, it represents the maximum distance within an integration time interval for which a minimum of 4 points on the UAV were recognized in the point cloud $\mathcal{P}_k(I_r^k)$. This distance was empirically calculated by manually determining the number of points detected at a distance of 4 m. Moreover, a simplified approximation of the number of points hitting the target was designed assuming that the point density remains constant, and the primary factor influencing the number of points is the spreading of points over distance due to the LiDAR's angular resolution as per Equation (15):

$$N = \frac{S^2}{d^2\theta} \tag{15}$$

where S represents the side length of the target surface, d is the distance from the sensor and θ represents the LiDAR's angular resolution in radians which accounts for the angular spacing between individual points.

The maximum distance d_{max} was then extrapolated using the Equation (16):

$$N_d = \frac{4^2 N_4}{d^2} \tag{16}$$

where N_d is the number of points detected at distance d, and N_4 is the number of points detected at a distance of 4 m.

To further enhance the accuracy and reliability of the tracking system, the AST and ADT tracking modalities are fused together. One intuitive method is the naive fusion which ignores the inherent correlations directly. In this sense, this method is too optimistic, and cannot guarantee consistent results [268].

In our method, we employ two different Kalman filters, each generating its own estimate based on the accumulated data. While these estimates originate from the same LiDAR sensor, the use of separate Kalman filters implies a degree of independence between the two estimation processes. This separation can be seen as a form of distributed processing, where different components (the two Kalman filters) are responsible for generating their own estimates. Its distributed nature can guide the selection of appropriate fusion techniques, such as Inverse Covariance Intersection

(ICI) [269], that represents a well-suited fusion rule for typical Kalman filter-based fusion problems [270].

ICI combines the covariance matrices of the two estimators ($\mathbf{\Sigma}_{\text{AST}}$ and $\mathbf{\Sigma}_{\text{ADT}}$) using a weighting factor $\omega \in [0, 1]$ to generate a fused matrix that more accurately represents the estimated state uncertainty.

The state estimations $\mathbf{x}_{\text{AST}}$ and $\mathbf{x}_{\text{ADT}}$ are fused into a final state $\mathbf{x}_{\text{UAV}}$ using a weighted average:

$$\mathbf{x}_{\text{UAV}} = \mathbf{K}_{\text{ICI}}\mathbf{x}_{\text{AST}} + \mathbf{L}_{\text{ICI}}\mathbf{x}_{\text{ADT}} \tag{17}$$

where the gains $\mathbf{K}_{\text{ICI}}$ and $\mathbf{L}_{\text{ICI}}$ are computed based on the fused covariance $\mathbf{\Sigma}_{\text{UAV}}$ and the individual covariances of the two measurements:

$$\begin{aligned}
\mathbf{\Sigma}_{\text{UAV}} &= \left(\mathbf{\Sigma}_{\text{AST}}^{-1} + \mathbf{\Sigma}_{\text{ADT}}^{-1} - \left(\omega\,\mathbf{\Sigma}_{\text{AST}} + (1-\omega)\;\mathbf{\Sigma}_{\text{ADT}}\right)^{-1}\right)^{-1} \\
\mathbf{K}_{\text{ICI}} &= \mathbf{\Sigma}_{\text{UAV}}\left(\mathbf{\Sigma}_{\text{AST}}^{-1} - \omega\left(\omega\,\mathbf{\Sigma}_{\text{AST}} + (1-\omega)\,\mathbf{\Sigma}_{\text{ADT}}\right)^{-1}\right) \\
\mathbf{L}_{\text{ICI}} &= \mathbf{\Sigma}_{\text{UAV}}\left(\mathbf{\Sigma}_{\text{ADT}}^{-1} - (1-\omega)\left(\omega\,\mathbf{\Sigma}_{\text{AST}} + (1-\omega)\,\mathbf{\Sigma}_{\text{ADT}}\right)^{-1}\right)
\end{aligned} \tag{18}$$

The optimal value of ω is found using Brent's algorithm [**?**] to minimize the trace of the inverse covariance matrix:

$$\text{trace}\left[\left(\mathbf{\Sigma}_{\text{AST}}^{-1} + \mathbf{\Sigma}_{\text{ADT}}^{-1} - \left(\omega\,\mathbf{\Sigma}_{\text{AST}} + (1-\omega)\,\mathbf{\Sigma}_{\text{ADT}}\right)^{-1}\right)^{-1}\right] \tag{19}$$

By fusing the two estimators at two frequencies, the tracking system can better handle challenging tracking scenarios, such as sensor noise, and provide more accurate and reliable estimates of the UAV state.

6.2.2 Experimental Results

The experimental platform consist of a Livox Horizon LiDAR ($81.7° \times 25.1°$ FoV) capable of generating non-repetitive point cloud scans up to 100 Hz, and an external positioning system to validate the extracted trajectories. To test our adaptive tracking algorithm, we evaluated its performance on several trajectories.

Metrics

First, to quantify the disparity between the LiDAR and the external position system estimates, we computed the error by taking the difference between the position estimates obtained from both systems for two distinct positions and orientations of the target. This analysis revealed a RMSE of 0.0143 m.

Table 26: Position error (RMSE) for both constant and adaptive integration methods (N/A when the error diverges because the estimated trajectory is incomplete. Unit: meter)

Track	**Tracking Method**										
	I_2	I_5	I_{10}	I_{20}	I_{50}	I^{KF}	$I^{\text{KF}}_{\text{AST}}$	$I^{\text{KF}}_{\text{ADT}}$	I^{EKF}	$I^{\text{EKF}}_{\text{AST}}$	$I^{\text{EKF}}_{\text{ADT}}$
T1	0.0786	0.0774	0.0788	0.0844	0.1019	0.0782	**0.0773**	0.0845	0.0852	0.087	0.0861
T2	0.0202	**0.0195**	0.0227	0.0332	0.0515	0.0253	0.0203	0.0335	0.0395	0.0371	0.046
T3	0.0491	**0.0421**	0.0451	0.0553	0.0803	0.0456	0.0431	0.0537	**0.0421**	0.0426	0.0466
T4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.0758	**0.0535**	0.1025
T5	N/A	0.0471	0.0564	0.0841	0.1682	0.0551	0.0501	0.0783	0.0377	**0.0314**	0.0624
T6	N/A	0.0551	0.0609	0.0915	N/A	0.0707	N/A	0.0861	0.0373	**0.0291**	0.0571
T7	N/A	N/A	N/A	N/A	N/A	0.0873	N/A	N/A	0.0580	**0.0538**	0.0733
T8	N/A	N/A	N/A	N/A	N/A	0.0939	N/A	0.1154	0.0530	**0.0487**	0.0732
T9	N/A	N/A	N/A	N/A	0.2576	N/A	N/A	N/A	0.0616	**0.0537**	0.0813
T10	N/A	0.0956	0.0828	0.1089	0.1911	0.0881	0.1167	0.1071	0.0522	**0.0460**	0.0730

To benchmark the proposed approach, we compared two versions of our adaptive method, one using a KF with a CV motion model, and the other using an EKF with a CTRV motion model. As baseline, we also included tracking keeping the integration time constant. To evaluate the tracking performance quantitatively, we used the RMSE metric, with the main results summarized in Table 26.

Indoor Experiments

Our results show that the proposed method outperforms the suggested baseline approach. As shown in Table 26, if the integration time is held constant, the error quickly increases as more scans are integrated. This means that accurate tracking is only possible if the lower end of the range is taken into account.

Despite the inherently higher error with lower scan frequencies, our method offers potential benefits in terms of robustness, efficiency, and flexibility: by adapting the integration time to the UAV motion dynamics, the method is more robust to changes in the environment and can handle unforeseen circumstances, such as sudden changes in direction. Additionally, the adaptive method is more efficient, as it only integrates the number of scans required to obtain accurate state estimates, rather than using a fixed integration time, and it allows for different integration times to be used in different parts of the trajectory. Therefore, although the combination of the two estimators does not always lead to the best accuracy, at the expenses of a slightly higher RMSE it increases robustness offering potential benefits that could be valuable in certain scenarios: it can effectively fuse the two simultaneous scan frequencies and provide accurate and reliable estimates of the UAV state, even when only one of the two estimators is available.

As shown in Table 26, specifically for the tracks T6, T7 and T8, when one or both single estimators fail to track the target, our method can still perform tracking along the whole trajectory by combining both methods using the ICI approach. This demonstrates the effectiveness of our adaptive method in handling challenging tracking scenarios, such as when one estimator is unavailable or unreliable.

Moreover, in 8 out of the 10 analyzed trajectories the EKF demonstrates lower RMSE compared to the KF. Notably, for trajectories T4 and T9, the KF alone fails to effectively complete the tracking task. This suggests that using the CTRV motion model extended to 3D along with an EKF is a viable option also for UAVs.

Furthermore, for both track T1 and T10, although our method does not achieve the overall best performance, it still results in lower errors compared to using the single estimators. In addition to the quantitative analysis, we provide visualizations of a subset of the trajectories obtained with our adaptive methods in Figure 74 and Figure 77. More results are available on the project's GitHub page. Both methods were able to estimate the overall trajectories well. We also observed that the combination of EKF and CTRV was able to track the target more accurately in situations where the UAV transitioned from a straight motion to a curve, as illustrated in the zoomed portion of Figure 74.

Initialization and Outdoors Experiments

While the presented outcomes demonstrate the feasibility of our proposed approach, it is worth noting that the quantitative results rely on the assumption of the initial position being already known. To address this issue, we have developed a method to detect the target's starting location in outdoor scenarios. Drawing inspiration from the effective tracking at short distances demonstrated by Sier et al. [135], who used signal images generated by a spinning LiDAR, we employed the same custom YOLOv5 model trained on panoramic signal images generated by an Ouster LiDAR [135] to detect the UAV. In our tests, we generate our own range images using a combination of depth and intensity values from a solid-state LiDAR point cloud.

To minimize noise and artifacts when creating a single image, we first integrate a total of 30 frames. Later, the 3D point cloud is projected onto a 2D plane, taking into account both the field of view and image resolution. The transformation process considers both the intensity and distance of each point, combining them through a weighted sum operation to produce the final result. Using only intensity or distance to detect drones from the background becomes challenging for the YOLO model due to the proximity to ground and walls as well as the reflectivity of both the background and the drone being similar.

Furthermore, normalization is applied to ensure appropriate contrast in the resulting image. Upon obtaining the preliminary 2D image, its quality is enhanced through filtering and interpolation: we first identify areas with zero values and substitute them

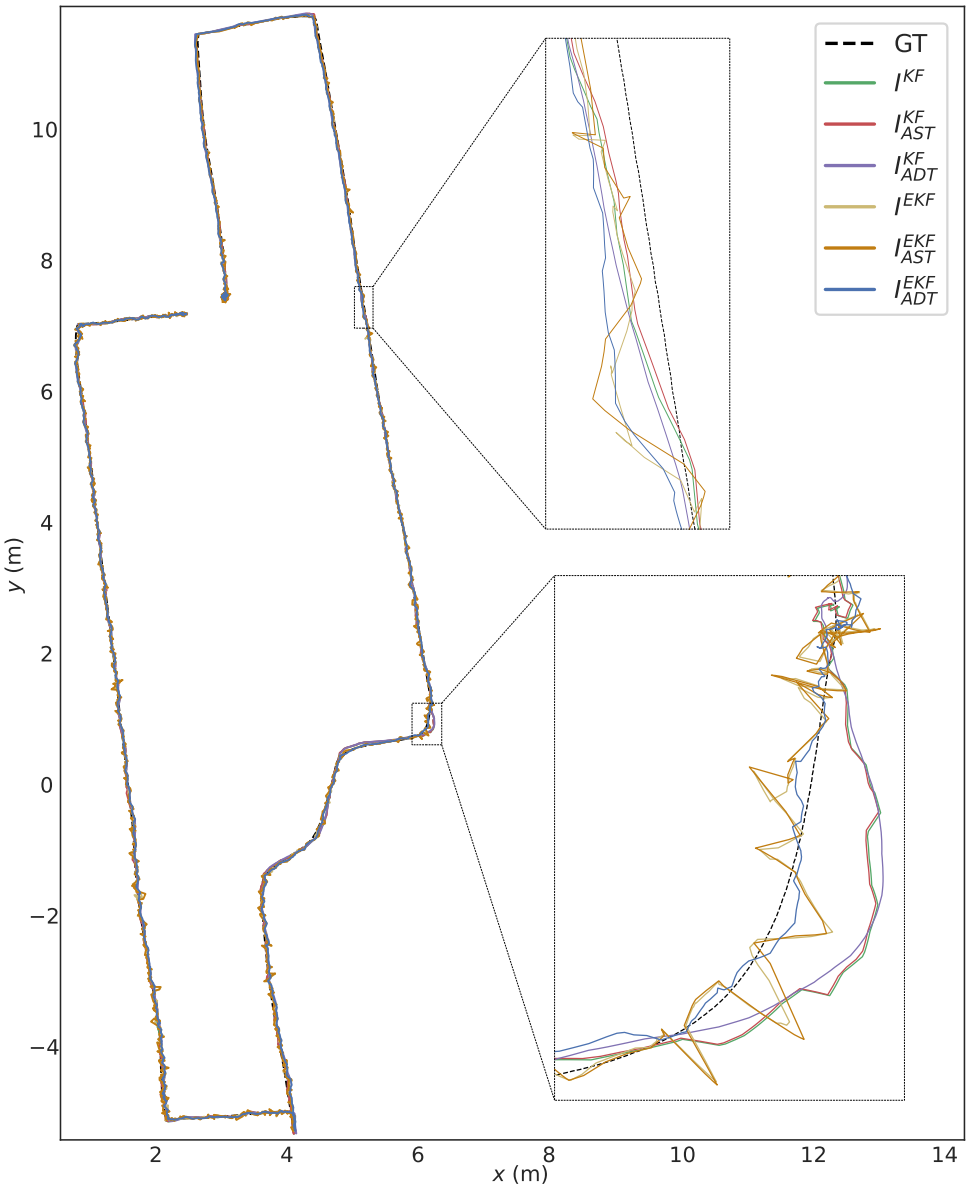
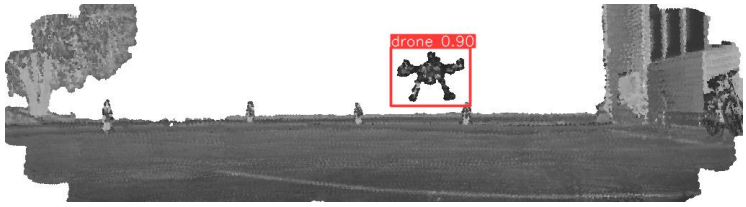
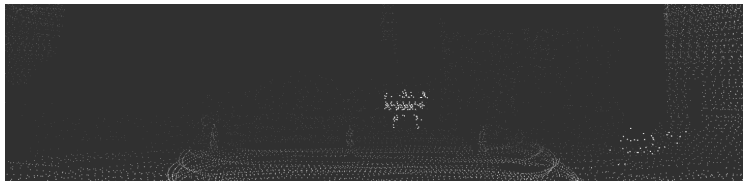


Figure 74: Comparison of trajectories estimated on T1.

