Enabling Multi-LiDAR Sensing in GNSS-Denied Environments: SLAM Dataset, Benchmark, and UAV Tracking with LiDAR-as-a-camera

Master of Science Thesis
University of Turku
Department of Computing
Turku Intelligent Embedded and Robotic Systems (TIERS) Lab
2023
Ha Sier

Supervisors:
MSc. Xianjia Yu
Prof. Tomi Westerlund

UNIVERSITY OF TURKU
Department of Computing

HA SIER: Enabling Multi-LiDAR Sensing in GNSS-Denied Environments: SLAM Dataset, Benchmark, and UAV Tracking with LiDAR-as-a-camera

Master of Science Thesis, 81 p.
Turku Intelligent Embedded and Robotic Systems (TIERS) Lab
August 2023

The rise of Light Detection and Ranging (LiDAR) sensors has profoundly impacted industries ranging from automotive to urban planning. As these sensors become increasingly affordable and compact, their applications are diversifying, driving precision, and innovation. This thesis delves into LiDAR's advancements in autonomous robotic systems, with a focus on its role in simultaneous localization and mapping (SLAM) methodologies and LiDAR as a camera-based tracking for Unmanned Aerial Vehicles (UAV).

Our contributions span two primary domains: the Multi-Modal LiDAR SLAM Benchmark, and the LiDAR-as-a-camera UAV Tracking. In the former, we have expanded our previous multi-modal LiDAR dataset by adding more data sequences from various scenarios. In contrast to the previous dataset, we employ different ground truth-generating approaches. We propose a new multi-modal multi-lidar SLAM-assisted and ICP-based sensor fusion method for generating ground truth maps. Additionally, we also supplement our data with new open road sequences with GNSS-RTK. This enriched dataset, supported by high-resolution LiDAR, provides detailed insights through an evaluation of ten configurations, pairing diverse LiDAR sensors with state-of-the-art SLAM algorithms. In the latter contribution, we leverage a custom YOLOv5 model trained on panoramic low-resolution images from LiDAR reflectivity (LiDAR-as-a-camera) to detect UAVs, demonstrating the superiority of this approach over point cloud or image-only methods. Additionally, we evaluated the real-time performance of our approach on the Nvidia Jetson Nano, a popular mobile computing platform.

Overall, our research underscores the transformative potential of integrating advanced LiDAR sensors with autonomous robotics. By bridging the gaps between different technological approaches, we pave the way for more versatile and efficient applications in the future.

Keywords: LiDAR, multi-modal LiDAR, LiDAR SLAM benchmark, SLAM, Dataset, LiDAR-as-a-camera, UAV Tracking

Contents

List of Figures

List of Tables

List Of Acronyms

APE	Absolute Pose Errors
API	Application Programming Interfaces
CNN	Convolutional Neural Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DOF	Degrees of Freedom
DL	Deep Learning
FOV	Field of View
FOG	Fiber Optic Gyro
GICP	Generalized Iterative Closest Point
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
GNN	Graph Neural Network
HOG	Histogram of Oriented Gradients
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
iSAM	Incremental Smoothing and Mapping
INS	Inertial Navigation System
KF	Kalman Filter
LiDAR	Light Detection and Ranging
LOAM	LiDAR Odometry and Mapping

Mocap	Motion Capture
MOT	Multi-Object Track-ing
MEMS	Micro-electro-mechanical Systems
NDT	Natural Distribution Transform
PTP	Precise Timestamp Protocol
RTK	Real Time Kinematics
ROI	Region of Interest
ROS	Robot Operating System
RANSAC	Random Sample Consensus
RMSE	Root Mean Square Error
R-CNN	Region-based Convolutional Neural Network
SLAM	Simultaneous Localization and Mapping
SOT	Single-Object Tracking
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
SPAD	Single-photon Avalanche Diode
SDK	Software Development Kit
SSD	Single Shot Detector
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicles
UGV	Unmanned Ground Vehicles
VRS	Virtual Reference Station
VCSEL	Vertical-cavity Surface-emitting Laser
YOLO	You Only Look Once

1 Introduction

LiDAR sensors have been adopted as the core perception sensor in many applications, from self-driving cars [1] to unmanned aerial vehicles (UAV) [2], including forest surveying and industrial digital twins [3]. High-resolution spinning LiDARs enable a high degree of awareness of the surrounding environments. More dense 3D point clouds and maps are increasing demand to support the next wave of ubiquitous autonomous systems as well as more detailed digital twins across industries. However, higher angular resolution comes at an increased cost in analog LiDARs requiring a higher number of laser beams or a more compact electronics and optics solution. New solid-state and other digital LiDARs are paving the way to cheaper and more widespread 3D LiDAR sensors capable of dense environment mapping [4]–[7].

So-called solid-state LiDARs overcome some challenges of spinning LiDARs in terms of cost and resolution, but introduce some new limitations in terms of a relatively small field of view (FoV) [6], [8]. Indeed, these LiDARs provide more sensing range at significantly lower cost [9]. Other limitations that affect traditional approaches to LiDAR data processing include irregular scanning patterns or increased motion blur.

Ouster LiDARs are a notable example among the latest generation of LiDARs, owing to their ability to generate dense point clouds and 360° panoramic views. Additionally, they capture low-resolution images that encode information about depth, reflectivity, or near-infrared light in their image pixels. This new type of LiDAR is referred to as a “LiDAR-as-a-camera sensor". One of the unique features of this sensor is its compatibility

with deep learning (DL) models of vision sensors without requiring additional camera setup and calibration [10]. This has the potential to improve upon traditional lidar-based object detection and tracking methods, which tend to be more complex and less advanced than vision sensors.

LiDAR datasets have emerged as a critical tool for applications advancing autonomous systems and improving environmental mapping. By structuring and organizing data systematically, these datasets lay a robust foundation for endeavors such as algorithm development, calibration techniques, and performance benchmarking. Such a framework permits researchers and technicians to detect anomalies, refine data processing strategies, and test solutions in simulated or virtual environments before venturing into real-world implementations. However, a lacuna exists in the current scenario. Comprehensive LiDAR datasets encompassing these nascent LiDAR technologies are few and far between. Furthermore, applications tailored to leverage the characteristics of these emerging LiDAR systems are still in their infancy, and their developmental potential remains to be fully realized.

This thesis seeks to fill the existing void in the field. Drawing from our prior research [9], we provide an in-depth assessment of cutting-edge SLAM algorithms using a diverse multi-modal LiDAR setup. For the robotics community, we present a rich multi-LiDAR dataset paired with a thorough LiDAR SLAM benchmark analysis. Beyond this, we detail a unique approach to craft ground truth trajectories, specially tailored for scenarios where MOCAP or GNSS/RTK isn't an option. Expanding our scope, we also investigate the promising capabilities of the innovative digital LiDAR, Ouster, with a specific focus on its utility in UAV tracking.

1.1 Significance and Motivation

There are few works that have benchmarked the performance of both spinning LiDAR and solid-state LiDAR in diverse environments, which limits the development of

general-purpose LiDAR-based simultaneous localization and mapping (SLAM) algorithms [9]. To bridge the gap in the literature, we present a benchmark that compares different modalities of LiDARs (spinning, solid-state) in diverse environments, including indoor offices, long corridors, halls, forests, and open roads. To allow for a more accurate and fair comparison, we introduce a new method for ground truth generation in larger indoor spaces (see Fig. 1.1). 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 than GNSS/RTK solutions, which also enhanced ground truth enables a significantly higher degree of quantitative benchmarking and comparison with respect to our previous work [9]. We hope for the extended dataset and ground truth labels, as well as more detailed data, to provide a performance reference for multi-modal LiDAR sensors in both structured and unstructured environments to both academia and industry.

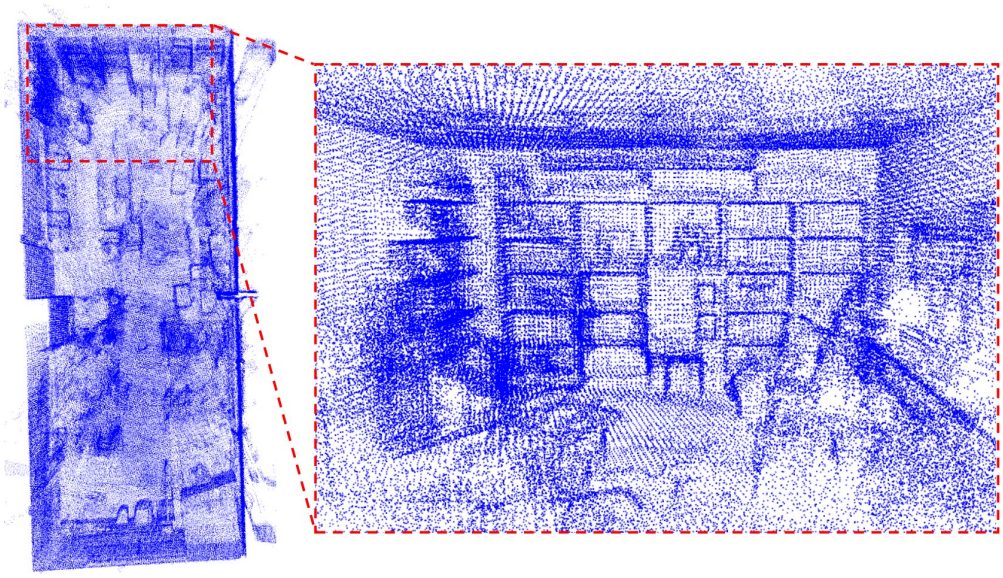


Figure 1.1: Ground truth map for one of the indoor sequences generated based on the proposed approach (SLAM-assisted ICP-based prior map).

Here, we also propose a drone tracking system based on the Ouster OS0-128 LiDAR, consisting of only a single Ouster LiDAR placed on the ground or nearby. The results

demonstrate that this design can provide precise drone flight trajectories and serve as an alternative to expensive equipment such as MoCAP systems and GNSS receivers. The core attribute of this project is that, with only a single Ouster LiDAR as the data input device, it combines the signal images of the LiDAR and point cloud data (see Fig. 1.2) to output accurate positioning results.

This project combines object detection in computer vision with point cloud data processing, rapidly locating the target of interest in the LiDAR's signal image, which greatly reduces the computation in point cloud processing. To the best of our knowledge, there is no existing work on drone tracking systems using this method. In our experiments, we use only one Ouster LiDAR and achieve an average reconstruction error of 3 cm when the UAV is 8.0 m from the LiDAR.

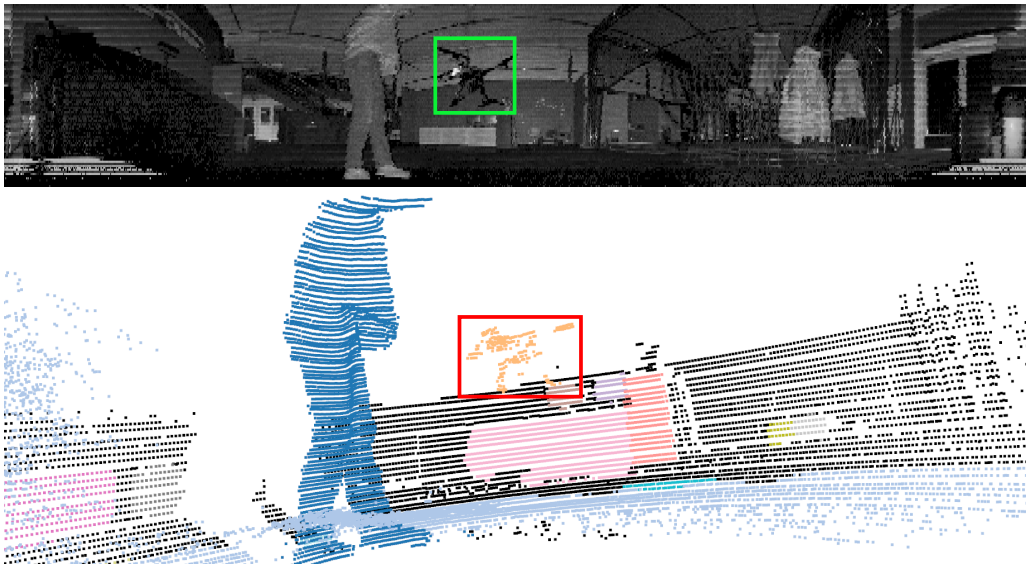


Figure 1.2: Exampe of signal image (top) and its corresponding point cloud(bottom, background removed).

1.2 Related Works

Previous research directly related to this project [11] includes the use of the Leica BLK360, an advanced precision imaging laser scanner, to generate a ground truth map.