(a) Drone detection on a stationary position on the ground before taking off.



(b) Drone detection with target hovering mid-air.



(c) Raw point cloud with target hovering mid-air (0.3 s integration).

Figure 75: Results of the YOLOv5 object detector trained in [135], applied on the generated range images in the outdoors experiment (a)-(b). In (c), we show a sample point cloud with the target hovering in mid-air.

with constants to prevent visual discontinuities. There are two distinct cases that lead to zero-valued pixels after point cloud projection: the sky and other background regions where the emitted laser fails to reflect, and areas within the environment where objects might be present but the LiDAR does not scan. Differentiating between these two cases is crucial for the task of image completion, as it allows for an accurate understanding of the context surrounding the missing pixels.

To remove noise artifacts, we use binary thresholding and a nearest-neighbor interpolation to fill in missing or noisy regions, which results in smoother and more accurate images. Figure 75 showcases the final generated depth map and the result of YOLOv5 object detection: the model was able to accurately detect the UAV and determine its initial position based on these images with no need for further training.

Finally, in Figure 76 we provide a visualization of the tracking algorithm performed with our adaptive method and unknown initial position. Notably, the $z = 0$ line in the figure corresponds to the ground instead of the LiDAR reference line. It should be noted that, due to the outdoor nature of the recorded data, ground truth

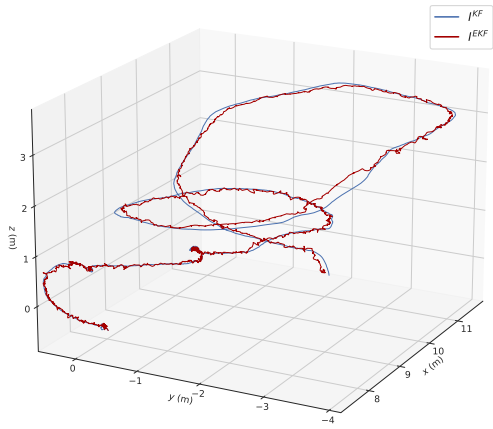


Figure 76: Trajectories estimated with the adaptive method and initialization from range images.

Table 27: Position error (RMSE) for our adaptive integration method for different types of weight functions (N/A when the error diverges because the estimated trajectory is incomplete. Unit: meter)

Track	**No Decay**						**Linear Decay**						**Exponential Decay**					
	I^{KF}	I^{KF}_{AST}	I^{KF}_{ADT}	I^{EKF}	I^{EKF}_{AST}	I^{EKF}_{ADT}	I^{KF}	I^{KF}_{AST}	I^{KF}_{ADT}	I^{EKF}	I^{EKF}_{AST}	I^{EKF}_{ADT}	I^{KF}	I^{KF}_{AST}	I^{KF}_{ADT}	I^{EKF}	I^{EKF}_{AST}	I^{EKF}_{ADT}
T1	0.0788	**0.0773**	0.0882	0.08615	0.087	0.0901	0.0814	0.0783	0.0973	0.0863	0.0886	0.101	0.0782	**0.0773**	0.0845	0.0852	0.087	0.0861
T2	0.023	0.0197	0.0346	0.0375	0.0365	0.0471	0.0309	**0.0195**	0.0439	0.04062	0.0371	0.0565	0.0253	0.0203	0.0335	0.0395	0.0371	0.046
T3	N/A	N/A	N/A	N/A	N/A	N/A	0.0506	0.0438	0.0661	0.0451	0.0444	0.0601	0.0456	0.0431	0.0537	**0.0421**	0.0426	0.0466
T4	N/A	N/A	N/A	0.7311	0.728	0.7409	N/A	N/A	N/A	0.735	0.7298	0.7475	N/A	N/A	N/A	0.0758	**0.0535**	0.1025
T5	0.0585	N/A	0.0872	0.0432	**0.0314**	0.0722	0.0674	N/A	0.1053	0.0503	0.0331	0.0937	0.0551	0.0501	0.0783	0.0377	**0.0314**	0.0624
T6	0.0708	N/A	0.0907	0.0393	**0.0288**	0.067	0.0778	N/A	0.1044	0.0467	0.0311	0.0837	0.0707	N/A	0.0861	0.0373	0.0291	0.0571
T7	N/A	N/A	N/A	0.06	**0.0538**	0.0797	0.09	N/A	N/A	0.0645	0.0551	0.0933	0.0873	N/A	N/A	0.058	**0.0538**	0.0733
T8	N/A	N/A	N/A	N/A	N/A	N/A	0.38	N/A	N/A	N/A	N/A	N/A	0.0939	N/A	0.1154	0.053	**0.0487**	0.0732
T9	N/A	N/A	N/A	0.0629	**0.0534**	0.0882	N/A	N/A	N/A	0.0677	0.0542	0.1058	N/A	N/A	N/A	0.0616	0.0537	0.0813
T10	0.1073	0.1153	0.1136	0.0539	0.0461	0.0955	N/A	0.1178	0.1303	0.0605	0.0475	0.1028	0.0881	0.1167	0.1071	0.0522	**0.0460**	0.073

data is not available.

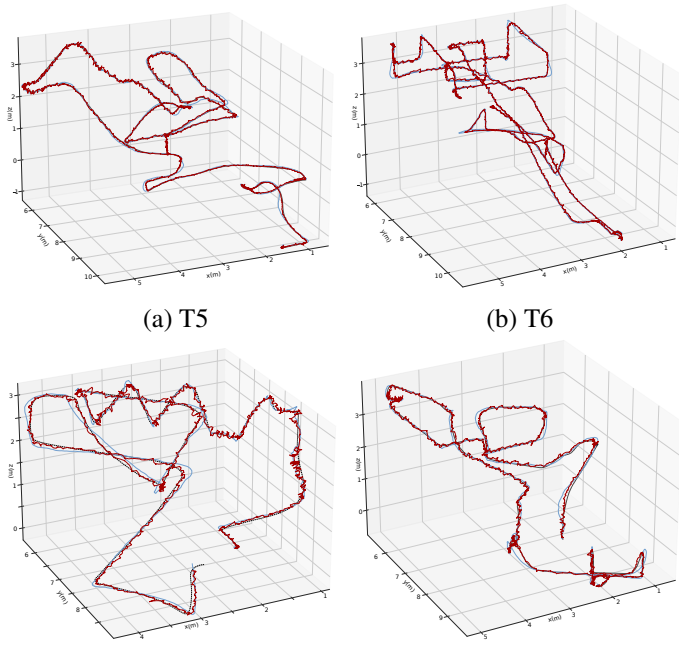
Ablation study

To assess the impact of different weighting functions on the performance of our proposed method, we conducted an ablation study. Specifically, we computed the root mean square error (RMSE) for each track using three different configurations:

(i) No weighting function: this configuration assigns equal weight to all points in the track, without considering their temporal order.

(ii) Linear decay: the weight assigned to each point decreases linearly with time.

(iii) Exponential decay: the weighting function is described in Equation (11).



(a) T5 (b) T6

(c) T8 (d) T10

Figure 77: Visualization of a subset of the trajectories analyzed.

By systematically evaluating the effects of different weighting functions, we have validated the effectiveness of our choice, as presented in Table 27.

The usage of a weight function improves robustness (specifically on tracks T3, T7, T8). In particular, using an exponential weighting function almost always (except for track T2 and T4) reduces the RMSE for the combined estimation and the estimation obtained accumulating a higher number of scans ($I^{\text{EKF}}_{\text{ADT}}$).

Similarly, we conducted ablation studies to verify the effectiveness of the ICI as fusion strategy. We compared it against a simple and a weighted average, where the covariance matrices of each estimator was used to weight its contribution. As shown in Table 28, for 7 out of the 10 tracks analyzed, ICI reports a lower RMSE.

6.3 Summary

First, we have presented a set of methods for detecting and tracking MAVs that are deployed from ground robots, assuming that the initial position is known. The focus has been on the introduction of a novel adaptive LiDAR scan integration method that enables more accurate MAV localization with high-frequency scans, robust and persistent tracking with longer frame integration times, and trajectory validation with low-frequency analysis. Experimental results from different settings confirm the

Table 28: Position error (RMSE) for our adaptive integration method for different types of fusion strategies (N/A when the error diverges because the estimated trajectory is incomplete. Unit: meter)

Track	Simple Average		Weighted Average		ICI	
	I^{KF}	I^{EKF}	I^{KF}	I^{EKF}	I^{KF}	I^{EKF}
T1	0.0795	0.0839	0.0799	0.0867	**0.0782**	0.0852
T2	**0.0235**	0.0361	0.0236	0.0361	0.0253	0.0395
T3	0.0472	**0.041**	0.0479	0.0413	0.0456	0.0421
T4	N/A	**0.733**	N/A	N/A	N/A	0.0758
T5	0.0617	0.0429	0.0634	0.044	0.0551	**0.0377**
T6	N/A	0.0395	N/A	0.0409	0.0707	**0.0373**
T7	N/A	0.0607	N/A	0.0618	0.0873	**0.058**
T8	N/A	N/A	N/A	N/A	0.0939	**0.053**
T9	N/A	0.0650	N/A	0.0667	N/A	**0.0616**
T10	0.0997	0.0554	0.0929	0.0565	0.0881	**0.0522**

better suitability of the different integration times for different scenarios or MAV behaviour, with our adaptive tracking being able to consistently track a MAV in places where a constant lidar scan frequency cannot. Finally, with an additional method to validate the trajectory based on the known shape and size of the MAV, we are able to confirm that the object being tracked meets the dimensional constraints.

After this work, we further presented a novel adaptive tracking approach for UAVs that fuses tracking information from two different scan frequencies from a single solid-state LiDAR sensor. One of the frequencies allows for high accuracy while the second enables more persistent tracking. Our method dynamically adjusts the LiDAR's frame integration time based on the UAV's travelling speed and distance from the sensor, allowing for accurate estimates of the UAV's state.

The experimental results demonstrate that by tailoring the frame integration time to the UAV's movement characteristics, our method outperforms the established baseline method, while also providing more reliable and precise tracking when the estimator from one of the scan frequencies is unavailable or unreliable. The proposed method leverages the Inverse Covariance Intersection method and Kalman filters to enhance tracking accuracy and handle challenging tracking scenarios, suggesting that using the CTRV motion model extended to 3D along with an EKF is a viable option also for UAVs.

In addition, to overcome the challenge of needing a known initial position for detection, we have developed a solution that generates range images in outdoor environments. These range images, created by combining depth and intensity data, can then be used by a YOLOv5 model to accurately detect the UAV's initial position.

7 Federated Learning Enhanced Visual Obstacle Avoidance in a Multi-Robot System

In an autonomous multi-robot system, each robot can have a DL model representing the environment based on their situated awareness, for different purposes, such as visual obstacle avoidance. Different robots may have the limitation of detecting the environment due to their limited resources. It is of great necessity to have a collective model to share their knowledge about the environment. Instead of an individual robot, in a multi-robot system, multiple robots, situated in various whereabouts, collaboratively performing particular tasks is more efficient and of high success rate in heterogeneous environments including unknown ones [271]. However, because of the limits of scenarios where vision-based obstacle avoidance is applied, it is still challenging for DL to collect enough visual inputs for collaborative learning and share them with other agents for privacy and security reasons. To address these concerns, we proposed an approach that included FL, Sim-to-Real (Sim2Real) via a photorealistic simulator, and lifelong learning. A conceptual illustration of such system is illustrated in Figure 3. We study the performance of FL over centralized learning in the simulated and real worlds separately, analyze the improvements of merging data from heterogeneous scenarios, and finally the potential for Sim2Real knowledge transfer.

To the best of our knowledge, this is also the first work to introduce an strategy for continuous learning together with FL and validating it in the real world. In addition to real-world robots, we also utilize simulations to obtain enough data for training the DL models. Due to the proliferation of photorealistic simulators, studies are increasingly relying on these simulators to supplement data collection in situations where we cannot reach or collect sufficient data. In addition, through these types of simulators, the deployment of robotic and autonomous systems can be more accessible. In this way, we believe it may be beneficial for real-world vision-based obstacle avoidance. Throughout their lives, humans and animals can acquire, fine-tune, and transfer knowledge and skills. It is instrumental and interesting for robots to have the same type of learning capabilities. To continually learn the model for obstacle avoidance like humans, we involved multiple robots performing other tasks in different places, including the simulator, into the FL-based lifelong learning system

for data collection, model training, and model sharing. We compare the performance of both AlexNet [272] and ResNet18 [273] (as a lightweight backbone) with centralized and FL approaches to model training. We evaluate the model on different synthetic and real-world datasets, and train the models on combinations of different subsets to better study the effect of environment heterogeneity during the training phase. New data is also acquired with a Clearpath Husky mobile robot while it is operating other tasks and autonomously navigating a new indoors environment. The new data is automatically labelled with additional sensor suited (LiDAR). This opens the door to wider usage of heterogeneous robot fleets where robots with additional sensor capabilities can generate labelled data to train models able to reproduce their autonomous behaviour with more limited sensors, mainly cameras.

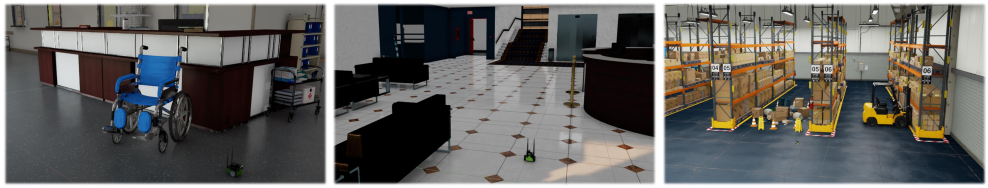
The main contribution of this work is as follows:

1. The design, implementation, deployment, and evaluation of a vision-based DL approach to obstacle avoidance in mobile robots in heterogeneous simulated and real scenarios. We then evaluate the performance benefits of such an approach over offline learning or learning from more limited data sources. We put an emphasis on the benefits when robots are deployed in heterogeneous environments, showing that collaborative learning improves performance even for robots that do not change their environment. For this work, we deploy robots in highly photorealistic and physically-accurate virtual environments and study the ability of such a setup for Sim2Real transfer. (in Section 7.1)

2. We investigated the possibility of federated and continuous learning within hybrid teams of simulated and real robotic agents in this work. We then evaluate the performance benefits of such an approach over offline learning or learning from more limited data sources. First, we evaluate two different deep obstacle avoidance neural networks with both synthetic and real-world data. For both of the architectures a FL-based knowledge sharing method (where the locally trained models are fused) is compared to a centralized training approach (where raw data needs to be aggregated before the training starts). Both models are validated with data from the photorealistic simulator and real-world environment separately. (in Section 7.2)

3. We analyzed the Sim2Real performance of two different deep obstacle avoidance models generated by FL methods, with our results showing that FL outperforms the centralized data aggregation methods. (in Section 7.2)

4. We integrate LiDAR-based navigation for automated labelled data gathering. We implemented an online FL-based visual obstacle avoidance system both in a simulator and real-world environment. With such a system, we can continuously collect data from obstacles and free paths and train the model while the robots operate other tasks. (in Section 7.2)



(a) Hospital (b) Office (c) Warehouse

Figure 78: Customized simulation environments from NVIDIA Isaac Sim

7.1 Federated Learning for Visual Obstacle Avoidance

7.1.1 Methodology

This section covers the different tools, simulation environments and robots utilized in the experiments. We describe the approaches to centralized and FL, and the DL models used for vision-based obstacle avoidance.

Simulation Settings

As a platform to validate the proposed approach in simulated robots, we have used NVIDIA Isaac Sim, powered by Omniverse. Isaac Sim is a scalable robotics simulation application and synthetic data generation tool that enables the creation of photorealistic, physically accurate virtual environments for developing, testing, and managing AI-based robots [274].

We have set up a series of simulation environments to gather data and validate the vision-based approach to obstacle avoidance. Three main environments are used to analyze the performance of the trained model, with a focus on heterogeneity of objects and backgrounds. The datasets used in this study include data from environments that replicate a hospital (see Figure 78a), a office room (see Figure 78b), and a warehouse (see Figure 78c).

To obtain sufficient data for model training, we used the NVIDIA Isaac Sim's Domain Randomization (DR) and Synthetic Data Recorder (SDR) features. By utilizing DR, we can select a specific object and randomly set its properties such as movement, rotation, light, and texture within a defined range. We can easily record the data generated by DR using SDR. These operations were carried out in the three customized environments mentioned previously and pictured in Figure 78. The distribution of the datasets is shown in Table 29, with $\mathcal{S}_i$ representing the dataset associated to environment i.

Table 29: Distribution of simulation datasets

	Hospital ($\mathcal{S}_0$)		**Office** ($\mathcal{S}_1$)		**Warehouse** ($\mathcal{S}_2$)	
	blocked	*free*	*blocked*	*free*	*blocked*	*free*
Prop.	44%	56%	64%	46%	60%	40%
Total	27%		54%		19%	

Real-World Experimental Settings

We utilize three mobile robots for real-world experiments and a local computing server for training the DL models. The platforms used in the experiments are three Jetbot robots from Waveshare (depicted in Figure 79), equipped by default with a wide-angle lens RGB camera and an embedded NVIDIA Jetson Nano development kit. In addition, a Rplidar A1 2D rangefinder has been installed on a 3D printed frame for automated data labeling when real-world data is collected. The local edge server used for training the obstacle avoidance models is equipped with an 8-core Intel i7-9700K processor, 64 GB of memory, and an NVIDIA GeForce RTX 2080 Ti GPU card.

Figure 79: Customized Jetbot platform

We deployed the mobile robots in three different indoor environments (office spaces, hallways and laboratory environments) to validate the obstacle avoidance policies trained with the different approaches. The three rooms vary in terms of objects present as obstacles, material texture, layout, and style. The distribution of the images acquired by the three robots is shown in Table 30, where $\mathcal{R}_{i,\, i\in\{0,1,2\}}$ represent each of the acquired datasets for the respective real-world environments. We intentionally imbalance the data in order to imitate a situation in which robots would not be able to collect data equally across different environments and opera-

Table 30: Distribution of real-world datasets

	Room 1 ($\mathcal{R}_0$)		**Room 2** ($\mathcal{R}_1$)		**Room 3** ($\mathcal{R}_2$)	
	blocked	*free*	*blocked*	*free*	*blocked*	*free*
Prop.	40%	60%	50%	50%	50%	50%
Total	11%		44%		45%	

tional conditions. We also do this to evaluate whether there is a performance impact in environments based on the amount of collected data. As such, *Room 1* only accounts for 11% of the total amount of collected images.

Vision-based Obstacle Avoidance Models

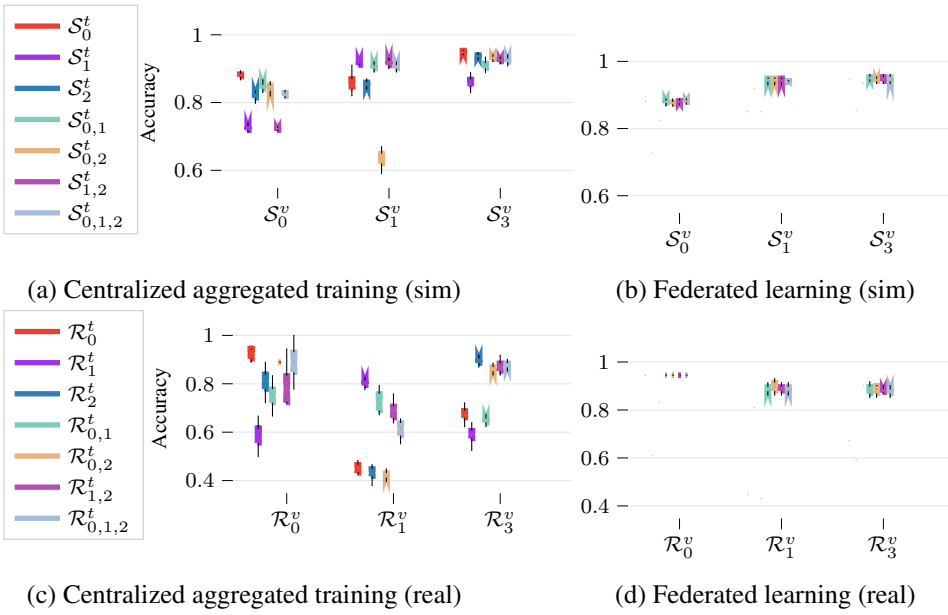
To achieve vision-based obstacle avoidance for the operation of mobile robots, we have chosen to train a generic DL model to assist robots in discriminating between different types of obstacles across heterogeneous environments. This approach comes in contrast to other options, including the detection of individual objects or semantic segmentation (e.g., for segmenting free floor from objects and walls). The selected approach enables us to focus on analyzing the performance of a FL approach and the ability for Sim2Real transfer rather than on the design of a specific obstacle avoidance strategy, which is the main objective of this study

More precisely, we utilized a deep CNN to carry out a vision-based obstacle classifier for only two classes, that define whether the environment ahead is *blocked* or *free* for the robot to navigate. Owing to the relative low level of complexity of the classifier and the limited size of the collected datasets, we have selected the AlexNet architecture as appropriate for such binary classification task. AlexNet has been established as the precedent for deep CNN as one of the most widely used backbones for executing various tasks across multiple domains.

We train different models for each separate dataset with both simulated and real data, as well as combinations of these. The models are trained using two approaches: a centralized learning approach that aggregates data from all robots in the local edge server and trains them at once; and a FL approach that only fuses the individual models trained in each of the different scenarios.

7.1.2 Experimental Results

Through this section, we report the experimental results obtained with data from both simulated and real robots. We show first the performance of the different approaches, with the latter part shifting towards the potential for Sim2Real knowledge transferability.



(a) Centralized aggregated training (sim)

(b) Federated learning (sim)

(c) Centralized aggregated training (real)

(d) Federated learning (real)

Figure 80: Accuracy of the different models obtained through centralized learning with aggregated data or federated learning with fused local models. These results are trained (^{t}) and validated (^{v}) with respective simulation datasets ($\mathcal{S}_i^t$, $\mathcal{S}_i^v$) and real datasets ($\mathcal{R}_i^t$, $\mathcal{R}_i^v$) independently.

Centralized Training vs. Federated Learning

The first objective of our experiments is to analyze the performance improvements that a FL approach brings over a centralized training with traditional data aggregation. To do this, we used the data we collected in the simulated hospital ($\mathcal{S}_0$), office ($\mathcal{S}_1$), and warehouse ($\mathcal{S}_2$) to train our model on each dataset and all possible combinations of two or three of the datasets. Equivalently for the FL approach, we run different training rounds in which we simulate that a different subset of robots is collaboratively learning without sharing any actual raw data. In this approach, only the models are fused and a common model updated iteratively. Figure 80a and Figure 80b report the accuracy of the different models for the centralized and federated approaches, respectively. For the FL results, the training happens only with combination of datasets from different environments.

The accuracy of models trained with real-world data is then shown in Figure 80c and Figure 80d. The data has been obtained with Jetbots navigating in three different office and laboratory indoor environments ($\mathcal{R}_{i,\, i\in 0,1,2}$).

In addition to the accuracy, we also calculate the area under the ROC (Receiver Operating Characteristics) curve (AUC) for each of the scenarios where training is

Table 31: Area under ROC curve (AUC) values for the aggregated centralized learning and federated learning approaches.

Validation datasets		Training datasets: Centralized learning with aggregated data							Training datasets: Federated learning			
		$\mathcal{S}_0^t$	$\mathcal{S}_1^t$	$\mathcal{S}_2^t$	$\mathcal{S}_{0,1}^t$	$\mathcal{S}_{0,2}^t$	$\mathcal{S}_{1,2}^t$	$\mathcal{S}_{1,2,3}^t$	$\mathcal{S}_{0,1}^t$	$\mathcal{S}_{0,2}^t$	$\mathcal{S}_{1,2}^t$	$\mathcal{S}_{1,2,3}^t$
Sim	$\mathcal{S}_0^v$	0.28	0.56	0.56	0.33	0.50	0.71	0.52	**0.85**	**0.85**	**0.85**	**0.85**
Sim	$\mathcal{S}_1^v$	0.33	0.50	0.75	0.33	0.50	0.75	0.60	0.94	0.93	0.93	**0.95**
Sim	$\mathcal{S}_2^v$	0.43	0.94	0.46	0.42	0.50	0.23	0.62	**0.96**	0.95	0.95	0.95
		$\mathcal{R}_0^t$	$\mathcal{R}_1^t$	$\mathcal{R}_2^t$	$\mathcal{R}_{0,1}^t$	$\mathcal{R}_{0,2}^t$	$\mathcal{R}_{1,2}^t$	$\mathcal{R}_{1,2,3}^t$	$\mathcal{R}_{0,1}^t$	$\mathcal{R}_{0,2}^t$	$\mathcal{R}_{1,2}^t$	$\mathcal{R}_{1,2,3}^t$
Real	$\mathcal{R}_0^v$	0.63	0.58	0.42	0.75	0.63	0.08	0.58	**0.88**	**0.88**	**0.88**	**0.88**
Real	$\mathcal{R}_1^v$	0.31	0.66	0.60	0.35	0.25	0.50	0.70	0.83	**0.85**	**0.85**	**0.85**
Real	$\mathcal{R}_2^v$	0.69	0.57	0.87	0.46	0.51	0.48	0.56	0.90	**0.92**	0.90	**0.92**

carried out through either the centralized or federated approaches. The results are reported in Table 31. This metric gives a better understanding of the reliability of the models. In this particular application scenario of robotic navigation, there is indeed a disparity in the cost of false negatives over false positives in terms of the robot's integrity. However, from the point of view of performance, false positives can degrade significantly the navigation speed and time, while low-frequency collisions can be mitigated with, e.g., bumper sensors. Therefore, the classification-threshold invariance of the AUC metric is relevant to this use case.

Through both the accuracy and AUC results, we see that there is a clear improvement when the models and not the data are aggregated. In addition to the better navigation results, this also brings other advantages. First, we optimize the networking resources, allowing for intermittent connectivity and potentially lower bandwidth requirements when the size of the data in training batches is significantly smaller than the models. Moreover, this allows for privacy-preserving collaboration between different end-users or robot operators, as the raw data does not need to be exposed to a central authority or service.

Sim-to-Real Performance Evaluation

In the last part of our experiments, we evaluate the ability of both the centralized and FL approaches to transfer knowledge from simulation environments to the real world. To do this, we rely on the same simulation environments but introduce an independent data validation set ($\mathcal{R}^*$) from a navigation mission across different types of indoor spaces. The accuracy for each of the trained models is shown in Figure 81. We can observe that relatively low performance is achieved with either approach when only one of the simulation environments is used for training the models. This may result from overfitting the model to non-realistic features in the simulated worlds. However, when heterogeneous data is introduced in training, the FL approach sig-

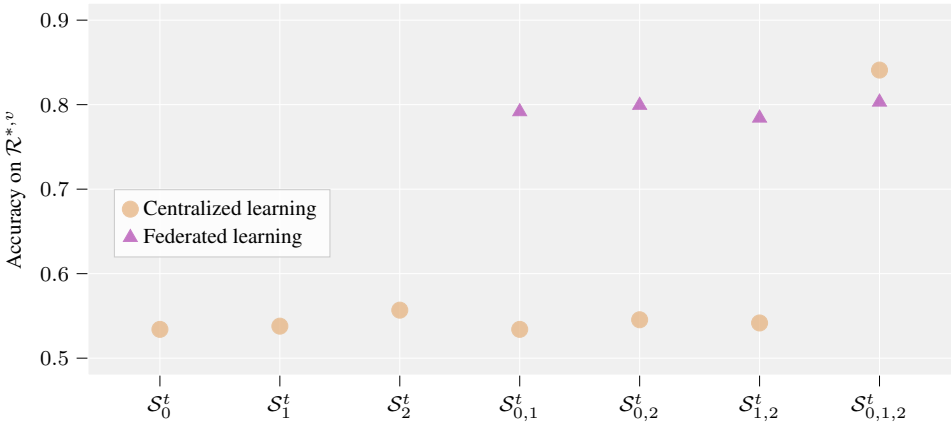


Figure 81: Sim-to-real accuracy of simulation-trained models validated on an independent real-world navigation dataset.

nificantly improves. Our results also show that only when aggregating data from all three simulation environments the centralized learning approach is able to improve the performance to the level of FL. The specific reason behind this behaviour requires further study and will be the object of future research.

7.2 Federated Learning Based Lifelong Learning for Visual Obstacle Avoidance

7.2.1 Methodology

This section outlines the robot platform utilized in this experiment, two different deep neural networks to do vision-based obstacle avoidance for further validation of FL performance, and navigation settings for continual learning.

Data Collection for Federated Learning

The details of the data gathering, including the usage of the photorealistic simulator NVIDIA Isaac Sim, the environment settings, and data distributions, are detailed in Section 7.1, from where we reuse the base simulation and training datasets. The datasets from 7.1 include data from three distinct scenarios in NVIDIA Isaac Sim and from Jetbot robots deployed in three real-world rooms to train and validate the vision-based obstacle avoidance models. In this work, part of the datasets from the simulator and real-world are also $\mathcal{S}_{i,\, i\in\{0,1,2\}}$ and $\mathcal{R}_{i,\, i\in\{0,1,2\}}$, respectively. Additional datasets acquired specifically for this work are introduced in the relevant sections, with Clearpath Husky training data referred to as $\mathcal{H}^S$ or $\mathcal{H}^R$ for the simulated and real robots, respectively.

Regarding data training hardware, in this work, we utilized a Lambda Vector workstation equipped with two RTX 3080 GPU cards and a 24-core AMD Threadripper 3960X processor to train our models for vision-based obstacle avoidance.

Vision-Based Obstacle Avoidance Models

We trained two distinct types of DL models to ensure FL's performance in visual obstacle avoidance and assess how performance differs. These two deep CNN are vision-based obstacle classifiers for two classes that define whether the environment ahead is *blocked* or *free* for the robot to navigate. Owing to the relatively low level of complexity of the classification task and the size of potential datasets for such tasks, we have selected the AlexNet and and ResNet18 architectures as appropriate for such binary classification task.