The research then used handheld OS1-64 point cloud data and the ground truth map to perform iterative closest point (ICP) matching, resulting in a 6 degrees of freedom (DOF) ground truth trajectory with an error of less than 3cm. However, the sensors used in this project were limited to only an Ouster OS1-64 and a Realsense stereo camera. Another study [12] used a car as a data collection platform and equipped it with 2 Velodyne-16 LiDARs, 2 SICK 2D LiDARs, 2 Setero cameras, and IMU, GPS, and other sensors. The study used an Incremental Smoothing and Mapping (iSAM [13]) pose graph SLAM framework to estimate the baseline trajectory and obtained rotation measurements from Fiber Optic Gyro (FOG) and VRS-GPS to compute sequential relative constraints for the SLAM framework. The data obtained from the sensors and optimization process can be used for high-precision pose graph computation. However, study [14] lacked point cloud data from new solid-state LiDARs such as Livox Horizon. The MIT Stata Center dataset used a PR2 ground robot as a platform for data collection, which was equipped with 2 RGBD cameras, 2 laser scanners, and IMU and other sensors. The study used 2D building floor plans for each level of the building and LiDAR point cloud data to perform iSAM matching, resulting in a reliable ground truth trajectory. However, the LiDAR used in this study was a laser scanner rather than modern solid-state LiDAR or new rotational radar LiDARs such as Ouster OS0-128.

3D object tracking can be broadly categorized into two types: multi-object tracking (MOT) and single-object tracking (SOT). In [15], it was proposed to use pre-existing 3D object detectors to extract oriented 3D bounding boxes from LiDAR point clouds. State estimation and data association were then performed using a combination of the 3D Kalman filter and the Hungarian algorithm. Research [16] introduced a novel joint MOT method based on graph neural networks (GNN), which can model relationships between variable objects in both spatial and temporal domains. SOT methods aim to track a single object based on a given template. Study [17] developed a Siamese tracker that encodes the model and candidate shape into a compact latent representation. The

encoding is regularized by decoding the latent representation into the object model shape. Research [18] utilizes spatiotemporal data association to robustly achieve object tracking, comprising two main parts: estimating object time information and updating the region of interest (ROI) using a cross-frame temporal motion model, then updating the predicted state based on paired spatiotemporal data.

In the presented study [5], a sophisticated multimodal tracking technique was delineated. This technique employs high-frequency scans to achieve precise state determination, whereas low-frequency scans are utilized to ensure sustained and resilient tracking. Additionally, terahertz processing is integrated for the discernment of both trajectories and targets. Nonetheless, a notable prerequisite for this approach is the provision of the UAV's initial positioning.

1.3 Contribution

In summary, this work analyzed state-of-the-art SLAM algorithms with a multimodal multi-LiDAR platform as an extension of our previous work [9]. And we also propose a UAV tracking approach based on the integration of images and 3D point clouds generated by an Ouster LiDAR sensor. This provides an effective supplement for positioning and external tracking in a GNSS-denied environment. The main contributions of this work are as follows:

1. A ground truth trajectory generation method leveraging the multi-modality of the data acquisition platform and high-resolution sensors with an average absolute pose error (APE) of 0.048 m, for environments where MoCAP or GNSS/RTK are unavailable or unreliable;

2. A new dataset in various environments with data from 5 different LiDAR sensors, one LiDAR camera, and one stereo fisheye camera as illustrated in Fig. 1.3. Ground truth data is provided for all sequences;

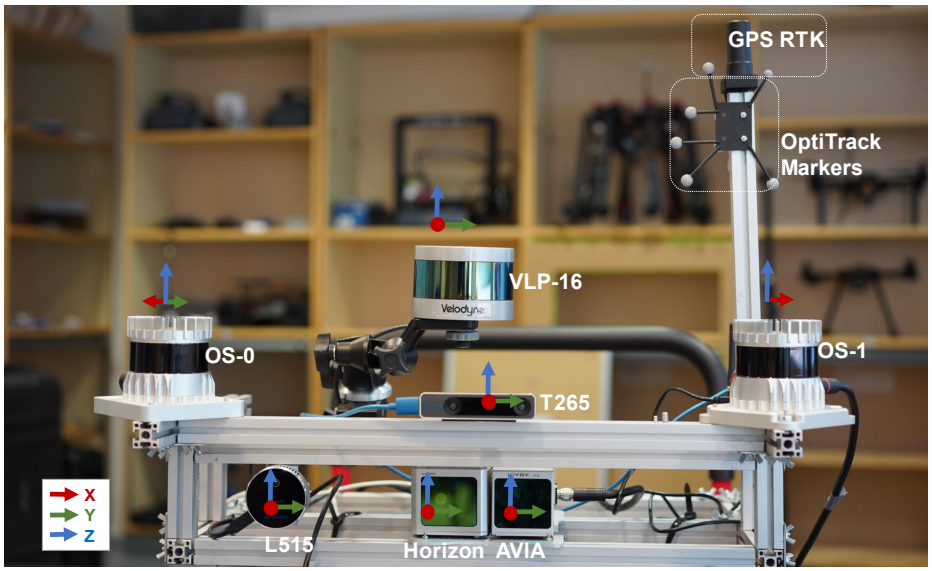


Figure 1.3: Multi-modal sensor data acquisition system with their coordinate frames (front view).

3. A benchmarking of ten 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.

4. A UAV tracking method for estimating trajectories of drones in the absence of Mo-CAP or GNSS/RTK that fuse signal image and point cloud data of Ouster LiDAR based on LiDAR-as-a-camera with an average absolute pose error (APE) of 0.06 m.

1.4 Structure

This paper is divided into 5 chapters as follows:

1. Chapter 2 briefly describes the open-source hardware and software systems that form the basis of the project's development.

2. Chapter 3 comprehensively introduces the Multi-model LiDAR SLAM benchmark from three levels of design overview, implementation, and experimental results.

3. Chapter 4 provides an in-depth exploration of the use of the new Ouster LiDAR in UAV tracking, presenting a comprehensive overview of the design, implementation, and experimental results of UAV tracking based on LiDAR-as-a-camera, covering all aspects of the technology.

4. Finally, the conclusions of the project and ideas for further research can be found in Chapter 5.

2 Background

In the present chapter, we conduct an exhaustive exploration into the core principles and theoretical constructs associated with Multi-Model Simultaneous Localization and Mapping (SLAM) Benchmarking and LiDAR-based Unmanned Aerial Vehicle (UAV) tracking. This entails an extensive exposition on 3D LiDAR, its integration with SLAM, and the methodology of SLAM Benchmarking. Simultaneously, this chapter will scrutinize auxiliary technologies including the Robot Operating System (ROS), point cloud processing techniques, trajectory estimation methodologies, and strategies for object detection.

2.1 LiDAR Sensor

LiDAR (Light Detection and Ranging) is a sensor technology that employs laser emission to scan an environment and subsequently constructs a three-dimensional representation of the surrounding landscape. The operational mechanism involves the emission of a laser beam, with subsequent measurement of the elapsed time required for the beam to reflect off an object and return to the sensor. Through calculating the distance to an array of points within the environment, a LiDAR system is capable of formulating a comprehensive 3D map of its vicinity. This technology finds extensive application in areas such as robotics, autonomous vehicles, and other disciplines that require a nuanced understanding of the environment, due to its real-time, high-resolution 3D modeling capabilities, proving particularly efficacious for tasks encompassing navigation, localization, and obstacle

avoidance. One of the principal merits of LiDAR technology is its capacity to generate high-resolution 3D environmental maps. These maps hold potential for a multitude of uses, including but not limited to, localization, path planning, and object detection and classification. Additionally, LiDAR sensors demonstrate commendable operational flexibility, exhibiting functionality in an array of settings, both indoor and outdoor. Compared to alternative sensing technologies, LiDAR exhibits superior robustness and reliability. In terms of operational distance, LiDAR sensors exhibit a broad range, from mere centimeters to several hundred meters. Furthermore, these sensors demonstrate versatility in mounting options, with compatibility extending to ground vehicles, aerial drones, and handheld devices.

LiDAR sensors comprise numerous variants, spanning from small, economically feasible sensors to larger, high-resolution systems. Broadly, these can be categorized into two main types: solid-state LiDAR and mechanically rotating LiDAR.

2.1.1 Solid-state LiDAR

Solid-state LiDAR represents a variant of LiDAR technology that utilizes a solid-state laser coupled with a detector to comprehend its environmental context. This technology primarily hinges upon the reflection or reception of waves to identify the properties of targets, drawing significantly from research conducted in the realm of 3D image sensors. The design of solid-state LiDAR, in fact, originates from infrared focusing plane imagers, incorporating a photodetector array positioned strategically on the focusing plane of the detector. Infrared rays emanating from an infinite distance traverse the optical system and are imaged upon these photodetectors on the system focusing plane. The detector then transposes the received light signal into an electrical equivalent, subsequently executing integrative amplification, sampling and holding. The electrical signal ultimately reaches the monitoring system through the output buffer and a multi-channel transmission system to materialize as an image. A salient advantage of solid-state LiDAR is its compact

dimensions and cost-effective production. The absence of moving components enables the design to be significantly smaller and more economical than its traditional LiDAR counterparts. As such, solid-state LiDAR is an optimal choice for applications where size and cost considerations hold paramount significance, such as in the case of autonomous vehicles and drones.

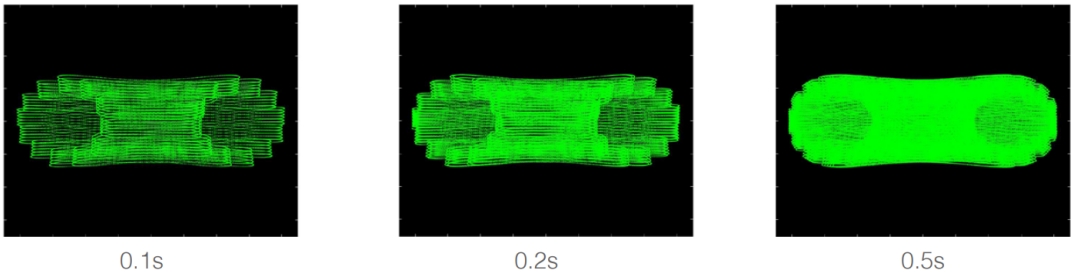


Figure 2.1: Point cloud patterns of the Livox Horizon LiDAR accumulated over an extended period [19] .

Within the burgeoning industry of solid-state LiDAR production, Livox emerges as one of the most prominent companies, distinguished by its substantial manufacturing scale and formidable research capabilities. Livox LiDAR systems are engineered to offer high-resolution, long-range three-dimensional sensing, facilitating a myriad of applications such as autonomous vehicles and robotics, among others.

A defining attribute of Livox LiDAR systems is their capacity to provide high-resolution three-dimensional sensing at extensive distances. For instance, certain models, including the Livox Mid-40, can scan up to a range of 200 meters. Conversely, other models, such as the Livox Horizon, extend their scanning reach up to 300 meters. This extended range renders Livox LiDAR systems highly suitable for applications demanding long-range sensing, such as autonomous vehicles or unmanned surveillance drones [20]. Moreover, the compact size and light weight of Livox LiDAR further solidifies its standing in the market.

The Livox Horizon specifically utilizes the company's unique non-repetitive scanning technology in conjunction with multi-laser and multi-Avalanche PhotoDiode (APD) DL-Pack technology, guaranteeing the generation of high-density point clouds. Fig. 2.1 il-

lustrates the distribution of point clouds within the Horizon's Field of View (FOV) within time intervals of 0.1s, 0.2s, and 0.5s. It is evident that as time progresses, the coverage within the FOV significantly augments.More detailed information about the surrounding environment revealed in [8].

2.1.2 Spinning LiDAR

Mechanical spinning LiDAR constitutes a variant of LiDAR technology that employs a mechanical system for environmental scanning, facilitating the creation of a three-dimensional map of the surroundings. Its operational mechanism involves emitting a laser beam and subsequently measuring the time taken for the beam to reflect off an object and return. The laser and detector are installed on a rotating platform, allowing the laser beam to encompass a broad FOV. Mechanical spinning LiDAR systems possess the capacity to generate high-resolution 3D environmental maps, which are invaluable for diverse purposes including localization, path planning, and object detection and classification. Mechanical spinning LiDARs also exhibit robustness and reliability, rendering them functional in a variety of environments, including indoor and outdoor settings. Nevertheless, mechanical rotation LiDARs have certain limitations as follows.

- Cost: Mechanical rotating LiDARs are typically expensive, especially for high-resolution systems, with current prices ranging approximately from 5000 to 10000 EUR.
- Size and weight: their size and weight can be considerable, making them challenging to install and integrate into smaller or lightweight systems.
- Complexity: due to its mechanical components, these systems exhibit increased complexity and a greater propensity for wear and failure than other types of LiDAR.