These two models are generic DL models designed to aid robots in discerning between various types of barriers in heterogeneous situations. This strategy contrasts with other possibilities, such as object detection or semantic segmentation (e.g., segmenting free floor from objects and walls). The chosen approach enables us to concentrate on examining the performance of a federated learning approach and its capacity for Sim2Real transfer rather than on developing a specific obstacle avoidance strategy, which is the study's primary purpose.

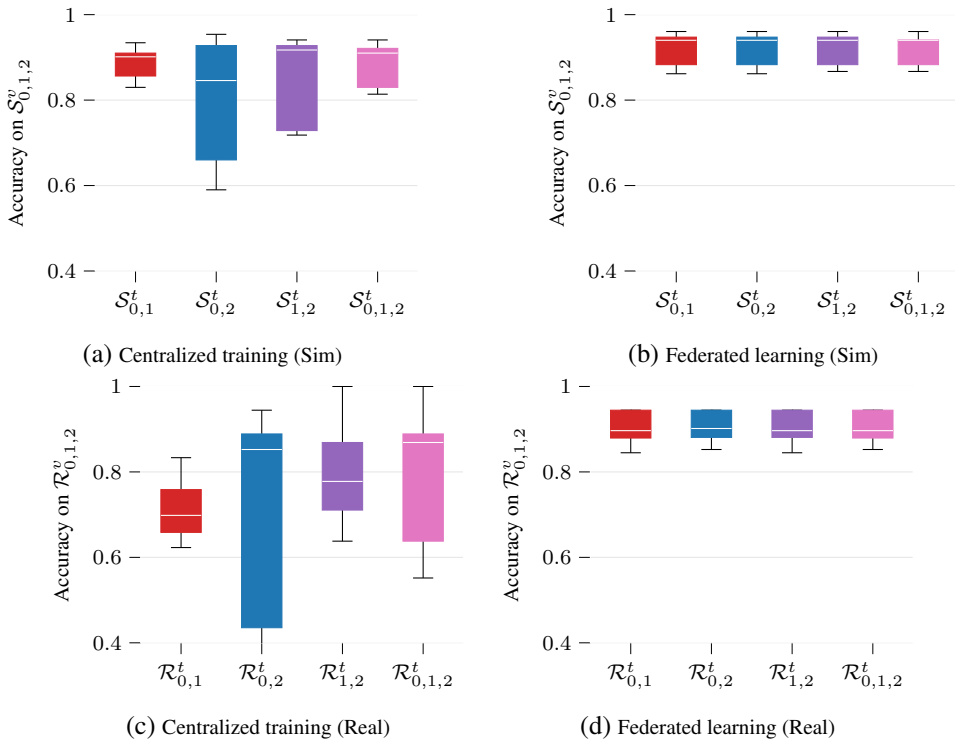
Figure 82: The customized Clearpath Husky platform

Proposed FL Based Lifelong Learning Obstacle Avoidance

By incorporating LiDAR into the Clearpath Husky navigation system, we developed a straightforward FL-based lifelong learning system for visual obstacle avoidance. We used LiDAR to differentiate between blocked and free space of visual data while performing specified navigation tasks. After collecting sufficient data, the robots will train the models independently and send them to the server for aggregation into a global model. Concerning the fusion of models in FL, federated averaging was applied in this work by taking the average of all model updates. The global model can then be deployed to another type of robot for obstacle avoidance performance evaluation.

Regarding the experimental environments, we implemented this with a Clearpath Husky in Figure 82 robot both in a simulator and real-world environments. Our customized Clearpath Husky robot platform is equipped with one Ouster OS0-128 LiDAR for measuring distances to objects when performing autonomous navigation tasks. When the distances are lower to a certain threshold, an Intel Realsense L515 camera will be triggered to capture the images as obstacles included continuously. Otherwise, the images will be categorized to free labels. Once a sufficient number of images are collected, we will train local models based on these data, fuse them to be a global model by FL methods, and then apply the global model to a real-world robot obstacle avoidance operation.

It is worth emphasizing that we used simulated data to help the continuous learning process, as obtaining real-world data is not always straightforward. In the following section, we evaluate the FL fused model's real-world deployment performance using either simulated or real-world data.



(a) Centralized training (Sim)

(b) Federated learning (Sim)

(c) Centralized training (Real)

(d) Federated learning (Real)

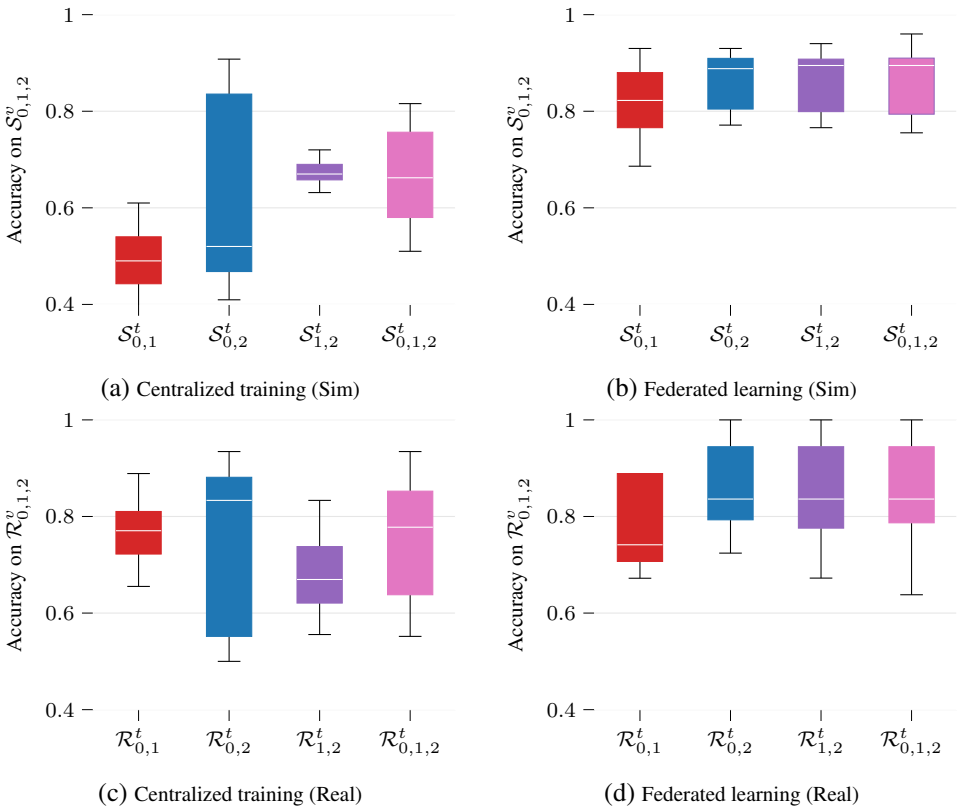
Figure 83: Accuracy of the different models obtained through centralized learning with aggregated data or federated learning with fused local models based on AlexNet. These results are trained (^{t}) and validated (^{v}) with respective simulation datasets ($\mathcal{S}^t_i$, $\mathcal{S}^v_i$) and real datasets ($\mathcal{R}^t_i$, $\mathcal{R}^v_i$) independently.

7.2.2 Experimental Results

This section presents experimental results obtained using data from both simulated and real robots. We first demonstrate the performance of various knowledge-sharing approaches for obstacle avoidance using AlexNet and ResNet18 models before delving into the possibility of Sim2Real knowledge transferability. Finally, we demonstrated the FL method's ability to facilitate lifelong learning by having Husky performing simulation and real-world navigation tasks.

Centralized Training vs. Federated Learning

Our experiments begin by analyzing the performance improvements of federated learning over traditional centralized training with data aggregation. Both simulated and real-world data are evaluated separately in this part. By including the AlextNet and ResNet18 in this study, we can compare how the performance will vary for cen-



(a) Centralized training (Sim)

(b) Federated learning (Sim)

(c) Centralized training (Real)

(d) Federated learning (Real)

Figure 84: Accuracy of models obtained through centralized learning with aggregated data or federated learning with fused local models based on ResNet18. These results are trained (^{t}) and validated (^{v}) with respective simulation datasets ($\mathcal{S}^t_i$, $\mathcal{S}^v_i$) and real datasets ($\mathcal{R}^t_i$, $\mathcal{R}^v_i$) independently.

tralized training and FL using different DL models. To accomplish this, we used the data we collected in the simulated hospital ($\mathcal{S}^t_0$), office ($\mathcal{S}^t_1$), and warehouse ($\mathcal{S}^t_2$) to train our model on each dataset and all possible combinations ($\mathcal{S}^t_{0,1}$, $\mathcal{S}^t_{0,2}$, $\mathcal{S}^t_{1,2}$, $\mathcal{S}^t_{0,1,2}$) of two or three of these datasets and validate the models on $\mathcal{S}^v_{i,\,i\in\{0,1,2\}}$. Equivalently for the federated learning approach, we run different training rounds in which we simulate that a different subset of robots is collaboratively learning without sharing any actual raw data. Only the models are fused in this approach, and a global model is updated iteratively. In the case of real-world data, we repeat the procedure above with $\mathcal{R}^t_0$, $\mathcal{R}^t_1$. $\mathcal{R}^t_2$ representing the training datasets from rooms, office and laboratory and $\mathcal{R}^t_{0,1}$, $\mathcal{R}^t_{0,2}$, $\mathcal{R}^t_{1,2}$, $\mathcal{R}^t_{0,1,2}$ representing the combinations of two or three of the previous sets while $\mathcal{R}^v_{i,\,i\in\{0,1,2\}}$ denotes the corresponding validation data. Figure 83 and Figure 84 report the accuracy of the different models (AlexNet and ResNet18) for centralized learning and FL, respectively. From the results, we found that FL

based approach is robust both in a sim and real separately, and its accuracy is competitive with traditional centralized data aggregation training methods for performing vision-based obstacle avoidance tasks.

Table 32: Area under ROC curve (AUC) values for the aggregated centralized learning and federated learning approaches.

		Training datasets (AlexNet, ResNet18)							
		Centralized learning with aggregated data				Federated learning			
		$\mathcal{S}^t_{0,1}$	$\mathcal{S}^t_{0,2}$	$\mathcal{S}^t_{1,2}$	$\mathcal{S}^t_{1,2,3}$	$\mathcal{S}^t_{0,1}$	$\mathcal{S}^t_{0,2}$	$\mathcal{S}^t_{1,2}$	$\mathcal{S}^t_{1,2,3}$
Validation datasets	Sim								
	$\mathcal{S}^v_0$	(0.33, 0.53)	(0.50, 0.58)	(0.71, 0.56)	(0.52, 0.31)	(**0.85**, 0.74)	(**0.85**, 0.78)	(**0.85**, **0.79**)	(**0.85**, 0.78)
	$\mathcal{S}^v_1$	(0.33, 0.54)	(0.50, 0.40)	(0.75, **0.96**)	(0.60, 0.21)	(0.94, 0.88)	(0.93, 0.89)	(0.93, 0.86)	(**0.95**, 0.89)
	$\mathcal{S}^v_2$	(0.42, 0.64)	(0.50, 0.50)	(0.23, 0.26)	(0.62, 0.23)	(**0.96**, 0.84)	(0.95, 0.92)	(0.95, **0.93**)	(0.95, 0.91)
		$\mathcal{R}^t_{0,1}$	$\mathcal{R}^t_{0,2}$	$\mathcal{R}^t_{1,2}$	$\mathcal{R}^t_{1,2,3}$	$\mathcal{R}^t_{0,1}$	$\mathcal{R}^t_{0,2}$	$\mathcal{R}^t_{1,2}$	$\mathcal{R}^t_{1,2,3}$
	Real								
	$\mathcal{R}^v_0$	(0.75, 0.42)	(0.63, 0.50)	(0.08, 0.33)	(0.58, **1.00**)	(**0.88**, 0.86)	(**0.88**, 0.75)	(**0.88**, 0.94)	(**0.88**, **1.00**)
	$\mathcal{R}^v_1$	(0.35, 0.68)	(0.25, 0.69)	(0.50, 0.48)	(0.70, 0.71)	(0.83, 0.80)	(**0.85**, **0.86**)	(**0.85**, 0.76)	(**0.85**, 0.82)
	$\mathcal{R}^v_2$	(0.46, 0.45)	(0.51, 0.42)	(0.48, 0.47)	(0.56, 0.67)	(0.90, 0.70)	(**0.92**, **0.87**)	(0.90, 0.73)	(**0.92**, 0.82)

Along with accuracy, we also calculate the area under the ROC curve (AUC) for each of the scenarios where training is conducted through either the centralized or federated approaches. The results are summarized in Table 32 where the AUC values for AlexNet-based and ResNet18-based models are enclosed with a bracket in order. This metric enables a better understanding of the models' reliability. In the context of robotic navigation, there is indeed a cost differential between false negatives over false positives in terms of the robots' integrity. However, from the point of view of performance, false positives can significantly degrade the navigation speed and time, while low-frequency collisions can be avoided with other sensors, considering as well that a false negative is not necessarily consistent over time and multiple observations of the same obstacle are processed before a collision may happen.

For both AlexNet and ResNet18, the AUC results together with the accuracy boxplots show that there is a performance boost with the federated learning approach in contrast to the centralized learning method where all data is first aggregated in a single training set.

Sim-to-Real Performance Evaluation

In terms of Sim2Real performance of the two architectures, we analyzed the performance of both the centralized and federated learning approaches to an independent real-world dataset. The results showing the potential for vision-based obstacle avoidance inference for both AlexNet and ResNet18 are shown in Figure 85. The results indicate that both AlexNet and ResNet18-based obstacle avoidance models implemented with the FL approach can outperform the ones with the centralized data aggregation method in terms of Sim2Real performance. Additionally, AlexNet is more suitable for performing obstacle avoidance tasks in our dataset.

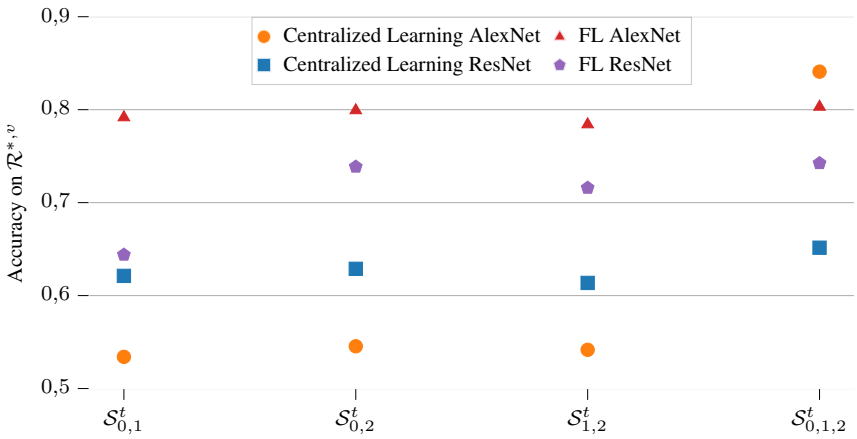
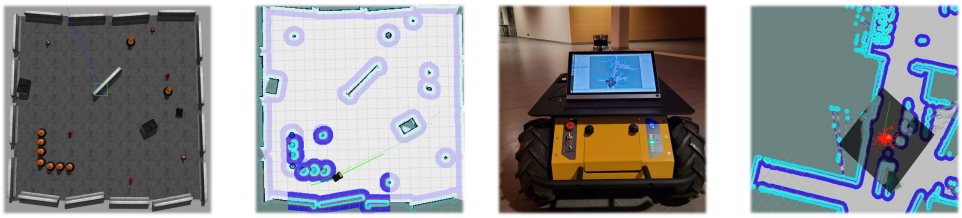


Figure 85: Evaluation of sim-to-real capability of the trained models.