However, it is worth noting that the prices for smaller industrial LiDAR systems have significantly declined in recent years, and are projected to further decrease due to emerg-

ing technologies such as Micro-Electro-Mechanical Systems (MEMS) mirrors, optical phased arrays, Single-Photon Avalanche Diode (SPAD) detectors, and Vertical-Cavity Surface-Emitting Laser (VCSEL) sources, as suggested in research [21] [22].

A noteworthy example of advancements in LiDAR technology is San Francisco-based LiDAR startup, Ouster, which has developed "digital LiDAR" systems. This refers to a fully semiconductor-based design of LiDAR, where thousands of optoelectronic devices, such as emitters and receivers, traditionally incorporated into LiDAR products are integrated into chips to achieve a "solid-state" form. In addition, Ouster has incorporated a rotating device into the LiDAR, enabling 360-degree scanning, giving rise to "rotating solid-state LiDAR".

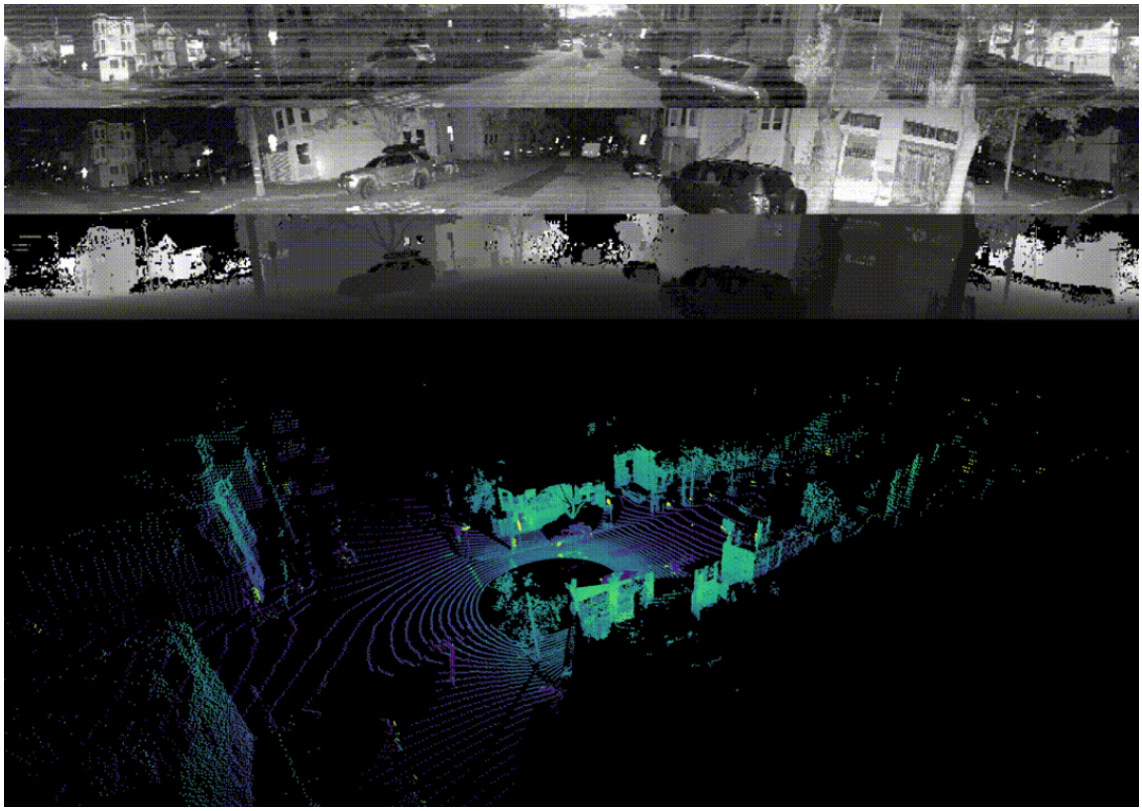


Figure 2.2: Simultaneous real-time image layers output from the OS0. From top to bottom are ambient, intensity, range, and point cloud data.

Digital LiDAR systems, such as Ouster's OS1 [23] series, are compact and lightweight, with resolutions ranging from 16 to 128 lines and weights less than 400 grams. Ouster

further supplements its offering with a powerful and efficient LiDAR driver [24] for customer service, which includes Application Programming Interfaces (APIs) for real-time access to LiDAR data and tools for data visualization and analysis. Fig. 2.2 displays the environmental, intensity, range, and point cloud data from the Ouster OS0, demonstrating the capabilities of these systems. The Ouster SDK also provides sample code and documentation to facilitate developers' integration of Ouster LiDARs into their systems.

2.2 3D LiDAR SLAM

Simultaneous Localization and Mapping (SLAM) serves as a crucial technology for autonomous systems including drones, robots, and self-driving cars. This technology enables these systems to navigate and concurrently map their surroundings in real-time, thus facilitating informed decision-making regarding environmental interaction and movement. 3D LiDAR (Light Detection and Ranging) sensors are frequently employed for SLAM due to their capacity to generate high-resolution 3D point clouds of the environment. By persistently measuring the distance to surrounding objects utilizing lasers, 3D LiDAR sensors are capable of crafting a meticulous map of the proximate environment. Numerous methodologies exist for implementing 3D LiDAR SLAM, among which feature-based methods (identifying distinct features in the point cloud data) and optimization-based methods (employing optimization algorithms to minimize the error between the current pose estimate and measured data) are the most common. A significant advantage of employing 3D LiDAR for SLAM is its ability to provide exceptionally accurate and dense 3D maps of the environment. This accuracy proves particularly beneficial for applications such as autonomous navigation, localization, and object recognition.

The primary 3D LiDAR SLAM algorithms currently in use include LiDAR-only [25] and those loosely-coupled [26] or tightly-coupled [27] with IMU data. Tightly-coupled approaches incorporate LiDAR and IMU data at an early stage, contrasted with SLAM methodologies that loosely fuse LiDAR and IMU outputs towards the end of their respec-

tive processing pipelines.

Regarding LiDAR-only methods, early work by Zhang et al. on LiDAR Odometry and Mapping (LOAM) introduced a method capable of achieving low-drift and low-computational complexity as early as 2014 [28]. Subsequent advancements have led to multiple variations of LOAM, enhancing its performance. LeGO-LOAM, for instance, incorporated ground point segmentation and a loop closure module, making it more lightweight while maintaining accuracy, and improving computational expense and lowering long-term drift [29]. Nevertheless, LiDAR-only approaches tend to be limited by a high susceptibility to featureless landscapes [30], [31]. By incorporating IMU data into the state estimation pipeline, SLAM systems can achieve greater precision and flexibility.

In LIOM [27], authors proposed a tightly-coupled approach with LiDAR-IMU fusion based on graph optimization which outperformed state-of-the-art LiDAR-only and loosely coupled systems. This better performance has caused subsequent research to concentrate on tightly-coupled approaches.Fast-LIO [32] is a practical tightly-coupled method that 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 [33] introduced a dynamic structure ikd-tree to the system, facilitating the incremental map update at every step and addressing computational scalability issues, while inheriting the tightly-coupled fusion framework from FAST-LIO.

While the vast majority of these algorithms perform effectively with spinning LiDARs, novel approaches are in demand, given the emergence of new sensor types such as solid-state Livox LiDARs. These have ushered in innovative sensing modalities, reduced FoV, and irregular samplings [9]. Current research efforts are focusing on enhancing existing SLAM algorithms to accommodate these new LiDAR characteristics. Loam livox [34] exemplifies these endeavors, providing a robust and real-time LOAM algorithm specifically designed for such LiDAR types. LiLi-OM [6], another tightly-coupled method, jointly minimizes the cost derived from both LiDAR and IMU measurements,

catering to both solid-state LiDARs and traditional LiDARs.

It merits attention that several other studies are targeting LiDAR odometry and mapping, not merely by integrating IMU data, but also incorporating visual information or other ranging data. This approach seeks to achieve more robust and accurate state estimation [35], [36].

2.3 SLAM Benchmarking Datasets

The evaluation of Simultaneous Localization and Mapping (SLAM) algorithm performance is a crucial task in the advancement of autonomous systems such as drones, robots, and self-driving cars. These systems are reliant on precise and efficient SLAM algorithms for real-time navigation and mapping of their environments. To compare the performance of various SLAM algorithms, researchers commonly employ benchmarks, consisting of standardized datasets and evaluation criteria. These benchmarks facilitate a fair and consistent comparison of different algorithms, enabling researchers to distinguish the most promising approaches and identify areas for potential improvement. In terms of the platforms hosting the sensors, extant datasets can be divided into two subcategories: extensive datasets with vehicles serving as platforms, commonly utilized in outdoor settings, and datasets employing ground robots or humans as platforms, typically used indoors, focusing on vision, IMU, and LiDAR modules. Table 2.1 presents fundamental information regarding the datasets discussed below. A systematic comparison of the popular datasets has been previously provided in Table 2.1 of our preceding work [9]. Among these datasets, it is noteworthy to mention that not all possess an analytical benchmark of 3D LiDAR SLAM based on multi-modality LiDARs.

Table 2.1: Comparison of related datasets with ours [9].

Dataset	Year	Environment	Ground Truth	LiDARs	Other
KITTI[37]	2013	Urban road	RTK_GPS/INS	3D-Velodyne HDL-64E @10 Hz	4× cameras , accel/gyro
NCLT[38]	2017	Urban Indoor Outdoor	GPS/INS	3D-Velodyne HDL-64E@10 Hz 2× 2D-Hokuyo @10/40 Hz	camera
Oxford RobotCar[39]	2017	Urban Road	GPS/INS	2× 2D-SICK @50 Hz 3D-SICK @12.5Hz	4 Camera; accel/gyro
RUGB Dataset[40]	2019	Unstructured outdoor	-	3D-Velodyne HDL-32E @10 Hz	GPS&IMU ; 3× cameras
nuScences[41]	2020	Urban Road	-	3D-32-Beams Lidar @20 Hz	6x Camera (RGB);GPS&IMU; 5x Radar@13Hz
Newer Colleage[11]	2020	Urban outdoor Vegetated	6DOF ICP	3D-Ouster-64 @10 Hz	D435i (Infrared); accel/gyro
DARPA[11]	2010	Structured Urban	GPS/INS	3D-Velodyne HDL-64E @15 Hz	Point Grey Firefly MV; accel/gyro
PandaSet[42]	2021	Urban road	-	3D-Hesai-Pandar64 @10 Hz 3D solid-state lidar@10 Hz	6x Cameras. GNSS&IMU
M2DGR [43]	2022	Urban In/Outdoors	Laser 3D tracker RTK_GPS/INS	3D VLP-32C @10 Hz	3 Cameras. GNSS&IMU
TIERS LiDAR Dataset	2022	Urban indoor Urban road Forest	6DOF MoCAP SLAM/ICP GNSS/RTK	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

2.3.1 Vehicle-based Dataset

The KITTI benchmark [37] merits significant attention as a dataset capable of assessing various tasks, including odometry, SLAM, object detection, tracking, among others. It provides 3D LiDAR point cloud data and ground-truth poses in authentic environments, thus allowing researchers to evaluate the precision and efficiency of disparate SLAM algorithms. KITTI utilizes RTK-GPS/INS to provide 6 degrees of freedom ground truth trajectories with an error of less than 10 cm across all traversals. However, the lack of synchronization between this dataset's IMU data and images could potentially impact the performance of numerous visual-inertial odometry methods. Furthermore, since GPS accuracy may be compromised in regions such as urban canyons, assured accuracy cannot be guaranteed in such areas. The MIT DARPA Urban Challenge dataset [38] is one of the main Unmanned Ground Vehicle (UGV) datasets, originally developed at MIT. It contains imagery, GPS, and point cloud data from Talos vehicles during a 90km traversal that spans seven hours of autonomous driving. The ground truth is furnished by the integration of high-precision GPS/INS.

The Oxford RobotCar dataset [39], currently the most extensive autonomous driv-

ing dataset, includes over 1000 km of driving in central Oxford across all conceivable weather conditions. This dataset encapsulates multiple features of complex urban areas, such as GPS loss, diverse pedestrian scenes on sidewalks, and dynamic entities like bicyclists. The nuScenes dataset [41], developed by the Motional team, is a large-scale dataset intended for autonomous driving that provides a comprehensive sensor suite from autonomous vehicles, including six cameras, one LiDAR, five radars, GPS, and IMU. The dataset comprises 1,000 driving scenarios in Boston and Singapore, two cities notorious for dense traffic and challenging driving environments. The selected scenes, each 20 seconds long, depict a diverse and intriguing assortment of driving maneuvers, traffic situations, and unexpected behaviors. The PandaSet dataset [42], a collaborative effort by Hesai and Scale AI, merges Hesai's leading-edge LiDAR sensor with Scale AI's high-quality data annotation. PandaSet includes data collected using a forward-looking LiDAR (PandarGT) and a mechanically rotating LiDAR (Pandar64) with similar image resolution. The gathered data is annotated using a combination of cuboid and segmentation annotation (Scale 3D Sensor Fusion Segmentation).

2.3.2 Mobile Robot or Human-carried Dataset

The newer University Vision and LiDAR Dataset [11] was motivated by the need for more inclusive mobile mapping sensors. This dataset encapsulates almost 6.7 kilometers at typical walking speeds around New College, Oxford University. Utilizing commercially available sensors, it includes challenging sequences characterized by rapid motion, severe shaking, abrupt lighting changes, and texture-less surfaces. The North Campus Long Term (NCLT) dataset [44] is another significant dataset discussed herein. Collected using a Segway ground robot, it covers the entirety of a university campus, both indoors and outdoors, spanning a distance of 147.4 kilometers. This dataset differentiates itself by employing LiDAR scan matching and high-precision RTK-GPS for ground truth.

The RUGD dataset [40], is designed with a focus on semantic comprehension of

unstructured outdoor environments for off-road autonomous navigation applications. The dataset comprises video sequences captured by cameras on mobile robotic platforms. The platforms employed for data collection are sufficiently compact to maneuver in cluttered environments and sufficiently robust to traverse challenging terrain, thereby exploring more unstructured aspects of the environment.

The M2DGR dataset [43], gathered by a ground robot outfitted with various sensors, including six fisheye and an RGB camera oriented skyward, an infrared camera, an event camera, a visual-inertial sensor (VI-Sensor), Inertial Measurement Unit (IMU), LiDAR, consumer-grade Global Navigation Satellite System (GNSS) receivers, and GNSS-IMU navigation systems with real-time kinematics (RTK) signals. This dataset encompasses 36 sequences captured in disparate environments, with ground-truth trajectories obtained using motion capture devices, among other methods.

2.4 UAV Detection and Tracking

This section reviews the literature in the field of UAV detection and tracking. Due to the limited research on tracking small objects such as UAVs based on LiDAR, we focus on: (i) UAV tracking with cameras; (ii) UAV tracking with LiDARs; and (iii) Applications of UAV tracking.

2.4.1 UAV tracking with cameras

Vision-based methods are widely used to track small objects and UAVs [45]–[47]. They can be divided into two categories: those that rely on passive or active visual markers, and those that detect and track objects in general, e.g., with traditional computer vision or deep learning. For example, [46] introduces a trinocular system with ground-based cameras to control a rotary-wing UAV in real time based on its key features. Alternatively, [48] presents an infrared binocular vision system with PTU and exosensors to track drones

cheaply under any weather and time conditions based on their infrared spectra. Recent developments in deep convolutional neural networks (CNNs) have boosted adoption in the field of object detection and tracking. Arguably, a large part of the state-of-the-art in tracking is based on deep learning methods. Recent advances in deep CNNs have improved object detection and tracking performance [49]. For instance, [47] proposes a CNN-based markerless UAV relative positioning system that allows the stable formation and autonomous interception of multiple UAVs.

Depth cameras can also detect UAVs and help them avoid obstacles using deep learning models that process depth maps [50]. While depth cameras can provide accurate position and size measurements, and vision sensors are generally capable of robust tracking and relative localization, our focus in this paper is on the use of Ouster LiDARs because of their flexibility with respect to environmental conditions and their ability to provides more accurate and informative signal images than depth cameras.

2.4.2 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. [5] 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 [51], 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 [52] 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 [53], [54].

2.4.3 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 [55], [56], and the potential for optimizing the utilization of UAVs as versatile mobile sensing platforms through tracking and detection [57].

The ability to track UAVs from unmanned ground vehicles (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 [58], [59], 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 visual-inertial odometry (VIO) has limitations, such as low-visibility situations [7], [60].

Similarly, tracking UAVs is crucial in the landing phase of the aerial system. Different methods using a ground-based stereo camera [61] or having the UAV carry an infrared camera to detect signals from the destination [62] 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.

2.5 Robot Operating System

The Robot Operating System (ROS) serves as a robust framework for creating robotics applications. It equips users with a comprehensive suite of tools and libraries for facilitating hardware abstraction, low-level device control, the implementation of frequently used functionalities, and inter-process message passing. A fundamental aspect of ROS is the notion of a node, which embodies a functional unit within a robotic system. Nodes communicate through a publish-subscribe model, where a node publishes data to a topic, and other nodes subscribe to the same topic to receive the data. This approach fosters a loosely coupled system, facilitating the addition, removal, or modification of functionalities without adversely affecting the entire system (refer to Fig. 2.3).

ROS also incorporates numerous libraries for typical robotic tasks, such as robot navigation, perception, and control. These libraries are realized as nodes that can be effortlessly integrated into a ROS system. Beyond the core ROS framework, a vast ecosystem of open-source packages extends further functionality. These packages offer capabilities including drivers for specific hardware platforms, support for robot simulations, and tools for robot integration and deployment.

ROS is widely deployed across various industries. For instance, Open Robotics is collaborating with Blue Origin (the suborbital spaceflight company established by Jeff Bezos) and NASA to cultivate Space ROS, a ROS variant engineered to fulfill the verification and validation requirements mandated for aerospace software ahead of mission deployment [63].

In essence, ROS simplifies the process for researchers and developers to create and experiment with robotics applications, courtesy of its comprehensive toolset and libraries designed for common tasks. Its modular and flexible design enables users to effortlessly

develop and disseminate new functionality, while its dynamic community ensures a rich repository of knowledge and support.

Leveraging ROS, we can design a package that enables specific LiDAR to interface with the entire system. A LiDAR driver procures data from a LiDAR sensor, processes the data, and publishes it to a ROS network in the form of a point cloud or other data types. Subsequently, other ROS nodes can subscribe to this data for executing tasks like localization, mapping, and obstacle avoidance. Generally, a LiDAR driver incorporates a configuration file that allows users to specify settings such as the LiDAR's IP address, port number, and data format. It may also comprise parameters for configuring the processing and publishing of LiDAR data, including the frame of reference, point cloud resolution, and data publishing frequency.

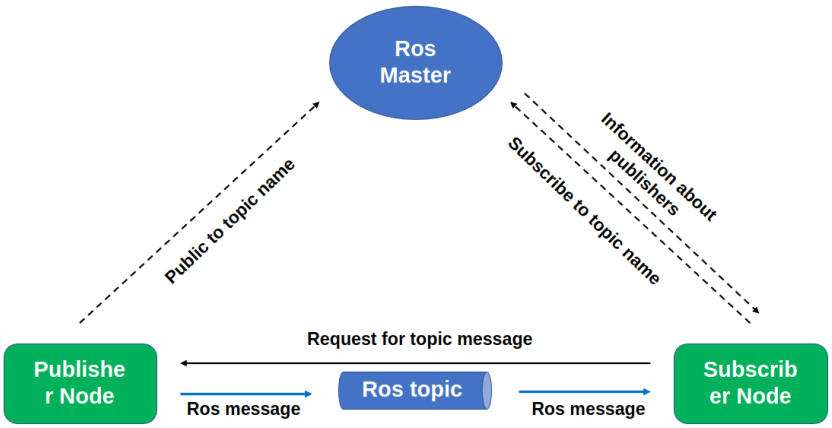


Figure 2.3: A brief working principle of ROS.

In the context of this thesis, ROS is employed, which necessitates the dissemination of ROS messages within a free space comprising three-dimensional coordinates (X, Y, Z). This mechanism facilitates inter-node information exchange and the coordination of actions within ROS, enabling the integration of our designs into more intricate robotic systems as modular components. This system setup further bestows upon our design an inherent flexibility and extensibility, allowing for the straightforward inclusion of new data types as necessitated by evolving project requirements.

2.6 Object Detection

Object detection, a critical task in the field of computer vision, aims to identify and localize specific classes of objects within an image or video. This technology finds extensive application in various fields, including facial recognition, autonomous vehicles, and security surveillance systems. Depending upon the utilized algorithms and the intricacy of the task, object detection techniques, derived from computer vision, can be broadly classified into three primary categories: Traditional Object Detection Techniques, Deep Learning-Based Techniques, and 3D Object Detection Techniques. The three categories mentioned above will be discussed in the next subsections, and the last subsections will discuss object detection based on LiDAR-as-a-camera.

2.6.1 Traditional Object Detection

Traditional object detection techniques hinge on handcrafted feature descriptors and machine learning classifiers. Widely employed feature extraction methods include the Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), and Histogram of Oriented Gradients (HOG) [64]–[66]. These algorithms identify points or regions of interest within an image and generate feature vectors predicated on the characteristics of these regions. For instance, HOG algorithms calculate histograms of gradient directions within localized sections of an image, thereby capturing shape information crucial for object identification.

Following feature descriptor computation, a machine learning classifier, such as a Support Vector Machine (SVM), is trained to differentiate between objects of interest and the background. An object detection system typically scans an image with a sliding window, extracting features and using the classifier to determine the presence of an object within each window. While these methods are fairly simple and computationally efficient, they may struggle with object appearance variations, intricate backgrounds, and other

complex conditions.

2.6.2 Deep Learning-Based Techniques

Deep learning-based techniques have significantly transformed the realm of object detection by enabling direct learning of feature representations from data, resulting in substantial improvements in accuracy.

Two-stage detectors such as R-CNN, Fast R-CNN, and Faster R-CNN [67], [68] initially generate a series of region proposals that might contain objects and subsequently classify each proposed region. This approach allows the system to concentrate computational resources on promising image areas, albeit the two-stage process can be slower than alternative methods.

One-stage detectors like YOLO [69] (You Only Look Once) and SSD [70] (Single Shot Detector) eschew the region proposal stage, directly conducting classification and bounding box regression on a dense sample of potential object locations. This approach can expedite detection times, but it may lead to a higher number of false positives.

Anchor-free detectors like CornerNet [71], CenterNet [72] predict the location and size of objects directly from feature maps, eliminating the need for predefined anchor boxes. This technique can potentially enhance detection performance by obviating the need to match ground truth objects with appropriate anchors—a process often prone to errors and difficulties.

2.6.3 3D Object Detection

3D object detection methods endeavor to identify the location and spatial extent of objects in three dimensions, thereby providing a more comprehensive understanding of a scene than their 2D counterparts. This technique is particularly crucial in applications like autonomous driving, where discerning the 3D layout of the environment is essential.

LiDAR-based techniques, such as PointRCNN and VoxelNet [73], [74], employ

point cloud data captured by a LiDAR sensor for object detection. While these methods can accurately estimate the 3D position and size of objects, they necessitate specialized hardware and substantial computational resources.

RGB-D-based methods, like Frustrum PointNets [75], utilize data from RGB-D cameras that capture both color images (RGB) and depth information (D). These methods strike a balance between accuracy and computational complexity but rely on the availability of depth information, which may not always be accessible.

Monocular image-based methods, such as Mono3D and MonoGRNet [76], [77], estimate the 3D position and size of objects from a single 2D image. While this is a challenging task due to the loss of depth information, these methods can operate with simple camera setups and are often more computationally efficient than other 3D object detection techniques.

2.6.4 Lidar-as-a-camera based Object Detection

Computer vision has made significant contributions to the field of robotic perception by enabling robots to understand and interpret visual information from their environment. Many autonomous systems across various commercial and research platforms rely heavily on computer vision for situational awareness, particularly through the use of vision sensors such as recognizing objects in the robot's environment. This allows the robot to understand the layout and structure of its surrounding environment and make informed decisions based on this information.

In recent years, the development of LiDAR technology has also played a crucial role in advancing robotic perception. LiDARs, such as the Ouster LiDAR OS0, OS1, and others, provide users with low-resolution infrared light images with a 360-degree field of view. LiDARs measure the time-of-flight of a laser signal to objects in the environment and the reflectivity of object reflections, making them not affected by changes in light such as darkness and daylight, and providing information that contains sufficient information.