(a) (b) (c) (d)

Figure 86: Husky navigation environment and maps. Figure 86a, 86b, 86c, 86d are the operating environment and the navigation map in the simulator and real-world, respectively.

Federated Continual Learning for Visual-Based Obstacle Avoidance

In this section, we describe the results from experiments where the Husky robot has been utilized both in a simulated playpen (see Figure 86a) and in a large-open indoor office environment (see Figure 86c) to operate autonomous navigation tasks in their navigation maps (see Figure 86b and in Figure 86d, respectively).

At the same time, while the Husky robot was operating autonomous navigation tasks in the sim and real environments, we collected images continuously for training local models for FL models fusion. By validation on an independent dataset $\mathcal{R}^{*,v}$, we evaluated the global models fused with the local model from simulated husky robot $\mathcal{H}^S$ only, the local model from real-world Husky robot $\mathcal{H}^R$ only, and the local models of both them $\mathcal{H}^{S,R}$. The results are shown in Figure 87. Regarding AlexNet, fusing the models either from simulation or real-world can improve the accuracy of

the vision-based obstacle avoidance. However, for ResNet18, our results show that fusing the local model from the simulator can improve the performance of the global model more than fusing the one from the real-world Husky. By fusing both of them, the accuracy can be improved, which shows the potential benefit of collaborative learning from both simulated and real robots.

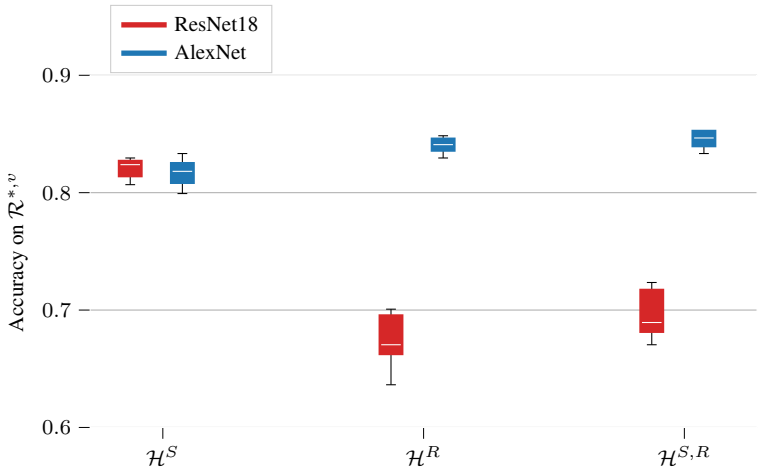


Figure 87: Accuracy of FL fused models based on images collected while Husky was operating navigation tasks in simulated and real-world environments for continual learning purposes

7.3 Summary

7.3.1 Federated Learning Enhanced Visual Obstacle Avoidance

We have presented a federated learning approach for vision-based obstacle avoidance in mobile robots that leverages data from both simulated agents and real robots with additional sensors. We have shown that interconnected robots relying on deep learning for vision-based navigation can aid each other without sharing raw data. Specifically, we show how training the same model with data from heterogeneous environments improves performance across the simulated and real worlds. More importantly, the performance improvements are better when the models are trained through a federated learning approach compared to centralized learning. In addition to the application-specific improvements, the federated learning approach brings inherent benefits in terms of communication optimization and preservation of data privacy, enabling collaboration across organizations or users. Finally, we have shown that the presented approach is able to transfer knowledge from simulation to reality effectively.

Owing to the potential for Sim2Real transfer and accounting for the better performance of federated over centralized learning, future work will be directed towards lifelong FL through a combination of simulated and real agents. Additionally, we will further explore the reasons behind the differences in performance across the approaches presented in this manuscript.

7.3.2 LiDAR Assisted Federated Lifelong Learning for Visual Obstacle Avoidance

This work presents an FL-based lifelong learning method for vision-based obstacle avoidance among various mobile robots involving both simulated and real-world environments. Rather than applying a single deep neural network, we analyzed the performance of the FL-based method compared with the centralized data aggregation method with two different deep neural networks, providing to better generalize results. More specifically, we found that the FL approach can bring competitive accuracy compared to centralized learning across the simulated and real worlds while also delivering inherent benefits in communication optimization and data privacy preservation, enabling collaboration across organizations or users. Additionally, we evaluated the FL method's Sim2Real vision-based obstacle avoidance performance. The result indicates that transferring the obstacle knowledge from simulation to reality using the FL method is more effective and stable. Within the FL-based lifelong learning system, one agent can improve its obstacle avoidance performance by aggregating models from local models in other agents situated either in simulated or real-world environments in our study.

8 Conclusion and Future Work

This thesis is designated to study the integration of the advanced multi-modal sensors for high-level robotic sensing and perception and meanwhile bring the privacy-preserving by utilizing Federated Learning into the process.

8.1 Conclusion

In this thesis, we first introduced novel datasets featuring multi-modal sensors for various applications, including SLAM, LiDAR odometry, localization, and UAV tracking. These datasets encompass a variety of LiDARs, such as spinning and solid-state LiDARs, and provide ground truth data via MOCAP, GNSS/RTK, and SLAM-assisted methods. Additionally, we reviewed state-of-the-art localization and tracking approaches.

Following the dataset introduction, we explored innovative multi-robot relative positioning methods to enhance robotic perception tasks, emphasizing UWB-based techniques. We evaluated UWB-based relative localization between UAVs and ground robots through simulations and outdoor experiments in urban environments with poor GNSS performance. The analysis revealed that UWB-based systems can serve as alternatives to RTK-GNSS, maintaining adequate accuracy for autonomous flight even when GNSS accuracy is insufficient.

Further research introduced a particle-filter-based approach for relative multi-robot localization integrating UWB ranges, robot odometry, and cooperative spatial detections. This method uses LSTM networks to predict ranging errors, improving accuracy and real-time performance. Unlike VIO-based approaches, our method employs cameras solely for cooperative spatial detections. Experiments demonstrated that this approach outperforms traditional multilateration in relative state estimation, with low CPU usage and memory consumption.

We also analyzed the performance of various deep learning models for object detection and semantic segmentation on LiDAR-generated images, collected in different environments and lighting conditions. Results indicated that state-of-the-art DL models perform well with this data type, suggesting expanded applications for LiDAR beyond current geometric methods. A key limitation noted was the lack of re-training with larger annotated datasets.

To address computational overhead in point cloud registration for LiDAR odom-

etry, we proposed a novel method using LiDAR-generated images. This approach preserves key points during down-sampling, maintaining performance comparable to using full raw point clouds, and proving more effective in scenarios prone to drift and rotational transformations.

Additionally, we presented a novel UAV tracking method using fused signal images and point clouds from an Ouster LiDAR, which does not require calibration with external cameras. Indoor experiments demonstrated the effectiveness of this approach compared to methods using only point clouds or signal images, and its feasibility on a Jetson Nano platform.

Finally, we proposed a federated learning (FL) approach for vision-based obstacle avoidance in mobile robots, leveraging data from both simulated and real environments. This method showed better performance improvements and communication optimization compared to centralized learning, preserving data privacy and enabling cross-organizational collaboration. Our FL-based lifelong learning system effectively transferred obstacle knowledge from simulation to reality, improving obstacle avoidance performance through model aggregation.

8.2 Future Work

The future work of the dissertation can be from different aspects. Instead of the benchmark tests only focus on SLAM algorithms based on spinning LiDAR and solid-state LiDAR, we can add benchmark tests based on cameras and even SLAM algorithms based on multiple sensor fusions in the future.

Regarding LiDAR as a camera, we can explore a wider variety of preprocessing techniques and study the performance benefits of re-training some of the studied network architectures with data from the LiDAR camera sensors. Furthermore, there is potential to seamlessly integrate the current LiDAR-generated image keypoint extraction process into the broader SLAM pipeline. For instance, one avenue of exploration could involve amalgamating features extracted from LiDAR-generated images with those derived from point cloud data, facilitating the development of a lightweight SLAM system complemented by additional sensors, such as an IMU.

In future work, we will concentrate on utilizing and adapting the FL-based lifelong learning system to perform other robotic navigation tasks with a view toward sim-to-real capabilities. We also find potential in dynamically adjusting the simulation environments based on real-world robot experiences, e.g., adding 3D models of new objects found.

List of References

[1] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296. IEEE, 2018.

[2] Xu Liu, Guilherme V Nardari, Fernando Cladera Ojeda, Yuezhan Tao, Alex Zhou, Thomas Donnelly, Chao Qu, Steven W Chen, Roseli AF Romero, Camillo J Taylor, and Vijay Kumar. Large-scale autonomous flight with real-time semantic slam under dense forest canopy. *IEEE Robotics and Automation Letters (RA-L)*, 2022.

[3] Qingqing Li, Jorge Peña Queralta, Tuan Nguyen Gia, Zhuo Zou, and Tomi Westerlund. Multi-sensor fusion for navigation and mapping in autonomous vehicles: Accurate localization in urban environments. *Unmanned Systems*, 8(03):229–237, 2020.

[4] Nina Varney, Vijayan K Asari, and Quinn Graehling. Dales: a large-scale aerial lidar data set for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 186–187, 2020.

[5] Juntao Yang, Zhizhong Kang, Sai Cheng, Zhou Yang, and Perpetual Hope Akwensi. An individual tree segmentation method based on watershed algorithm and three-dimensional spatial distribution analysis from airborne lidar point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:1055–1067, 2020.

[6] Jiarong Lin and Fu Zhang. Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3126–3131. IEEE, 2020.

[7] Kailai Li, Meng Li, and Uwe D Hanebeck. Towards high-performance solid-state-lidar-inertial odometry and mapping. *IEEE Robotics and Automation Letters*, 6(3):5167–5174, 2021.

[8] Zheng Liu, Fu Zhang, and Xiaoping Hong. Low-cost retina-like robotic lidars based on incommensurable scanning. *IEEE/ASME Transactions on Mechatronics*, 2021.

[9] Jin Hyeok Yoo, Yecheol Kim, Jisong Kim, and Jun Won Choi. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. In *European Conference on Computer Vision*, pages 720–736. Springer, 2020.

[10] Huazan Zhong, Hao Wang, Zhengrong Wu, Chen Zhang, Yongwei Zheng, and Tao Tang. A survey of lidar and camera fusion enhancement. *Procedia Computer Science*, 183:579–588, 2021.

[11] Maria Tsiourva and Christos Papachristos. Lidar imaging-based attentive perception. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 622–626. IEEE, 2020.

[12] Ardi Tampuu, Romet Aidla, Jan Are van Gent, and Tambet Matiisen. Lidar-as-camera for end-to-end driving. *arXiv preprint arXiv:2206.15170*, 2022.

[13] Yu Xianjia, Li Qingqing, Jorge Peña Queralta, Jukka Heikkonen, and Tomi Westerlund. Applications of uwb networks and positioning to autonomous robots and industrial systems. In *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–6. IEEE, 2021.

[14] Dae-Ho Kim, Arshad Farhad, and Jae-Young Pyun. Uwb positioning system based on lstm classification with mitigated nlos effects. *IEEE Internet of Things Journal*, 2022.

[15] Hao Xu, Yichen Zhang, Boyu Zhou, Luqi Wang, Xinjie Yao, Guotao Meng, and Shaojie Shen.

Omni-swarm: A decentralized omnidirectional visual–inertial–uwb state estimation system for aerial swarms. *IEEE Transactions on Robotics*, 2022.

[16] Zhiren Xun, Jian Huang, Zhehan Li, Chao Xu, Fei Gao, and Yanjun Cao. Crepes: Cooperative relative pose estimation towards real-world multi-robot systems. *arXiv preprint arXiv:2302.01036*, 2023.

[17] Yu Xianjia, Li Qingqing, Jorge Pena Queralta, Jukka Heikkonen, and Tomi Westerlund. Cooperative uwb-based localization for outdoors positioning and navigation of uavs aided by ground robots. In *2021 IEEE International Conference on Autonomous Systems (ICAS)*, pages 1–5. IEEE, 2021.

[18] Kun Zhang, Chong Shen, Qun Zhou, Haifeng Wang, Qian Gao, and Yushan Chen. A combined gps uwb and marg locationing algorithm for indoor and outdoor mixed scenario. *Cluster Computing*, 22(3):5965–5974, 2019.

[19] Hao Xu, Luqi Wang, Yichen Zhang, Kejie Qiu, and Shaojie Shen. Decentralized visual-inertial-uwb fusion for relative state estimation of aerial swarm. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8776–8782. IEEE, 2020.

[20] Tianxia Liu, Bofeng Li, Ling Yang, Jing Qiao, Wu Chen, et al. Tightly coupled integration of gnss/uwb/vio for reliable and seamless positioning. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[21] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[22] Jorge Peña Queralta, Tuan Nguyen Gia, Hannu Tenhunen, and Tomi Westerlund. Edge-ai in lora-based health monitoring: Fall detection system with fog computing and lstm recurrent neural networks. In *2019 42nd international conference on telecommunications and signal processing (TSP)*, pages 601–604. IEEE, 2019.

[23] Aly Metwaly, Jorge Peña Queralta, Victor Kathan Sarker, Tuan Nguyen Gia, Omar Nasir, and Tomi Westerlund. Edge computing with embedded ai: Thermal image analysis for occupancy estimation in intelligent buildings. In *Proceedings of the INTelligent Embedded Systems Architectures and Applications Workshop 2019*, pages 1–6, 2019.

[24] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[25] Yi Liu, Xingliang Yuan, Zehui Xiong, Jiawen Kang, Xiaofei Wang, and Dusit Niyato. Federated learning for 6g communications: Challenges, methods, and future directions. *China Communications*, 17(9):105–118, 2020.

[26] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[27] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.

[28] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition(CVPR)*, pages 11621–11631, 2020.

[29] Zhi Yan, Li Sun, Tomáš Krajník, and Yassine Ruichek. Eu long-term dataset with multiple sensors for autonomous driving. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10697–10704. IEEE, 2020.

[30] Nicholas Carlevaris-Bianco, Arash K Ushani, and Ryan M Eustice. University of michigan north campus long-term vision and lidar dataset. *The International Journal of Robotics Research*, 35 (9):1023–1035, 2016.

[31] Milad Ramezani, Yiduo Wang, Marco Camurri, David Wisth, Matias Mattamala, and Maurice Fallon. The newer college dataset: Handheld lidar, inertial and vision with ground truth. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4353–4360. IEEE, 2020.

[32] Jie Yin, Ang Li, Tao Li, Wenxian Yu, and Danping Zou. M2dgr: A multi-sensor and multi-scenario slam dataset for ground robots. *IEEE Robotics and Automation Letters*, 7(2):2266–2273, 2021.

[33] Maggie Wigness, Sungmin Eum, John G Rogers, David Han, and Heesung Kwon. A rugd dataset for autonomous navigation and visual perception in unstructured outdoor environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5000–5007. IEEE, 2019.

[34] Chaoyue Niu, Danesh Tarapore, and Klaus-Peter Zauner. Low-viewpoint forest depth dataset for sparse rover swarms. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8035–8040. IEEE, 2020.

[35] Gabriela Gresenz, Jules White, and Douglas C Schmidt. An off-road terrain dataset including images labeled with measures of terrain roughness. In *2021 IEEE International Conference on Autonomous Systems (ICAS)*, pages 1–5. IEEE, 2021.

[36] Zachary Pezzementi, Trenton Tabor, Peiyun Hu, Jonathan K Chang, Deva Ramanan, Carl Wellington, Benzun P Wisely Babu, and Herman Herman. Comparing apples and oranges: Off-road pedestrian detection on the national robotics engineering center agricultural person-detection dataset. *Journal of Field Robotics*, 35(4):545–563, 2018.

[37] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. Pandaset: Advanced sensor suite dataset for autonomous driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3095–3101. IEEE, 2021.

[38] Dávid Rozenberszki and András L Majdik. Lol: Lidar-only odometry and localization in 3d point cloud maps. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4379–4385. IEEE, 2020.

[39] Weikun Zhen, Sam Zeng, and Sebastian Soberer. Robust localization and localizability estimation with a rotating laser scanner. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 6240–6245. IEEE, 2017.

[40] Haoyang Ye, Yuying Chen, and Ming Liu. Tightly coupled 3d lidar inertial odometry and mapping. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3144–3150. IEEE, 2019.

[41] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 2014.

[42] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[43] Qingqing Li, Paavo Nevalainen, Jorge Peña Queralta, Jukka Heikkonen, and Tomi Westerlund. Localization in unstructured environments: Towards autonomous robots in forests with delaunay triangulation. *Remote Sensing*, 12(11):1870, 2020.

[44] Paavo Nevalainen, Parisa Movahedi, Jorge Peña Queralta, Tomi Westerlund, and Jukka Heikkonen. Long-term autonomy in forest environment using self-corrective slam. In *New Developments and Environmental Applications of Drones*, pages 83–107. Springer, 2022.

[45] Wei Xu and Fu Zhang. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robotics and Automation Letters*, 6(2):3317–3324, 2021.

[46] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 2022.

[47] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 5135–5142. IEEE, 2020.

[48] Qingqing Li, Xianjia Yu, Jorge Peña Queralta, and Tomi Westerlund. Multi-modal lidar dataset for benchmarking general-purpose localization and mapping algorithms. *arXiv preprint arXiv:2203.03454*, 2022.

[49] J. Lin *et al.* Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3126–3131. IEEE, 2020.

[50] Kailai Li, Meng Li, and Uwe D Hanebeck. Towards high-performance solid-state-lidar-inertial odometry and mapping. *IEEE Robotics and Automation Letters*, 6(3):5167–5174, 2021.

[51] Jiarong Lin and Fu Zhang. R 3 live: A robust, real-time, rgb-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10672–10678. IEEE, 2022.

[52] Thien-Minh Nguyen, Muqing Cao, Shenghai Yuan, Yang Lyu, Thien Hoang Nguyen, and Lihua Xie. Viral-fusion: A visual-inertial-ranging-lidar sensor fusion approach. *IEEE Transactions on Robotics*, 38(2):958–977, 2021.

[53] Jiarong Lin, Xiyuan Liu, and Fu Zhang. A decentralized framework for simultaneous calibration, localization and mapping with multiple lidars. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4870–4877. IEEE, 2020.

[54] Zheng Liu and Fu Zhang. Balm: Bundle adjustment for lidar mapping. *IEEE Robotics and Automation Letters*, 6(2):3184–3191, 2021.

[55] Wei Xu and Fu Zhang. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robotics and Automation Letters*, 6(2):3317–3324, 2021.

[56] Yuewen Zhu, Chunran Zheng, Chongjian Yuan, Xu Huang, and Xiaoping Hong. Camvox: A low-cost and accurate lidar-assisted visual slam system. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5049–5055. IEEE, 2021.

[57] David M Rosen, Kevin J Doherty, Antonio Terán Espinoza, and John J Leonard. Advances in inference and representation for simultaneous localization and mapping. *Annual Review of Control, Robotics, and Autonomous Systems*, 4, 2021.

[58] Fuhu Che, Abbas Ahmed, Qasim Zeeshan Ahmed, Syed Ali Raza Zaidi, and Muhammad Zeeshan Shakir. Machine learning based approach for indoor localization using ultra-wide bandwidth (uwb) system for industrial internet of things (iiot). In *International Conference on UK-China Emerging Technologies (UCET)*, pages 1–4. IEEE, 2020.

[59] Kasper Støy. Using situated communication in distributed autonomous mobile robotics. In *SCAI*, volume 1, pages 44–52. Citeseer, 2001.

[60] Hessam Mohammadmoradi, Milad Heydariaan, and Omprakash Gnawali. Srac: Simultaneous ranging and communication in uwb networks. In *15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 9–16. IEEE, 2019.

[61] Wang Shule, Carmen Martínez Almansa, Jorge Peña Queralta, Zhuo Zou, and Tomi Westerlund. Uwb-based localization for multi-uav systems and collaborative heterogeneous multi-robot systems. *Procedia Computer Science*, 175:357–364, 2020.

[62] Zhuo Zou. *Impulse radio UWB for the internet-of-things: a study on UHF/UWB hybrid solution*. PhD thesis, KTH Royal Institute of Technology, 2011.

[63] Matteo Ridolfi, Stef Vandermeeren, Jense Defraye, Heidi Steendam, Joeri Gerlo, Dirk De Clercq, Jeroen Hoebeke, and Eli De Poorter. Experimental evaluation of uwb indoor positioning for sport postures. *Sensors*, 18(1):168, 2018.

[64] Abdulrahman Alarifi, AbdulMalik Al-Salman, Mansour Alsaleh, Ahmad Alnafessah, Suheer Al-Hadhrami, Mai A Al-Ammar, and Hend S Al-Khalifa. Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, 16(5):707, 2016.

[65] J. Peña Queralta, C. Martínez Almansa, F. Schiano, D. Floreano, and T. Westerlund. Uwb-based system for uav localization in gnss-denied environments: Characterization and dataset. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4521–4528, 2020. doi: 10.1109/IROS45743.2020.9341042.

[66] Nicola Macoir, Jan Bauwens, Bart Jooris, Ben Van Herbruggen, Jen Rossey, Jeroen Hoebeke, and Eli De Poorter. Uwb localization with battery-powered wireless backbone for drone-based inventory management. *Sensors*, 19(3):467, 2019.

[67] Thien-Minh Nguyen, Abdul Hanif Zaini, Chen Wang, Kexin Guo, and Lihua Xie. Robust target-relative localization with ultra-wideband ranging and communication. In *IEEE international conference on robotics and automation (ICRA)*, pages 2312–2319. IEEE, 2018.

[68] Dušan Kocur and Mária Švecová. Signal processing for monitoring of static persons using uwb sensors: A survey. In *IEEE MTT-S International Microwave Biomedical Conference (IMBioC)*, volume 1, pages 1–3. IEEE, 2019.

[69] Shivani Soni, Meha Shrivastava, and Abhishek Agwekar. A detail concept and survey on ultra wide band (uwb) antennas. In *2nd International Conference on Data, Engineering and Applications (IDEA)*, pages 1–7. IEEE, 2020.

[70] Rovin Tiwari, Raghavendra Sharma, and Rahul Dubey. A survey on uwb wearable antenna for body area network application. *International Journal of Engineering Technologies and Management Research*, 5(2):274–280, 2018.

[71] Yi Fang, Guojun Han, Pingping Chen, Francis CM Lau, Guanrong Chen, and Lin Wang. A survey on dcsk-based communication systems and their application to uwb scenarios. *IEEE Communications Surveys & Tutorials*, 18(3):1804–1837, 2016.

[72] Abdulrahman *et al.* A. Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, 16(5), 2016. ISSN 1424-8220. doi: 10.3390/s16050707. URL `https://www.mdpi.com/1424-8220/16/5/707`.

[73] Carmen Martínez Almansa, Wang Shule, Jorge Peña Queralta, and Tomi Westerlund. Autocalibration of a mobile uwb localization system for ad-hoc multi-robot deployments in gnss-denied environments. *arXiv preprint arXiv:2004.06762*, 2020.

[74] A. Prorok, L. Gonon, and A. Martinoli. Online model estimation of ultra-wideband tdoa measurements for mobile robot localization. In *IEEE International Conference on Robotics and Automation*, pages 807–814, 2012. doi: 10.1109/ICRA.2012.6224869.

[75] Janis Tiemann, Andrew Ramsey, and Christian Wietfeld. Enhanced uav indoor navigation through slam-augmented uwb localization. In *IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2018.

[76] Yang Song, Mingyang Guan, Wee Peng Tay, Choi Look Law, and Changyun Wen. Uwb/lidar fusion for cooperative range-only slam. In *international conference on robotics and automation (ICRA)*, pages 6568–6574. IEEE, 2019.

[77] Weikun Zhen and Sebastian Scherer. Estimating the localizability in tunnel-like environments using lidar and uwb. In *International Conference on Robotics and Automation (ICRA)*, pages 4903–4908. IEEE, 2019.

[78] Chiara Bonsignori, Fabio Condomitti, Marco Del Gamba, Federico Garzelli, Leonardo Lossi, Francesco Mione, Alessandro Noferi, and Alessio Vecchio. Estimation of user's orientation via wearable uwb. In *16th International Conference on Intelligent Environments (IE)*, pages 80–83. IEEE, 2020.

[79] Yu-Yao Chen, Shih-Ping Huang, Ting-Wei Wu, Wei-Ting Tsai, Chong-Yi Liou, and Shau-Gang Mao. Uwb system for indoor positioning and tracking with arbitrary target orientation, optimal anchor location, and adaptive nlos mitigation. *IEEE Transactions on Vehicular Technology*, 69 (9):9304–9314, 2020.

[80] Ernst-Johann Theussl, Dimitar Ninevski, and Paul O'Leary. Measurement of relative position and orientation using uwb. In *IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6. IEEE, 2019.

[81] Francisco Molina Martel, Juri Sidorenko, Christoph Bodensteiner, Michael Arens, and Urs Hugentobler. Unique 4-dof relative pose estimation with six distances for uwb/v-slam-based devices. *Sensors*, 19(20):4366, 2019.

[82] Meng-Gang Li, Hua Zhu, Shao-Ze You, and Chao-Quan Tang. Uwb-based localization system aided with inertial sensor for underground coal mine applications. *IEEE Sensors Journal*, 20 (12):6652–6669, 2020.

[83] Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4):837–855, 2018.

[84] Kexin Guo, Zhirong Qiu, Cunxiao Miao, Abdul Hanif Zaini, Chun-Lin Chen, Wei Meng, and Lihua Xie. Ultra-wideband-based localization for quadcopter navigation. *Unmanned Systems*, 4 (01):23–34, 2016.

[85] Li Qingqing, Yu Xianjia, Jorge Pena Queralta, and Tomi Westerlund. Adaptive lidar scan frame integration: Tracking known mavs in 3d point clouds. *arXiv preprint arXiv:2103.04069*, 2021.

[86] Leehter Yao, Yeong-Wei Andy Wu, Lei Yao, and Zhe Zheng Liao. An integrated imu and uwb sensor based indoor positioning system. In *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8. IEEE, 2017.

[87] Weiwei Kong, Daibing Zhang, and Jianwei Zhang. A ground-based multi-sensor system for autonomous landing of a fixed wing uav. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1303–1310. IEEE, 2015.

[88] Wahab Khawaja, Ozgur Ozdemir, Fatih Erden, Ismail Guvenc, and David W Matolak. Uwb air-to-ground propagation channel measurements and modeling using uavs. In *IEEE Aerospace Conference*, pages 1–10. IEEE, 2019.

[89] Yang Qi, Yisheng Zhong, and Zongying Shi. Cooperative 3-d relative localization for uav swarm by fusing uwb with imu and gps. In *Journal of Physics: Conference Series*, volume 1642, page 012028. IOP Publishing, 2020.

[90] J. Li, Y. Bi, K. Li, K. Wang, F. Lin, and B. M. Chen. Accurate 3d localization for mav swarms by uwb and imu fusion. In *IEEE 14th International Conference on Control and Automation (ICCA)*, pages 100–105, 2018. doi: 10.1109/ICCA.2018.8444329.

[91] Thien-Minh Nguyen, Thien Hoang Nguyen, Muqing Cao, Zhirong Qiu, and Lihua Xie. Integrated uwb-vision approach for autonomous docking of uavs in gps-denied environments. In *International Conference on Robotics and Automation (ICRA)*, pages 9603–9609. IEEE, 2019.

[92] Ran Liu, Chau Yuen, Tri-Nhut Do, Dewei Jiao, Xiang Liu, and U-Xuan Tan. Cooperative relative positioning of mobile users by fusing imu inertial and uwb ranging information. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5623–5629. IEEE, 2017.

[93] Meng-gang Li, Hua Zhu, Shao-ze You, Lei Wang, Zheng Zhang, and Chao-quan Tang. Imu-aided ultra-wideband based localization for coal mine robots. In *International Conference on Intelligent Robotics and Applications*, pages 256–268. Springer, 2019.

[94] Daquan Feng, Chunqi Wang, Chunlong He, Yuan Zhuang, and Xiang-Gen Xia. Kalman-filter-based integration of imu and uwb for high-accuracy indoor positioning and navigation. *IEEE Internet of Things Journal*, 7(4):3133–3146, 2020.

[95] Thien-Minh Nguyen, Shenghai Yuan, Muqing Cao, Yang Lyu, Thien Hoang Nguyen, and Lihua Xie. Viral-fusion: A visual-inertial-ranging-lidar sensor fusion approach. *arXiv preprint arXiv:2010.12274*, 2020.

[96] Hao Xu, Yichen Zhang, Boyu Zhou, Luqi Wang, and Shaojie Shen. Omni-swarm: An aerial swarm system with decentralized omni-directional visual-inertial-uwb state estimation. *arXiv preprint arXiv:2103.04131*, 2021.

[97] Qin Shi, Xiaowei Cui, Wei Li, Yu Xia, and Mingquan Lu. Visual-uwb navigation system for unknown environments. In *the 31st International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2018)*, pages 3111–3121, 2018.

[98] Jorge Peña Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju, and Tomi Westerlund. Collaborative multi-robot search and rescue: Planning, coordination, perception and active vision. *IEEE Access*, 8:191617–191643, 2020. doi: https://ieeexplore.ieee.org/document/9220149?source=authoralert.