Research by Xianjia et al [10]. has shown the potential of deep learning-based image object detection and instance segmentation on LiDAR signal images. Xianjia et al found that some advanced DL models can process low-resolution images by interpolating them to sufficient resolution, and have good performance. This demonstrates the potential of deep learning in expanding the capabilities of LiDARs and further enhancing the perception capabilities of robots. In the next section, we tested the effect of the YOLOv5 object detection algorithm on the information image of Ouster OS0-128, and the final results were particularly optimistic, as shown in Fig. 2.4

In our project, we applied the yolov5 algorithm to the signal images generated by the Ouster OS0-128 LiDAR in order to perform object detection and classification. YOLOv5 [78] is a real-time object detection system developed by ultralytics company. It is the latest version in the YOLO series of object detection systems, which are known for their speed and accuracy. YOLOv5 is based on a convolutional neural network (CNN) architecture, which is trained to predict object bounding boxes and class labels from input images. The architecture consists of several layers, including a backbone network, neck, and head layers. The backbone network extracts features from the input image, while the neck and head layers refine these features and predict the object bounding boxes and class labels. One of the key features of YOLOv5 is its use of anchor boxes, which are pre-defined bounding boxes used to guide the prediction of the object bounding boxes. This allows YOLOv5 to be more efficient and accurate compared to other object detection systems that do not use anchor boxes.

We applied the yolov5 algorithm to the signal images generated by Ouster OS0-128, and our experiments showed good results, as shown in Fig. 2.4.

2.7 3D Piont Could Processing

Point cloud computation is the process of processing and analyzing point clouds to extract useful information or generate new representations. The point cloud clustering

Figure 2.4: The results of running YOLOV5 on the signal images from the Ouster OS0-128.

algorithm, point cloud registration algorithm, and point cloud noise removal algorithm are introduced in this chapter.

2.7.1 Point Cloud Clustering

Point cloud clustering is a specialized technique utilized to partition a set of points within a point cloud into unique groups or clusters. This method is predominantly employed in the analysis of 3D point cloud data generated by sensors like 3D LiDAR (Light Detection and Ranging) and RGB-D cameras. This technique has significant applications in object recognition, scene segmentation, and 3D reconstruction, contributing to an automated separation of different objects or structures within a scene, and thereby facilitating more precise analysis and interpretation of the data. The prevalent approaches to point cloud clustering, including density-based [79], region-growing [80], and model-based methods [81], offer distinct advantages and limitations in their application.

Density-based methods, such as DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [79] [82], identify clusters based on the density of points within a region. Points in dense regions are assigned to the same cluster, while points in sparse areas are typically treated as noise or outliers. This technique is effective in discovering clusters of arbitrary shapes and sizes, and it is resistant to noise. However, it may struggle in datasets with varying density regions.

Region-growing methods [80], on the other hand, initiate clusters by growing a re-

gion around a seed point. This approach is often used in the segmentation of 3D point clouds, where the geometric relationships between points are used to grow the regions. The primary advantage of region-growing methods, such as the Euclidean Cluster Extraction algorithm, is their ability to identify non-convex clusters based on spatial proximity. However, these methods can be sensitive to the choice of seed points and are susceptible to noise and varying densities.

Model-based clustering methods [81], like Gaussian Mixture Models (GMM) [83] [84], use a probabilistic model to assign points to clusters. They fit a parametric model to the data and assign points to clusters based on their likelihood under the model. These methods can effectively handle clusters of different sizes and shapes and are capable of inferring missing data. Nevertheless, they can be computationally expensive and rely on the assumption that data inherently follows the model, which may not always hold true. In conclusion, each point cloud clustering method provides its unique set of benefits and challenges. Their efficiency largely depends on the characteristics of the dataset, including its density, noise level, and the inherent geometric or spatial relationships within the data. Thus, the selection of a suitable clustering approach should be influenced by the specific requirements of the task at hand.

2.7.2 Point Cloud Registration

Point cloud registration involves aligning two or more point clouds to a common coordinate system. This process is commonly utilized in robotics and computer vision for a range of purposes, such as 3D mapping, object recognition, and pose estimation. There are various methods for performing point cloud registration, including feature-based and point-based approaches. An optimization algorithm is typically employed to minimize the difference between the point clouds by adjusting their pose. The effectiveness of the registration depends on the chosen optimization algorithm and the initial pose of the point clouds.

In our design, we employ Generalized Iterative Closest Point (GICP) [85], which is an algorithm for point cloud registration, which aligns two point clouds by minimizing the error between point clouds. It is an extension of the Iterative Closest Point (ICP) algorithm, a widely used technique for point cloud registration. GICP is designed to handle point clouds with non-uniform density, which can be a problem for traditional ICP. It does this by weighting the points in the point cloud using a covariance matrix, which allows it to give more weight to points with more information. GICP works by iteratively aligning point clouds by minimizing the error between point clouds. In each iteration, it finds the closest point in the two point clouds and adjusts the pose of one of the point clouds to minimize the error. It continues this process until the error between point clouds is minimized or the algorithm converges. GICP has been shown to be effective for various point cloud registration tasks and is often used in robotics and computer vision applications. Finally, we use this algorithm to obtain a dense and accurate ground turth map, as shown in Fig. 1.1.

2.7.3 Point Cloud Noise Removal

Point cloud noise removal is a technique used to clean and filter point cloud data, which is commonly used in 3D scanning and mapping applications and it is an important step in the point cloud processing pipeline. These points are typically collected using laser scanners or other 3D sensing devices, and they can contain a significant amount of noise, such as outliers, errors, and inaccuracies. Noise removal is an essential step in the point cloud processing pipeline as it improves the overall quality and accuracy of the data. The goal of noise removal is to separate the useful information from the noise, while preserving the integrity of the underlying data. There are many different methods for removing noise from point clouds, including statistical methods, spatial methods, and machine learning techniques. Statistical methods are based on the assumption that noise is random and follows a specific distribution. These methods typically involve identify-

ing and removing points that fall outside of a specified threshold or range. For example, the RANSAC [86] (Random Sample Consensus) algorithm is a popular statistical method that uses a random sampling technique to identify and remove outliers from point clouds. Spatial methods, on the other hand, are based on the spatial properties of the data. These methods typically involve grouping points into clusters or regions, and then removing points that do not conform to the characteristics of the surrounding points. For example, the DBSCAN [79] (Density-Based Spatial Clustering of Applications with Noise) algorithm is a popular spatial method that groups points into clusters based on their density and proximity to other points. Machine learning techniques are also used for point cloud noise removal. These methods typically involve training a model on a set of labeled point cloud data, and then using the model to classify and remove noise from new point cloud data. For example, deep learning-based methods like Convolutional Neural Networks (CNN) can be trained to classify points as noise or signal, and then remove the noise points.

The point cloud map used in this study contains a variety of scenes, including forests, open roads, indoor halls, and corridors. The characteristics of point cloud data in indoor and outdoor environments are quite different. In indoor environments, there are more planar features such as walls, tables, and cabinets, while in outdoor environments, there are more objects with variable characteristics such as branches and trunks. The significant difference between indoor and outdoor environments makes it difficult to use spatial methods for point cloud noise removal, as these methods require specific parameter settings such as minimum radius and minimum number of points. Generalization is a challenge for machine learning-based point cloud noise removal techniques because the model needs to be able to perform well on new, unseen scenes, even if it has not been trained on them before. Additionally, these techniques can be computationally expensive, particularly when dealing with large point clouds. Our ground truth map have a high number of points, such as the indoor hall point cloud map which consists of 180,000 points. This can make it

challenging to use these techniques in real-time applications or on resource-constrained devices. After evaluating various point cloud noise removal techniques, it was determined that statistical filtering was the most appropriate method for eliminating noise in the point cloud data. This decision was based on the following reasons:

- Robustness: Statistical methods have been shown to be robust to outliers and capable of handling large amounts of noise in the point cloud data.
- Versatility: These techniques can be applied to different types of noise, including Gaussian noise, impulse noise, and mixed noise.
- Efficiency: Many statistical techniques can be implemented using fast algorithms, allowing for real-time processing of large volumes of point cloud data.

In light of the above considerations, statistical filtering was deemed the most suitable method for this specific application.

3 Multi-Modal LiDAR SLAM Benchmark

In this chapter, the overall design of the Multi-model slam benchmark is first introduced in detail. The design includes the design principles and layout of the software and hardware, and then the execution steps of the SLAM benchmark and related experimental settings are shown, and finally the experimental results are comprehensively analyzed.

3.1 Design Overview

In this section, the hardware and software design of the data collection platform for the Multi-model SLAM Benchmark is discussed in detail. The algorithm for generating the ground truth map is carefully introduced, as well as the calibration of multiple LiDARs and the data structure of the multi-modal LiDAR dataset.

3.1.1 Hardware Design

The primary objective of our sensor system is to offer data from a diverse range of LiDAR sensors, each boasting unique characteristics. These range from cutting-edge, cost-effective solid-state LiDARs, to 3D rotating LiDARs of varying resolutions and vertical fields of view, as well as LiDAR-based cameras. To this end, our data collection apparatus comprises three rotating LiDARs: the 16-channel Velodyne LiDAR (VLP-16),

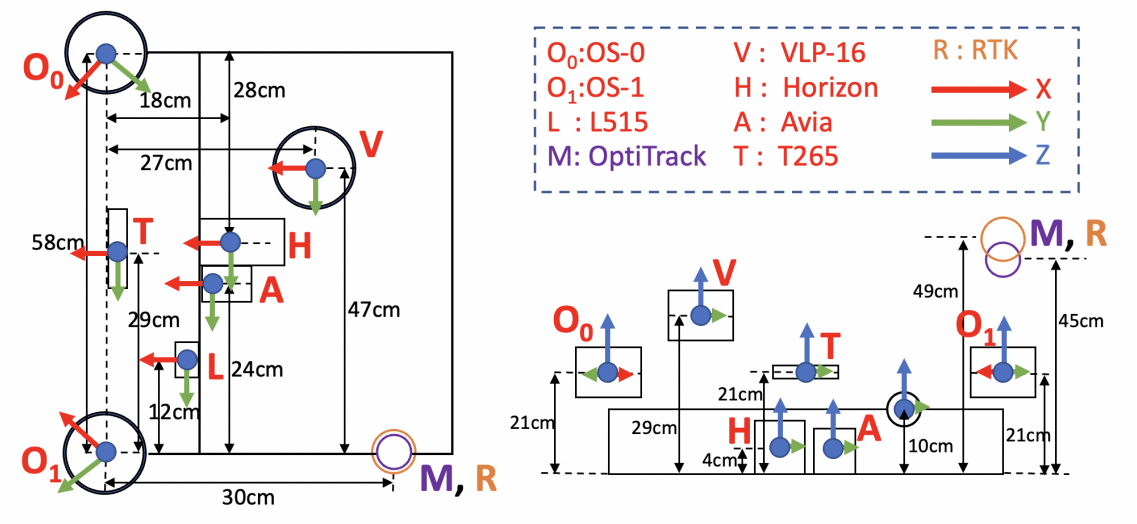


Figure 3.1: Our data collection interface, shown from above (on the left) and from the front (on the right).

the 64-channel Ouster LiDAR (OS1), and the 128-channel Ouster LiDAR (OS0). In the domain of solid-state LiDAR, we have incorporated two models from Livox: the Horizon, characterized by its near-rectangular field of view, and the Avia, which possesses an almost circular visual field. The configuration is rounded off with the inclusion of the Intel RealSense L515 LiDAR camera.

In terms of the spatial arrangement, both the Horizon and Avia LiDARs are centrally mounted, oriented forward. To their left front sits the L515 camera. The OS0 and OS1 sensors are positioned slightly elevated; the OS1 is tilted 45 degrees to the right and the OS0, 45 degrees to the left. Crowning the setup, the Velodyne LiDAR is positioned with its x-axis front-facing. For a more granular insight into the relative distances, positions, and orientations, one can consult Fig. 3.1. Affixed atop an aluminum rod is the Optitrack marker set, purposed for MoCAP-based ground truth. This placement strategy optimizes its visibility and detection span, conveniently adjacent to the Holybro H-RTK F9P Helical GPS Module. On the technical front, to guarantee brisk and uninterrupted data transfer, the LiDAR is tethered to a Gigabit Ethernet router and an integrated computer, which boasts specs like an Intel i5-9300HF processor, 16 GB DDR4 RAM, and an SSD storage of 1 TB. The Optitrack system is hardwired to this computer via Ethernet, but on a distinct

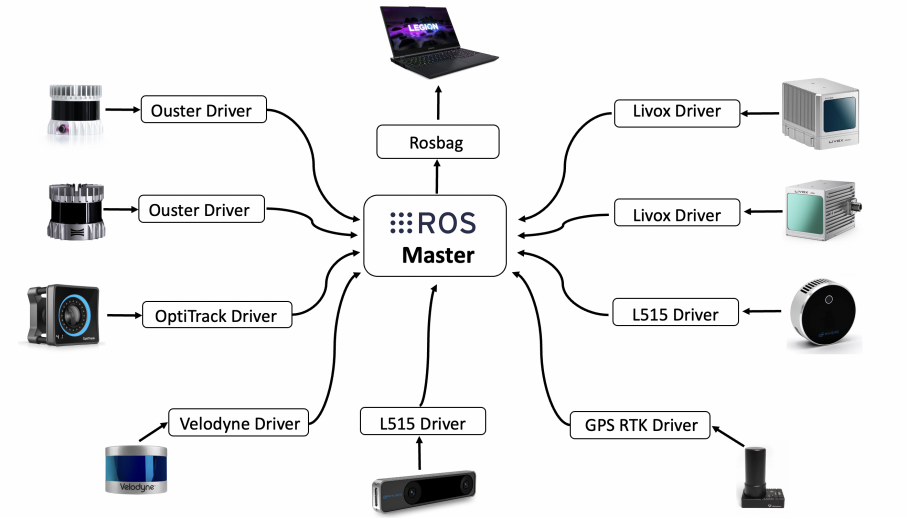


Figure 3.2: ROS interfaces and sampling rates for the distinct LiDAR sensors integrated into our platform.

interface separate from the LiDAR. Rounding off the connectivity, the RealSense L515 camera interfaces through a USB 3.0 port.