[99] Nikhil Nigam, Stefan Bieniawski, Ilan Kroo, and John Vian. Control of multiple uavs for persistent surveillance: Algorithm and flight test results. *IEEE Transactions on Control Systems Technology*, 20(5):1236–1251, 2011.

[100] Jorge Pena Queralta, Li Qingqing, Fabrizio Schiano, and Tomi Westerlund. Vio-uwb-based collaborative localization and dense scene reconstruction within heterogeneous multi-robot systems. *arXiv preprint arXiv:2011.00830*, 2020.

[101] Jorge Peña Queralta, Jenni Raitoharju, Tuan Nguyen Gia, Nikolaos Passalis, and Tomi Westerlund. Autosos: Towards multi-uav systems supporting maritime search and rescue with lightweight ai and edge computing. *arXiv preprint arXiv:2005.03409*, 2020.

[102] Jorge Peña Queralta, Li Qingqing, Tuan Nguyen Gia, Hong-Linh Truong, and Tomi Westerlund. End-to-end design for self-reconfigurable heterogeneous robotic swarms. IEEE, 2020. doi: https://doi.org/10.1109/DCOSS49796.2020.00052.

[103] Vinod Kristem, Somasundaram Niranjayan, Seun Sangodoyin, and Andreas F Molisch. Experimental determination of uwb ranging errors in an outdoor environment. In *ieee international conference on communications (icc)*, pages 4838–4843. IEEE, 2014.

[104] Vinod Kristem, Andreas F Molisch, Somasundaram Niranjayan, and Seun Sangodoyin. Coherent uwb ranging in the presence of multiuser interference. *IEEE transactions on wireless communications*, 13(8):4424–4439, 2014.

[105] Juan Antonio Corrales, FA Candelas, and Fernando Torres. Hybrid tracking of human operators using imu/uwb data fusion by a kalman filter. In *3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 193–200. IEEE, 2008.

[106] Luzheng Bi, Cuntai Guan, et al. A review on emg-based motor intention prediction of continuous human upper limb motion for human-robot collaboration. *Biomedical Signal Processing and Control*, 51:113–127, 2019.

[107] Pedro Neto, Miguel Simão, Nuno Mendes, and Mohammad Safeea. Gesture-based human-robot interaction for human assistance in manufacturing. *The International Journal of Advanced Manufacturing Technology*, 101(1):119–135, 2019.

[108] Bernhard Groβwindhager, Michael Stocker, Michael Rath, Carlo Alberto Boano, and Kay Römer. Snaploc: An ultra-fast uwb-based indoor localization system for an unlimited number of tags. In *18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 61–72. IEEE, 2019.

[109] Milad Heydariaan, Hossein Dabirian, and Omprakash Gnawali. Anguloc: Concurrent angle of arrival estimation for indoor localization with uwb radios. In *16th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 112–119. IEEE, 2020.

[110] Michael Stocker, Bernhard Großwindhager, Carlo Alberto Boano, and Kay Römer. Towards secure and scalable uwb-based positioning systems. In *IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 247–255. IEEE, 2020.

[111] s.r.o. Sewio Networks. Uwb real-time location system. `https://www.sewio.net/real-time-location-system-rtls-on-uwb`. [Online] - Last access: 2021-03-20.

[112] Infsoft GmbH. Tracking of floor conveyors and goods in logistics. `https://tinyurl.com/35dwrhjv`. [Online] - Last access: 2021-03-20.

[113] Shaohua Huang, Yu Guo, Shanshan Zha, Falin Wang, and Weiguang Fang. A real-time location system based on rfid and uwb for digital manufacturing workshop. *Procedia Cirp*, 63:132–137, 2017.

[114] Samet Güler, Mohamed Abdelkader, and Jeff S Shamma. Peer-to-peer relative localization of aerial robots with ultrawideband sensors. *IEEE Transactions on Control Systems Technology*, 29 (5):1981–1996, 2020.

[115] Alwin Poulose and Dong Seog Han. Uwb indoor localization using deep learning lstm networks. *Applied Sciences*, 10(18):6290, 2020.

[116] Tianyu Wang, Keke Hu, Zhihang Li, Kangbo Lin, Jian Wang, and Yuan Shen. A semi-supervised learning approach for uwb ranging error mitigation. *IEEE Wireless Communications Letters*, 10 (3):688–691, 2020.

[117] Matteo Ridolfi, Jaron Fontaine, Ben Van Herbruggen, Wout Joseph, Jeroen Hoebeke, and Eli De Poorter. Uwb anchor nodes self-calibration in nlos conditions: A machine learning and adaptive phy error correction approach. *Wireless Networks*, 27(4):3007–3023, 2021.

[118] Jaron Fontaine, Matteo Ridolfi, Ben Van Herbruggen, Adnan Shahid, and Eli De Poorter. Edge inference for uwb ranging error correction using autoencoders. *IEEE access*, 8:139143–139155, 2020.

[119] Hongchao Wang, Xuexuan Wang, Yuan Xue, and Yemeng Jiang. Uwb-based indoor localization using a hybrid wknn-lstm algorithm. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 1720–1725. IEEE, 2020.

[120] Sedat Dogru and Lino Marques. Drone detection using sparse lidar measurements. *IEEE Robotics and Automation Letters*, 7(2):3062–3069, 2022.

[121] Jan Razlaw, Jan Quenzel, and Sven Behnke. Detection and tracking of small objects in sparse 3d laser range data. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2967–2973. IEEE, 2019.

[122] Marcus Hammer, Marcus Hebel, Björn Borgmann, Martin Laurenzis, and Michael Arens. Potential of lidar sensors for the detection of uavs. In *Laser Radar Technology and Applications XXIII*, volume 10636, pages 39–45. SPIE, 2018.

[123] Marcus Hammer, Marcus Hebel, Martin Laurenzis, and Michael Arens. Lidar-based detection and tracking of small uavs. In *Emerging Imaging and Sensing Technologies for Security and Defence III; and Unmanned Sensors, Systems, and Countermeasures*, volume 10799, pages 177–185. SPIE, 2018.

[124] Ismail Guvenc, Farshad Koohifar, Simran Singh, Mihail L Sichitiu, and David Matolak. Detection, tracking, and interdiction for amateur drones. *IEEE Communications Magazine*, 56(4): 75–81, 2018.

[125] S.Hengy *et al.* Multimodal uav detection: study of various intrusion scenarios. In *Electro-Optical Remote Sensing XI*, volume 10434, page 104340P. International Society for Optics and Photonics, 2017.

[126] Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanskỳ, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Voječh Spurnỳ, et al. Darpa subterranean challenge: Multi-robotic exploration of underground environments. In *International Conference on Modelling and Simulation for Autonomous Systesm*, pages 274–290. Springer, 2019.

[127] Matěj Petrlík, Tomáš Báča, Daniel Heřt, Matouš Vrba, Tomáš Krajník, and Martin Saska. A robust uav system for operations in a constrained environment. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):2169–2176, 2020.

[128] Li Qingqing, Jorge Pena Queralta, Tuan Nguyen Gia, and Tomi Westerlund. Offloading monocular visual odometry with edge computing: Optimizing image quality in multi-robot systems. In *Proceedings of the 2019 5th International Conference on Systems, Control and Communications*, pages 22–26, 2019.

[129] Weiwei Kong, Daibing Zhang, Xun Wang, Zhiwen Xian, and Jianwei Zhang. Autonomous landing of an uav with a ground-based actuated infrared stereo vision system. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2963–2970, 2013. doi: 10.1109/IROS.2013.6696776.

[130] Yang Gui, Pengyu Guo, Hongliang Zhang, Zhihui Lei, Xiang Zhou, Jing Du, and Qifeng Yu. Airborne vision-based navigation method for uav accuracy landing using infrared lamps. *Journal of Intelligent & Robotic Systems*, 72, 11 2013. doi: 10.1007/s10846-013-9819-5.

[131] Angus Pacala. Lidar as a camera – digital lidar's implications for computer vision. *Ouster Blog*, 2018.

[132] Tao Wu, Hao Fu, Bokai Liu, Hanzhang Xue, Ruike Ren, and Zhiming Tu. Detailed analysis on generating the range image for lidar point cloud processing. *Electronics*, 10(11):1224, 2021.

[133] Point Cloud Library. How to create a range image from a point cloud. URL `https://pcl.readthedocs.io/projects/tutorials/en/latest/range_image_creation.html`. Accessed on September 13, 2023.

[134] Ardi Tampuu, Romet Aidla, Jan Aare van Gent, and Tambet Matiisen. Lidar-as-camera for end-to-end driving. *Sensors*, 23(5), 2023. ISSN 1424-8220. doi: 10.3390/s23052845. URL `https://www.mdpi.com/1424-8220/23/5/2845`.

[135] Ha Sier, Xianjia Yu, Iacopo Catalano, Jorge Pena Queralta, Zhuo Zou, and Tomi Westerlund. Uav tracking with lidar as a camera sensor in gnss-denied environments. In *2023 International Conference on Localization and GNSS (ICL-GNSS)*, pages 1–7. IEEE, 2023.

[136] Xianjia Yu, Sahar Salimpour, Jorge Peña Queralta, and Tomi Westerlund. General-purpose deep learning detection and segmentation models for images from a lidar-based camera sensor. *Sensors*, 23(6):2936, 2023.

[137] Xieyuanli Chen, Ignacio Vizzo, Thomas Läbe, Jens Behley, and Cyrill Stachniss. Range image-based lidar localization for autonomous vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5802–5808, 2021. doi: 10.1109/ICRA48506.2021.9561335.

[138] Ben Kehoe, Sachin Patil, Pieter Abbeel, and Ken Goldberg. A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, 12(2): 398–409, 2015.

[139] Robert Bogue. Cloud robotics: a review of technologies, developments and applications. *Industrial Robot: An International Journal*, 2017.

[140] Cynthia Matuszek. Grounded language learning: Where robotics and nlp meet (invited talk). In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2018.

[141] Javier Ruiz-del Solar, Patricio Loncomilla, and Naiomi Soto. A survey on deep learning methods for robot vision. *arXiv preprint arXiv:1803.10862*, 2018.

[142] Haoran Li, Qichao Zhang, and Dongbin Zhao. Deep reinforcement learning-based automatic exploration for navigation in unknown environment. *IEEE transactions on neural networks and learning systems*, 31(6):2064–2076, 2019.

[143] Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:1912.10944*, 2019.

[144] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[145] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[146] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[147] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. Tifl: A tier-based federated learning system. In *Proceedings of the 29th international symposium on high-performance parallel and distributed computing*, pages 125–136, 2020.

[148] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[149] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 794–797. IEEE, 2020.

[150] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020.

[151] International Federation of Robotics. Ifr forecast: 1.7 million new robots to transform the world´s factories by 2020. `https://ifr.org/news/ifr-forecast-1.7-million-new-robots-to-transform-the-worlds-factories-by-20/`. [Online] - Last access: 2021-04-11.

[152] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[153] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[154] Farzad Niroui, Kaicheng Zhang, Zendai Kashino, and Goldie Nejat. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617, 2019.

[155] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

[156] Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5129–5136. IEEE, 2018.

[157] Zhenheng Tang, Shaohuai Shi, Xiaowen Chu, Wei Wang, and Bo Li. Communication-efficient distributed deep learning: A comprehensive survey. *arXiv preprint arXiv:2003.06307*, 2020.

[158] Rajesh Kumar, Abdullah Aman Khan, Sinmin Zhang, WenYong Wang, Yousif Abuidris, Waqas Amin, and Jay Kumar. Blockchain-federated-learning and deep learning models for covid-19 detection using ct imaging. *arXiv preprint arXiv:2007.06537*, 2020.

[159] Anwaar Ulhaq and Oliver Burmeister. Covid-19 imaging data privacy by federated learning design: A theoretical framework. *arXiv preprint arXiv:2010.06177*, 2020.

[160] Boyi Liu, Bingjie Yan, Yize Zhou, Yifan Yang, and Yixian Zhang. Experiments of federated learning for covid-19 chest x-ray images. *arXiv preprint arXiv:2007.05592*, 2020.

[161] Wuhui Chen, Yuichi Yaguchi, Keitaro Naruse, Yutaka Watanobe, Keita Nakamura, and Jun Ogawa. A study of robotic cooperation in cloud robotics: Architecture and challenges. *IEEE Access*, 6:36662–36682, 2018.

[162] Hehua Yan, Qingsong Hua, Yingying Wang, Wenguo Wei, and Muhammad Imran. Cloud robotics in smart manufacturing environments: Challenges and countermeasures. *Computers & Electrical Engineering*, 63:56–65, 2017.

[163] AWS. Aws robomaker. `https://aws.amazon.com/robomaker/`. [Online] - Last access: 2021-04-09.

[164] AUTOLAB: UC Berkeley Automation Lab. Dex-net. `https://berkeleyautomation.github.io/dex-net/`. [Online] - Last access: 2021-04-09.

[165] Google. Google cloud robotics. `https://googlecloudrobotics.github.io/core/`. [Online] - Last access: 2021-04-09.

[166] Eric P Xing, Qirong Ho, Wei Dai, Jin Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data*, 1(2):49–67, 2015.

[167] Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pages 344–353. PMLR, 2019.

[168] Huaming Wu, Ziru Zhang, Chang Guan, Katinka Wolter, and Minxian Xu. Collaborate edge and cloud computing with distributed deep learning for smart city internet of things. *IEEE Internet of Things Journal*, 7(9):8099–8110, 2020.

[169] Haotian Jiang, James Starkman, Yu-Ju Lee, Huan Chen, Xiaoye Qian, and Ming-Chun Huang. Distributed deep learning optimized system over the cloud and smart phone devices. *IEEE Transactions on Mobile Computing*, 20(1):147–161, 2019.

[170] Yitao Chen, Kaiqi Zhao, Baoxin Li, and Ming Zhao. Exploring the use of synthetic gradients for distributed deep learning across cloud and edge resources. In *2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[171] Shaohuai Shi, Xianhao Zhou, Shutao Song, Xingyao Wang, Zilin Zhu, Xue Huang, Xinan Jiang, Feihu Zhou, Zhenyu Guo, Liqiang Xie, et al. Towards scalable distributed training of deep learning on public cloud clusters. *Proceedings of Machine Learning and Systems*, 3, 2021.

[172] Yiran Li, Hongwei Li, Guowen Xu, Tao Xiang, Xiaoming Huang, and Rongxing Lu. Toward secure and privacy-preserving distributed deep learning in fog-cloud computing. *IEEE Internet of Things Journal*, 7(12):11460–11472, 2020.

[173] Davit Buniatyan. Hyper: Distributed cloud processing for large-scale deep learning tasks. In *2019 Computer Science and Information Technologies (CSIT)*, pages 27–32. IEEE, 2019.

[174] Jia Duan, Jiantao Zhou, and Yuanman Li. Privacy-preserving distributed deep learning based on secret sharing. *Information Sciences*, 527:108–127, 2020.

[175] Yitian Zhang, Hojjat Salehinejad, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Privacy preserving deep learning with distributed encoders. In *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1–5. IEEE, 2019.