3.1.2 Software Information

Our software infrastructure is constructed exclusively on ROS Melodic, running on Ubuntu 18.04. A visual representation of the ROS drivers and the data publishing rates for various sensors can be found in Fig. 3.2. Given the absence of hardware signals for synchronizing sensor data, similar to other documented datasets in references like [87], we tackle the data synchronization challenge by operating all sensor drivers and data recording applications locally on a high-performance computing system. This strategy, combined with our networking hardware, effectively diminishes data transmission latency both at the hardware and software junctures, ensuring data is timestamped right at the ROS driver level. Broadening the dataset's utility, we've also incorporated timestamps from the built-in internal oscillators of both Livox and Ouster LiDARs. This is pertinent for both point cloud and IMU data. Moreover, every ROS message header includes its own timestamp.

Figure 3.3: From an aerial perspective of the point cloud data assembled during the calibration of diverse LiDARs, the Livox Horizon and Avia are distinctly represented by shades of red and green. Concurrently, the VLP-16, OS1, OS0, and L515 sensors manifest themselves through point clouds colored in respective hues of purple, yellow, blue, and black.

3.1.3 Calibration and Synchronization

Efficient extrinsic parameters calibration is crucial to multi-sensor platforms, especially for handmade devices where the extrinsic parameters may change due to unstable connections or distortion of the material during transit. Similar to our previous work [9], we calculated the extrinsic parameter of sensors before each data collecting process. Fig 3.3 shows the calibration result of sample LiDAR data from one of the indoor data sequences. Different to our previous work[9], where the timestamp of Ouster and Livox LiDARs are kept based on their own clock, we synchronized all LiDAR sensors in ethernet mode via the software-based precise timestamp protocol (PTP) [88]. 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 3 ms, as shown in Fig. 3.4.

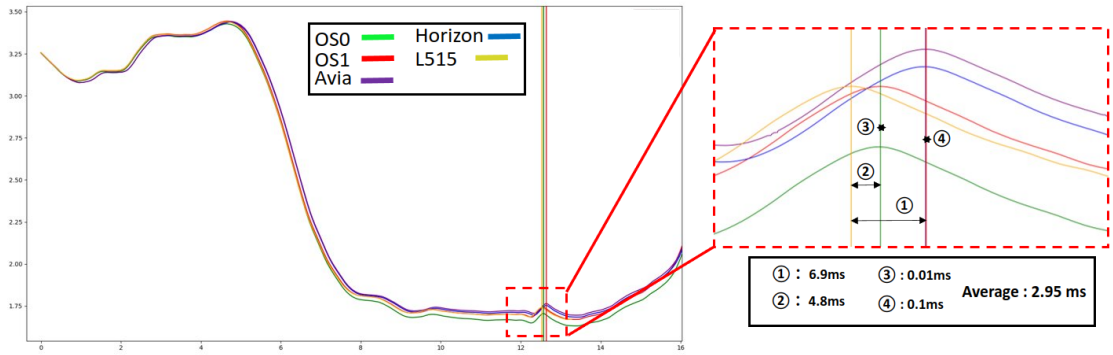


Figure 3.4: The change in the yaw value of the IMU of each LiDAR in the dataset. It can be seen from the picture that the average time offset of the dataset does not exceed 3ms.

3.1.4 SLAM assisted Ground Truth Map

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 [11], where a survey-grade 3D imaging laser scanner Leica BLK360 scanner is unitized 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 clouds within the FoV increases with time. Therefore, we integrated multiple frames when the platform was 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 clouds 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-degree 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. Fig. 1.1 shows a ground truth map of sequence indoor08 generated based on Algorithm 1.

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**

  $\mathcal{M}_{ap} \leftarrow ICP(\mathcal{S}_{cache}, \mathbf{p}_k, \mathcal{M}_{ap})$;

  $\mathcal{S}_{cache}.clear()$;

Let $\mathcal{P}_{sk}$ be the point clouds produced by the spinning LiDAR, $\mathcal{P}_{dk}$ be the point clouds 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

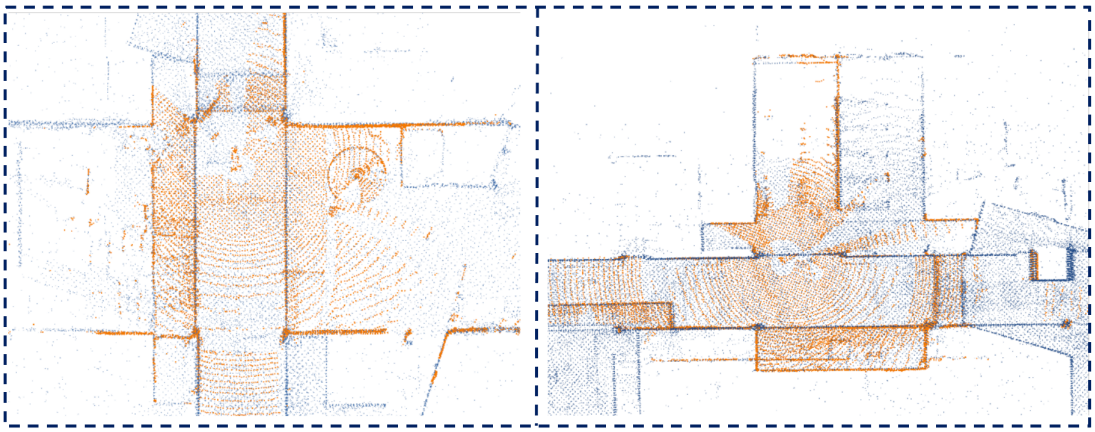


Figure 3.5: NDT localization with ground truth map (blue) where the current laser scan (orange) is aligned.

diverse environments. Therefore, LeGo-LOAM [29] 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 clouds 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 clouds $\mathcal{M}_{ap}$ 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.

After the ground truth map is generated, we employ the normal NDT method in [89] to match the real-time point cloud data from spinning LiDAR against the HR map as the Fig. 3.5 shows to get the platform position in ground truth map. The matching result from the NDT localizer is treated as the ground truth.

3.1.5 Data Structures of the Dataset

The data used in this research is collected using the Robot Operating System (ROS) and is stored in the rosbag format, which is a widely used standard in the robotics com-

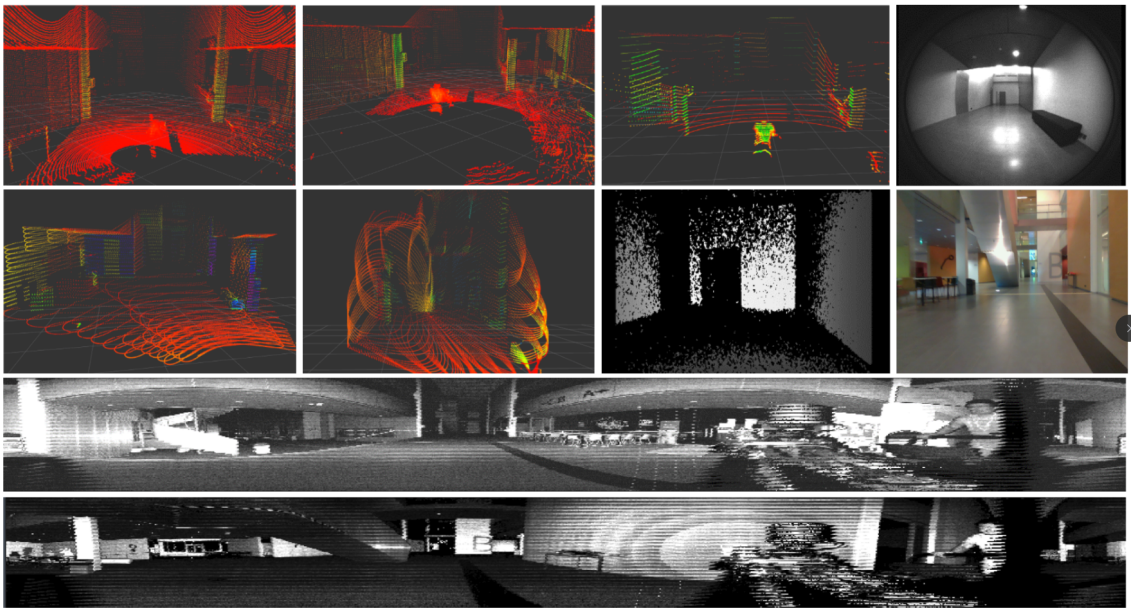


Figure 3.6: Our dataset is captured by a rich set of sensors. A subset of the data from the Indoor11 sequence is visualized here. The first row displays laser radar data from OS1, OS0, and VLP-16, as well as a fisheye image from T265; the second row displays point cloud data from Avia and Horizon, as well as depth and RGB images from L515; the third and fourth rows display images from OS1 and OS0 respectively.

munity. A subset of the data from various sensors is displayed in Fig. 3.6. The dataset includes the following types of data:

- **Spinning LiDARs Point Clouds**: Data from three spinning LiDARs (VLP-16, OS0-128, and OS1-64) is captured using the ROS message type"*sensor_msgs/PointCloud*". Every point in this cloud carries four values: x, y, z (the local Cartesian coordinates), I (laser reflectance for that specific point).

- **Solid-State LiDARs Point Clouds**: The Avia and Horizon solid-state LiDARs utilize Livox's proprietary format, labeled "*livox_ros_driver/CustomMsg*". This format includes a base timestamp for the inaugural point of each frame and offset times for the ensuing points. This design accommodates the unique, non-repetitive pattern of these LiDARs, ensuring time differences between points are discernible. Such data is pivotal for the de-skew process that rectifies point cloud data distortions due to sensor motion [90]. Recognizing the versatility of standard ROS

messages for visualization and compatibility [91], tools have been supplied for data conversion. This process transforms the Livox format into the standard "*sensor_msgs/PointCloud*", yielding data points with five values:x, y, z, I and C - the last representing the line number and the point's timestamp.

- **LiDAR Camera Imagery**: The RealSense L515 LiDAR camera yields RGB images (resolution: 1920x1080) and depth images (resolution: 1024x768) at a 10 Hz frequency. They are channeled through the message type sensor_msgs/Image. Depth estimations benefit from the integral LiDAR sensor.

- High-Resolution Spinning LiDAR Imagery: Both high-res Ouster LiDARs, OS0-128 and OS1-64, provide fixed-resolution range images, near-infrared imagery, and signal images. Each pixel imparts data on distance, light intensity, and object reflectance. These images are relayed at 10 Hz, boasting a 16-bit linear photo response per pixel, under the message type *sensor_msgs/Image*.

- **Inertial Data**: Four integrated 6-axis IMU sensors exist across the Ouster and Livox LiDARs. These IMUs, comprising 3-axis gyroscopes and 3-axis accelerometers, broadcast data at frequencies of 100 Hz and 200 Hz for the Ouster and Livox respectively. The rosbags carry this IMU data under the type *sensor_msgs/Imu*.

- **Ground Truths**: Various sequences hold distinct ground truths. For indoor settings, MoCAP or Slam/ICP techniques are utilized. Conversely, outdoor sequences lean on GNSS/RTK. MoCAP data is framed in the format *geometry_msgs/PoseStamped*. GNSS/RTK adheres to *sensor_msgs/NavSatFix*, while Slam/ICP-derived ground truth is also in the *geometry_msgs/PoseStamped* format.

3.2 Implementation

This section describes in detail the details of the data collection of the Multi-model slam benchmark, including the environment of the dataset sequence, how to obtain ground truth from the high-definition point cloud map, and the details of the Odometry benchmark and run-time evaluation experiments.

3.2.1 Collect Data for Multi-modal LiDAR Dataset

In order to implement the mobile data collection platform, as outlined in Section 3.1.1, a total of five different LiDARs and two LiDAR cameras were incorporated into the platform. The platform also includes an independent power supply to provide power, as illustrated in Fig. 1.3. In order to ensure the completeness of the dataset, a variety of environments such as indoor offices, long corridors, halls, forests, and open roads were selected as the sequences for the dataset, as illustrated in Table 3.1.

Fig. 3.7 illustrates the various sequences of data sets, composed of map data samples. The first row, from left to right, shows the sequence *Forest01*, the sequence *Road03*, and the sequence *Indoor10*. The second row, from left to right, displays the sequence *Indoor06/Indoor07* and the sequence *Indoor09*, followed by the sequence *Indoor11*. Two sequences are provided for the forest environment. The data was collected in the forest of Turku, Finland (60°28014.3”N22°19054.8”E) during winter, specifically on snow-covered ground. Sequence *Forest01* features a square-shaped trajectory, while *Forest02* comprises a straight trajectory. Both sets contain MoCAP data. These sequences can support research in various areas such as tree counting and tree trunk diameter estimation. The significant difference in the structure of the environment between urban and forest settings also allows for a better evaluation of LiDAR-based general odometry, localization, and mapping algorithms, as well as the robustness of these algorithms.

Indoor dataset consists of 11 data sequences, with six of them being a new sequence

Table 3.1: List of data sequences in our extended dataset. The table includes the sequences introduced in our previous work [9], together with new sequences showcasing new ground truth data sources. The five LiDARs indicated (5x LiDARs) and cameras are listed in Table 4.1.

Sequence	Description	Ground Truth	Sensor setup
Forest01-02	Previous dataset [9]	MoCAP/ SLAM	5x LiDARs
Indoor01-05	Previous dataset [9]	MoCAP/ SLAM	L515
Road01-02	Previous dataset [9]	SLAM	Optitrack
Indoor06	Lab space (easy)	MoCAP	
Indoor07	Lab space (hard)	MoCAP	5x LiDARs
Indoor08	Classroom space	SLAM+ICP	L515
Indoor09	Corridor (short)	SLAM+ICP	T265
Indoor10	Corridor (long)	SLAM+ICP	Optitrack
Indoor11	Hall (large)	SLAM+ICP	GNSS
Road03	Open road	GNSS RTK	

called *Indoors06-11*, which was collected in various indoor environments such as classrooms, labs, and corridors of ICT-City in Turku, Finland. Sequence *Indoors06-7* was collected in a laboratory, where data from a motion capture system (MoCAP) is also available. This sequence features faster rotations and sudden movements, as the sensors were positioned closer to objects in front of and around them. This poses a more challenging situation for odometry estimation algorithms that are primarily based on scan-matching methods, as most of the solid-state LiDAR views are covered by single-featured objects or walls in close proximity.

The environment of sequence *Indoor10* is a rectangular corridor on the 3rd floor of ICT-City, which contains less rotation and sudden movement. The data of sequence *Indoor08* was recorded in a classroom of ICT-City, where the sensors moved in an ellip-

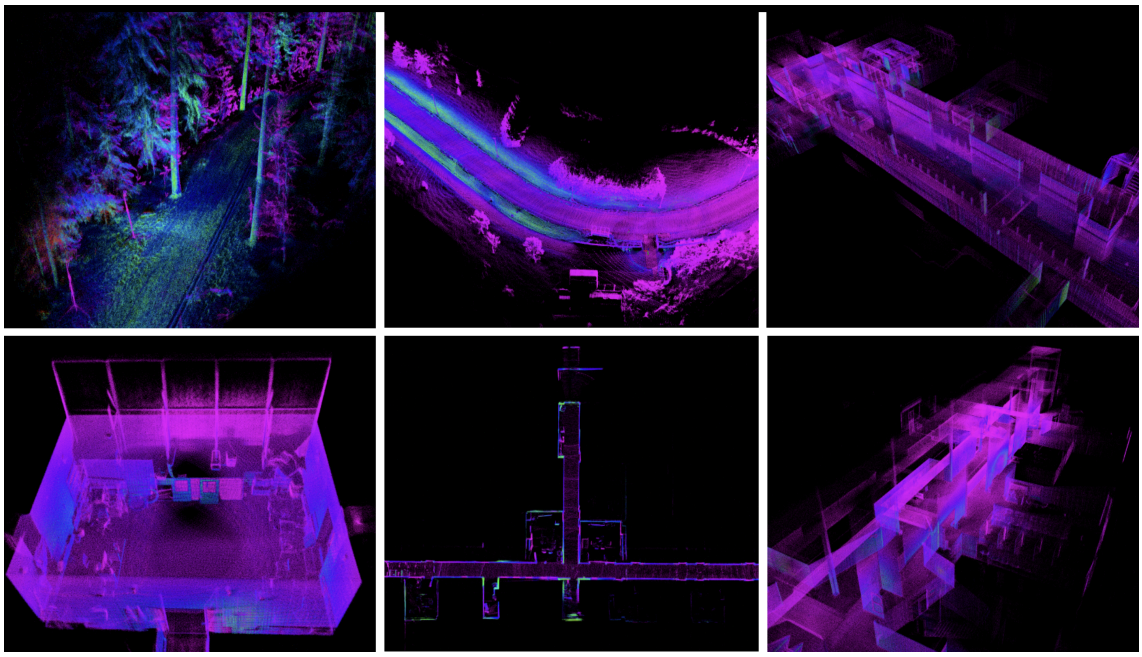


Figure 3.7: Samples of map data from varied dataset sequences are presented. Arranged from left to right and subsequently top to bottom, the visualizations showcase maps derived from a forest, an open road, an elongated corridor, followed by a spacious indoor lab area, another extensive corridor, and lastly, a hall.

tical trajectory. While the number of rotations was more, the rotation speed was slower compared to the sequence *Indoors07*. The environment corresponding to the sequence *Indoor09* is a narrow and long corridor on the sixth floor of ICT-City. It includes not only a relatively gentle straight corridor, but also a rotating place where the sensors are very close to the wall. Sequence *Indoor11* was collected in the hall on the first floor of ICT-City, which roughly contains all possible environments, such as a wide hall, a narrow storage cabinet wall, and a narrow corridor on a slope. Finally, the dataset includes there open road environmental sequences around the ICT-City building in Turku, Finland. The length of the *Road01* is more than 50m, while the traversed length of the track in *Road02* and *Road03* is about 500 m. The difference between Sequence *Road02* and *Road03* is that the sequence *Road03* uses Real Time Kinematic (RTK) to record the ground turth track, which has higher positioning accuracy and no cumulative error.

In order to improve the accuracy of the ground truth map, it is necessary to remove

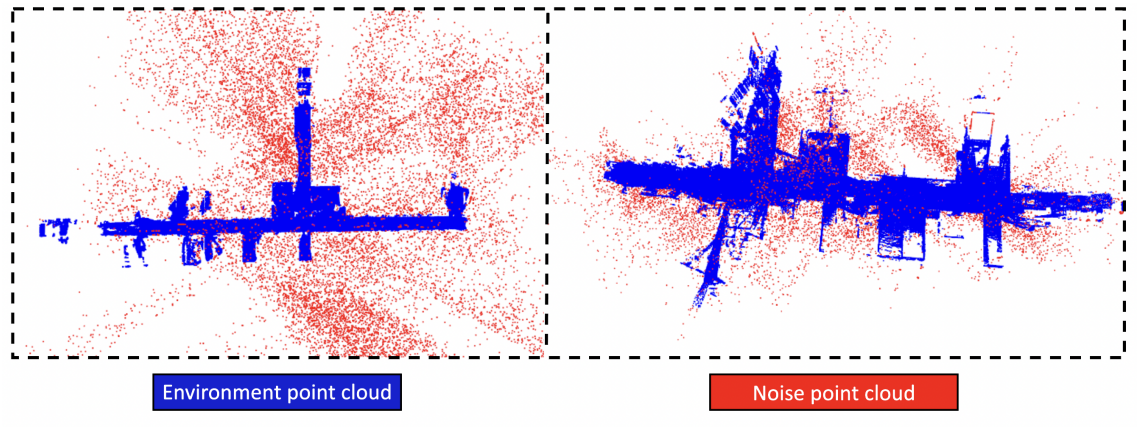


Figure 3.8: Remove the noise in the ground turth map, the red point cloud is the noise point cloud, and the blue point cloud is the environment point cloud.

noise, outliers, errors, and inaccurate points from the point cloud. In this study, we employed statistical filtering to eliminate point cloud noise, as shown in Fig. 3.8. The effectiveness of this approach is demonstrated in the results.

3.2.2 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. Fig. 3.9 (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 Fig. 3.9 (d) show that the maximum difference does not exceed 5 cm.

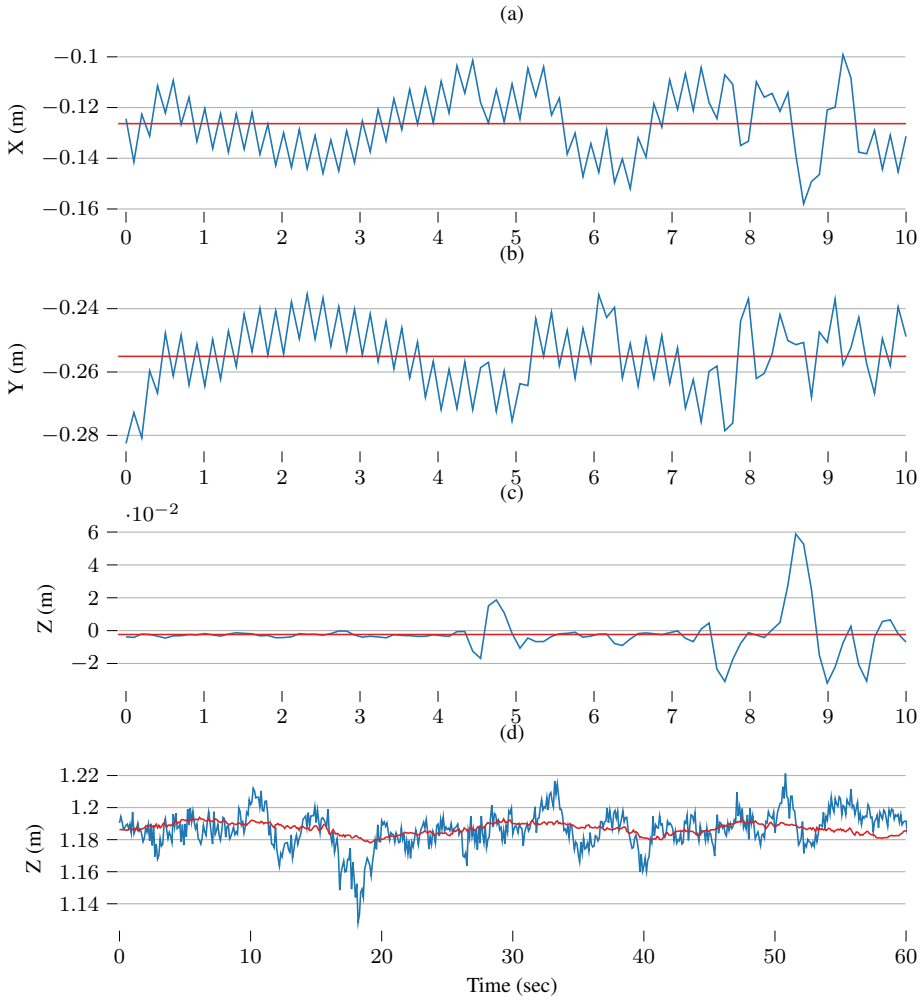


Figure 3.9: **(a) (b) (c):** Ground truth position values for the first 10 seconds of the dataset when the device was stationary. Red lines show the mean values over this period of time. **(d):** Comparison of NDT-based ground-truth z-values (green) to MoCAP-based z-values (red) over the course of 60 seconds of the dataset while the device was in motion.

3.2.3 Setup for 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

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

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

filter-based methods, FAST-LIO (FLIO) [3] [92], 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] [91] 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.

In this study, the operating system used is Ubuntu 18.04. All SLAM algorithms were executed in the Robot Operating System (ROS) Melodic environment. Additionally, the default configuration value was used for each hyperparameter in each SLAM system.

3.2.4 Setup for Run-time Evaluation

We conducted this experiment on 4 different platforms. First, 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). Then, the Jetson Xavier AGX, a popular computing platform for mobile robots, 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 Nvidia Xavier NX is also 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. For the NX, we choose the 15 W power mode (all 6 cores). Finally, the UP Xtreme board features 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 memory utilization is measured with a ROS resource monitor tool [6]. 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.