[176] Praneeth Vepakomma, Otkrist Gupta, Abhimanyu Dubey, and Ramesh Raskar. Reducing leakage in distributed deep learning for sensitive health data. *arXiv preprint arXiv:1812.00564*, 2019.

[177] Jianguo Chen, Kenli Li, Qingying Deng, Keqin Li, and S Yu Philip. Distributed deep learning model for intelligent video surveillance systems with edge computing. *IEEE Transactions on Industrial Informatics*, 2019.

[178] Niranjan Balachandar, Ken Chang, Jayashree Kalpathy-Cramer, and Daniel L Rubin. Accounting for data variability in multi-institutional distributed deep learning for medical imaging. *Journal of the American Medical Informatics Association*, 27(5):700–708, 2020.

[179] Ho Bae, Jaehee Jang, Dahuin Jung, Hyemi Jang, Heonseok Ha, and Sungroh Yoon. Security and privacy issues in deep learning. *arXiv preprint arXiv:1807.11655*, 2018.

[180] Georgios A Kaissis, Marcus R Makowski, Daniel Rückert, and Rickmer F Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6):305–311, 2020.

[181] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1605–1622, 2020.

[182] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. Learning to detect malicious clients for robust federated learning. *arXiv preprint arXiv:2002.00211*, 2020.

[183] Latif U Khan, Walid Saad, Zhu Han, and Choong Seon Hong. Dispersed federated learning: Vision, taxonomy, and future directions. *arXiv preprint arXiv:2008.05189*, 2020.

[184] Yunlong Lu, Xiaohong Huang, Ke Zhang, Sabita Maharjan, and Yan Zhang. Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles. *IEEE Transactions on Vehicular Technology*, 69(4):4298–4311, 2020.

[185] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[186] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pages 493–506, 2020.

[187] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[188] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[189] Peter Biber and Wolfgang 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)*, volume 3, pages 2743–2748. IEEE, 2003.

[190] Jinyong Jeong, Younggun Cho, Young-Sik Shin, Hyunchul Roh, and Ayoung Kim. Complex urban lidar data set. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6344–6351. IEEE, 2018.

[191] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[192] Marco Lixia, Andrea Benigni, Alessandra Flammini, Carlo Muscas, Ferdinanda Ponci, and Antonello Monti. A software-only ptp synchronization for power system state estimation with pmus. *IEEE Transactions on Instrumentation and Measurement*, 61(5):1476–1485, 2012.

[193] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 573–580. IEEE, 2012.

[194] Jorge Pena Queralta, Qingqing Li, Fabrizio Schiano, and Tomi Westerlund. Vio-uwb-based collaborative localization and dense scene reconstruction within heterogeneous multi-robot systems. In *2022 International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 87–94. IEEE, 2022.

[195] Yang Bai, Koki Asami, Mikhail Svinin, and Evgeni Magid. Cooperative multi-robot control for monitoring an expanding flood area. In *2020 17th International Conference on Ubiquitous Robots (UR)*, pages 500–505. IEEE, 2020.

[196] Junwon Seo, Luis Duque, and Jim Wacker. Drone-enabled bridge inspection methodology and application. *Automation in construction*, 94:112–126, 2018.

[197] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[198] Iacopo Catalano, Ha Sier, Xianjia Yu, Tomi Westerlund, and Jorge Peña Queralta. Uav tracking with solid-state lidars: Dynamic multi-frequency scan integration. *arXiv preprint arXiv:2304.12125*, 2023.

[199] Hazim Shakhatreh, Ahmad H Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *IEEE Access*, 7: 48572–48634, 2019.

[200] Tuan Li, Hongping Zhang, Zhouzheng Gao, Qijin Chen, and Xiaoji Niu. High-accuracy positioning in urban environments using single-frequency multi-gnss rtk/mems-imu integration. *Remote sensing*, 10(2):205, 2018.

[201] Jae-One Lee and Sang-Min Sung. Assessment of positioning accuracy of uav photogrammetry based on rtk-gps. *Journal of the Korea Academia-Industrial cooperation Society*, 19(4):63–68, 2018.

[202] Kiyoung Kim, Jaemook Choi, Junyeon Chung, Gunhee Koo, In-Hwan Bae, and Hoon Sohn. Structural displacement estimation through multi-rate fusion of accelerometer and rtk-gps displacement and velocity measurements. *Measurement*, 130:223–235, 2018.

[203] Jorge Peña Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju, and Tomi Westerlund. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *IEEE Access*, 8:191617–191643, 2020.

[204] Cheng Hui, Chen Yousheng, Li Xiaokun, and Wong Wing Shing. Autonomous takeoff, tracking and landing of a uav on a moving ugv using onboard monocular vision. In *Proceedings of the 32nd Chinese control conference*, pages 5895–5901. IEEE, 2013.

[205] Pileun Kim, Leon C Price, Jisoo Park, and Yong K Cho. Uav-ugv cooperative 3d environmental mapping. In *Computing in Civil Engineering 2019: Data, Sensing, and Analytics*, pages 384–392. American Society of Civil Engineers Reston, VA, 2019.

[206] Xianjia Yu, Qingqing Li, Jorge Peña Queralta, and Tomi Westerlund. Applications of uwb networks and positioning to autonomous robots and industrial systems. *arXiv preprint arXiv:2103.13488*, 2021.

[207] Ming He, Chaozheng Zhu, Qian Huang, Baosen Ren, and Jintao Liu. A review of monocular visual odometry. *The Visual Computer*, 36(5):1053–1065, 2020.

[208] Andrei Cramariuc, Lukas Bernreiter, Florian Tschopp, Marius Fehr, Victor Reijgwart, Juan Nieto, Roland Siegwart, and Cesar Cadena. maplab 2.0–a modular and multi-modal mapping framework. *IEEE Robotics and Automation Letters*, 8(2):520–527, 2022.

[209] Marco Tranzatto, Takahiro Miki, Mihir Dharmadhikari, Lukas Bernreiter, Mihir Kulkarni, Frank Mascarich, Olov Andersson, Shehryar Khattak, Marco Hutter, Roland Siegwart, et al. Cerberus in the darpa subterranean challenge. *Science Robotics*, 7(66):eabp9742, 2022.

[210] Yanjun Cao and Giovanni Beltrame. Vir-slam: Visual, inertial, and ranging slam for single and multi-robot systems. *Autonomous Robots*, 45:905–917, 2021.

[211] Elizabeth R Boroson, Robert Hewitt, Nora Ayanian, and Jean-Pierre de la Croix. Inter-robot range measurements in pose graph optimization. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4806–4813. IEEE, 2020.

[212] Wenda Zhao, Jacopo Panerati, and Angela P Schoellig. Learning-based bias correction for time difference of arrival ultra-wideband localization of resource-constrained mobile robots. *IEEE Robotics and Automation Letters*, 6(2):3639–3646, 2021.

[213] Fisher Shi. Object detection and tracking using deep learning and ouster python sdk. `https://ouster.com/insights/blog/object-detection-and-tracking-using-deep-learning-and-ouster-python-sdk`. [Online] - Last access: 2024-06-17.

[214] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.

[215] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[216] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[217] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[218] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[219] Glenn Jocher, K Nishimura, T Mineeva, and R Vilariño. yolov5. *Code repository https://github.com/ultralytics/yolov5*, 2020.

[220] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018.

[221] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. A survey on instance segmentation: state of the art. *International journal of multimedia information retrieval*, 9(3):171–189, 2020.

[222] Yuhui Yuan, Xiaokang Chen, Xilin Chen, and Jingdong Wang. Segmentation transformer: Object-contextual representations for semantic segmentation. *arXiv preprint arXiv:1909.11065*, 2019.

[223] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9799–9808, 2020.

[224] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[225] Kenji Koide, Masashi Yokozuka, Shuji Oishi, and Atsuhiko Banno. Voxelized gicp for fast and accurate 3d point cloud registration. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11054–11059. IEEE, 2021.

[226] Pierre Dellenbach, Jean-Emmanuel Deschaud, Bastien Jacquet, and François Goulette. Ct-icp: Real-time elastic lidar odometry with loop closure. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5580–5586. IEEE, 2022.

[227] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. Kiss-icp: In defense of point-to-point icp–simple, accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters*, 8(2):1029–1036, 2023.

[228] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[229] Hans Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, Carnegie Mellon University, Pittsburgh, PA, September 1980.

[230] Nilanjan Dey, Pradipti Nandi, Nilanjana Barman, Debolina Das, and Subhabrata Chakraborty. A comparative study between moravec and harris corner detection of noisy images using adaptive wavelet thresholding technique. *arXiv preprint arXiv:1209.1558*, 2012.

[231] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. doi: 10.1109/CVPR.1994.323794.

[232] Edward Rosten and Tom 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*, pages 430–443. Springer, 2006.

[233] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 778–792, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15561-1.

[234] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517, 2012. doi: 10.1109/CVPR.2012.6247715.

[235] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[236] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410.

[237] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33833-8.

[238] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, 2011. doi: 10.1109/ICCV.2011.6126542.

[239] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[240] Pablo Fernández Alcantarilla, Jesús Nuevo, and Adrien Bartoli. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *British Machine Vision Conference*, 2013.

[241] Xin Yang and K.-T. Tim Cheng. Local difference binary for ultrafast and distinctive feature description. *IEEE transactions on pattern analysis and machine intelligence*, 36:188–194, 01 2014. doi: 10.1109/TPAMI.2013.150.

[242] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[243] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. Comparative evaluation of binary features. In *European Conference on Computer Vision*, 2012. URL `https://api.semanticscholar.org/CorpusID:17555345`.

[244] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65:43–72, 11 2005. doi: 10.1007/s11263-005-3848-x.

[245] Shaharyar Ahmed Khan Tareen and Zahra Saleem. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–10, 2018. doi: 10.1109/ICOMET.2018.8346440.

[246] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[247] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, 2017.

[248] Dibyendu Mukherjee, Q. M. Jonathan Wu, and Guanghui Wang. A comparative experimental study of image feature detectors and descriptors. *Machine Vision and Applications*, 26:443–466, 05 2015. doi: 10.1007/s00138-015-0679-9.

[249] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. doi: 10.1109/CVPR.2008.4587706.

[250] David Bojanić, Kristijan Bartol, Tomislav Pribanic, Tomislav Petković, Yago Diez, and Joaquim Mas. On the comparison of classic and deep keypoint detector and descriptor methods. pages 64–69, 09 2019. doi: 10.1109/ISPA.2019.8868792.

[251] Christoph Sager, Christian Janiesch, and Patrick Zschech. A survey of image labelling for computer vision applications. *Journal of Business Analytics*, 4(2):91–110, 2021. doi: 10.1080/2573234X.2021.1908861. URL `https://doi.org/10.1080/2573234X.2021.1908861`.

[252] Christopher J. Rapson, Boon-Chong Seet, M. Asif Naeem, Jeong Eun Lee, Mahmoud Al-Sarayreh, and Reinhard Klette. Reducing the pain: A novel tool for efficient ground-truth labelling in images. In *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–9, 2018. doi: 10.1109/IVCNZ.2018.8634750.

[253] Li Qingqing, Yu Xianjia, Jorge Peña Queralta, and Tomi Westerlund. Multi-modal lidar dataset for benchmarking general-purpose localization and mapping algorithms. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3837–3844, 2022. doi: 10.1109/IROS47612.2022.9981078.

[254] F. Neuhaus *et al.* . Mc2slam: Real-time inertial lidar odometry using two-scan motion compensation. In *German Conference on Pattern Recognition*, pages 60–72. Springer, 2018.

[255] Dimosthenis C Tsouros, Stamatia Bibi, and Panagiotis G Sarigiannidis. A review on uav-based applications for precision agriculture. *Information*, 10(11):349, 2019.

[256] Dongliang Wang, Quanqin Shao, and Huanyin Yue. Surveying wild animals from satellites, manned aircraft and unmanned aerial systems (uass): A review. *Remote Sensing*, 11(11):1308, 2019.

[257] Abel Gawel, Yukai Lin, Théodore Koutros, Roland Siegwart, and Cesar Cadena. Aerial-ground collaborative sensing: Third-person view for teleoperation. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–7. IEEE, 2018.

[258] Francesco Nex and Fabio Remondino. Uav for 3d mapping applications: a review. *Applied geomatics*, 6:1–15, 2014.

[259] M. Nieuwenhuisen *et al.* . Autonomous navigation for micro aerial vehicles in complex gnss-denied environments. *Journal of Intelligent & Robotic Systems*, 84(1):199–216, 2016.

[260] Nan Jiang, Kuiran Wang, Xiaoke Peng, Xuehui Yu, Qiang Wang, Junliang Xing, Guorong Li, Jian Zhao, Guodong Guo, and Zhenjun Han. Anti-uav: A large multi-modal benchmark for uav tracking. *arXiv preprint arXiv:2101.08466*, 2021.

[261] L. Qingqing *et al.* Offloading Monocular Visual Odometry with Edge Computing: Optimizing Image Compression Ratios in Multi-Robot Systems. In *The 5th ICSCC*. IEEE, 2019.

[262] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4. IEEE, 2011.

[263] Jorge Peña Queralta, Carmen Martínez Almansa, Fabrizio Schiano, Dario Floreano, and Tomi Westerlund. Uwb-based system for uav localization in gnss-denied environments: Characterization and dataset. *arXiv preprint arXiv:2003.04380*, 2020.

[264] Iacopo Catalano, Jorge Peña Queralta, and Tomi Westerlund. Evaluating the performance of multi-scan integration for UAV lidar-based tracking. *arXiv preprint*, 2023.

[265] Samuel S Blackman and Robert Popoli. *Design and analysis of modern tracking systems*. Artech House Publishers, 1999.

[266] Oscar Javier Montañez, Marco Javier Suarez, and Eduardo Avendano Fernandez. Application of data sensor fusion using extended kalman filter algorithm for identification and tracking of moving targets from lidar–radar data. *Remote Sensing*, 15(13):3396, 2023.

[267] Ge Guo and Shijie Zhao. 3d multi-object tracking with adaptive cubature kalman filter for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 8(1):512–519, 2022.

[268] Wangyan Li and Fuwen Yang. Information fusion over network dynamics with unknown correlations: An overview. *International Journal of Network Dynamics and Intelligence*, pages 1–14, 2023.

[269] Benjamin Noack, Joris Sijs, Marc Reinhardt, and Uwe D Hanebeck. Decentralized data fusion with inverse covariance intersection. *Automatica*, 79:35–41, 2017.

[270] Benjamin Noack, Joris Sijs, and Uwe D Hanebeck. Inverse covariance intersection: New insights and properties. In *2017 20th International Conference on Information Fusion (Fusion)*, pages 1–8. IEEE, 2017.

[271] Ertug Olcay, Fabian Schuhmann, and Boris Lohmann. Collective navigation of a multi-robot system in an unknown environment. *Robotics and Autonomous Systems*, 132:103604, 2020.

[272] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[273] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[274] Nvidia. Nvidia isaac sim. `https://developer.nvidia.com/isaac-sim`. [Online] - Last access: 2024-06-06.