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

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

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

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

Table 3.2: Absolute position 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

3.3 Results

In this study, we evaluated popular 3D Lidar SLAM algorithms in multiple data sequences of various scenarios, including indoor, outdoor, and forest environments.

3.3.1 LiDAR Odometry Benchmarking

We provide a quantitative analysis of the odometry error based on the ground truth in Table 3.2. 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 errors (APE) [93] is employed as the core evaluation metric. We calculated the error of each trajectory with the open-source EVO toolset [7].

The findings indicate that FAST_LIO, when paired with high-resolution spinning LiDARs OS0 and OS1, delivers the most consistent and robust performance, adeptly completing trajectories across various sequences with commendable precision. This ro-

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

bustness is particularly evident in the *Indoor09* sequence, which features an elongated corridor. In this challenging environment, while other methods faltered, FAST_LIO using high-resolution LiDAR remained unyielding.

In outdoors, solid-state LiDAR-based SLAM systems, exemplified by LIOL_Hori, match or even surpass the performance of rotating LiDARs when paired with suitable algorithms. However, their efficacy diminishes considerably in indoor settings. Examining the open road sequence, *Road03*, all SLAM methods showcased stellar performance, producing trajectories without notable hindrances. In contrast, for the indoor sequence, *Indoor06*, both Avia-based and Horizon-based FLIO systems managed to trace the sensor trajectory, but they did so with significant drift accumulation.

Across all sequences, the spinning LiDAR-based methods demonstrated consistent efficiency. This was anticipated given their comprehensive environmental view, which typically possesses distinct geometrical features. Observing the *Indoor10* sequence, which also features a lengthy corridor, nearly all methods reconstructed the entire trajectory. Standout performances were evident from OS0-FLIO and OS1-FLIO, as they achieved precise alignment between the start and end positions. This superior performance can be attributed to OS0's channel superiority over OS1, which we theorize results in reduced cumulative angular drift.

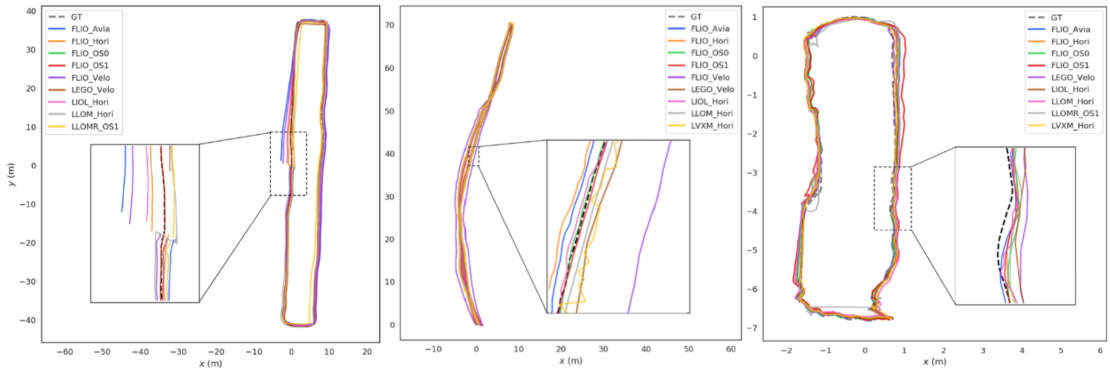


Figure 3.10: From left to right, the trajectory comparison of sequences *Indoor10*, *Road03*, and *Forest01*

In addition to the quantitative trajectory analysis, we visualize trajectories gener-

ated by selected methods in 3 representative environments (indoors, outdoors, forest) in Fig. 3.10. Full reconstructed paths are available in the dataset repository.

3.3.2 Mapping Quality Comparison

In an analysis of the mapping results produced by various LiDARs in indoor settings, as depicted in Fig. 3.11, we find intriguing distinctions between the performances of different methodologies and devices.

Fig. 3.11 vividly showcases that the LIOL approach, when used with solid-state LiDAR, yields the most intricate and lucid map structures. Intriguingly, these maps were produced using the default configurations for each method, without any alterations to parameters, such as map update frequencies. This observation aligns perfectly with the quantitative outcomes gleaned from experiments in forested terrains using identical sensors and algorithms.

A deeper dive into Fig. 3.11 reveals that among the methods studied, the Horizon-based LIOL emerges as the top performer in mapping capability. However, its prowess diminishes in intricate environments, as demonstrated by sequences like *indoors06-09*, where LIOL encounters mapping failures attributed to drift. Nonetheless, the FLIO method, when coupled with OS0 and OS1 LiDARs, also demonstrates commendable mapping proficiency. This is likely owed to the expansive field of view (FOV) and superior resolution presented by OS0 and OS1. When these performances are set against Velodyne's efforts, the latter seems to fall short. Velodyne's larger resolution is its Achilles' heel, evident in its near inability to accurately reconstruct the letter B sign, as displayed in Fig. 3.11. Methods such as LVMX, LLOM, and LLOMR predominantly prioritize the estimation of the mobile platform's pose over point cloud mapping capabilities. Consequently, the maps they generate are comparatively less detailed and lack the clarity seen with other methods.

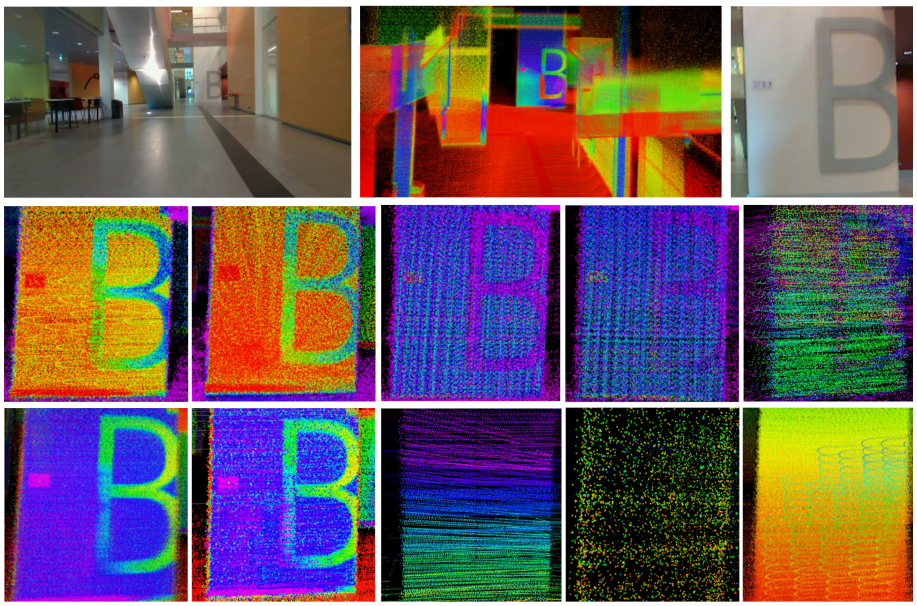


Figure 3.11: Qualitative comparison of the mapping quality. Frist row from left to right shows RGB full view image, full view Horizon-based LIOL and close view RGB image. Second row row from left to right shows OS0, OS1, Velodyne, Avia and Horizon-based FLIO. Bottom row from left to right shows the Horizon-based LIOL, Horizon, OS1-based LLOM and LLOMR, Velodyne's LeGo-LOAM maps and Horizon-based LVXM, respectively.

3.3.3 Run-time Evaluation across Certain Computing Platforms

We conducted this experiment on 4 different platforms. First, 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). Then, the Jetson Xavier AGX, a popular computing platform for mobile robots, 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 Nvidia Xavier NX is also 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. For the NX, we choose the 15 W power mode (all 6 cores). Finally, the UP Xtreme board features 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 memory uti-

Table 3.3: Average run-time resource (CPU/RAM) utilization and performance (pose calculation speed) comparison of selected SLAM methods across multiple platforms. For the pose publishing frequency, the data is played at 15 times the real speed. CPU utilization of 100% equals one full processor core.

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

lization 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 SLAM system into same version, and each hyperparameter in the SLAM system is configured with the default values. The results are shown in Table 3.3. The memory utilization of each selected SLAM approach among the two processor architectures platforms are 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 towarsd the low end (see Table 3.2), and has a very low pose publishing frequency. Fast-LIO performs well, especially on embedded 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.

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

4 LiDAR-as-a-camera Based UAV Tracking

In this chapter, firstly, the overall design of LiDAR Based UAV tracking is introduced in detail, mainly related to hardware and the principle of fusion of Ouster LiDAR signal image and point cloud, and a multi-method comparison experiment is designed, and finally, its experiment is comprehensively analyzed

4.1 Design Overview

In this section, the hardware equipment used in LiDAR-based UAV tracking is discussed in detail, and the software design of the UAV tracking algorithm is introduced. The fusion algorithm of Ouster LiDAR signal image and point cloud is emphasized.

4.1.1 Hardware Design

The first consideration in the hardware design for this UAV tracking system is the identification of an appropriate LiDAR system. In the acquisition of a LiDAR sensor or system for a specialized undertaking, a careful assessment of several critical parameters is indispensable. These encompass but are not confined to FOV, resolution, range, and operational environment.

Among these elements, FOV and resolution are frequently deemed as principal con-

siderations. FOV, in reference to a LiDAR sensor or system, denotes the angular range within which it can detect objects. It is of utmost importance to select a LiDAR system with a FOV that is well-suited for the intended application. In contrast, the resolution of the LiDAR sensor or system pertains to the density of detectable points or beams within a delineated area. A LiDAR system of higher resolution has the capacity to generate more granular data, thereby improving the precision of the output.

Table 4.1: The sensor specifications for the dataset introduced in our previous work [9].

	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	BOSCH BMI085	LiDAR camera	N/A	70°×43°(±3°)	N/A	9 m	30 Hz	-
RealSense T265	BOSCH BMI055	fisheye cameras	N/A	163±5°	N/A	N/A	30 Hz	-

For a detailed comparison, we may refer to Table 4.1 from our previous work [94], which delineates the characteristics and parameters of several LiDAR systems. Notably, the OS1-64 system offers configurable angular resolution by adjusting the vertical field of view. Livox LiDARs employ a non-repetitive scanning pattern to attain higher angular resolution, albeit at the cost of extended integration times. Range measurements for LiDARs are typically provided by the manufacturer and are based on 80% Lambertian reflectivity and 100 klx sunlight, a methodology distinct from other range measurement techniques. By comparing the FOV parameters of various LiDAR systems, it becomes evident that rotating LiDARs generally exhibit superior FOVs. The Ouster OS0-128 stands out as the optimal choice, with coverage that is nearly comprehensive. Fig. 4.1 offers a more intuitive depiction of the FOV discrepancies among several LiDAR systems, revealing the FOV of the Ouster OS0-128 to be approximately 16 times and 6 times that of the Livox Horizon and Livox Avia, respectively.

When assessing the resolution, as per Table 4.1, it is noteworthy that solid-state LiDAR does not present explicit resolution data. This omission is due to Livox's adoption of an innovative scanning pattern [95] characterized by a non-repetitive scan mode, causing the LiDAR's light beams to accumulate over time. For a more intuitive comparison of the resolution between the Ouster OS0-128 and Livox Horizon, please refer to Fig. 4.1. The image at the bottom represents a single frame of signal data from the Ouster OS0-128, while images a, b, c, and d respectively present signal data merged from 1 frame, 3 frames, 5 frames, and 10 frames of Livox Horizon point cloud data. As time progresses, the signal image of Livox Horizon becomes progressively distinct. Upon reaching a cumulative time of 0.5 seconds, the image clearly exhibits a drone, though two circular regions on the left and right remain comparatively blurry. A comprehensive view of the full range only emerges when the cumulative time reaches 1.0 seconds.

In contrast, the signal image of the Ouster OS0-128 does not rely on the accumulation of time, providing a clear representation of both drone and human figures within a single frame. It should be noted that while the Livox Horizon can generate a point cloud with higher density than that of the Ouster OS0-128, provided there is sufficient time for accumulation, this method proves inadequate for tracking drones due to their potential maximum speed of 15m/s.

From the above comparison, we can conclude that Ouster OS0-128 is superior to Livox LiDAR in terms of FOV and instantaneous resolution.

Once the appropriate LiDAR sensor is selected, the remainder of the system's hardware design is relatively straightforward. The components involved in this design are minimalistic, comprising an Ouster OS0-128 LiDAR, an associated power supply, an Ethernet cable, and a laptop computer. The Ouster OS0-128 LiDAR is strategically placed at an elevated height from the ground to maximize its visibility and detection range.

To ensure low latency and high-speed data transmission, the Ouster OS0-128 LiDAR is connected to a gigabit Ethernet router. The system is complemented by a laptop

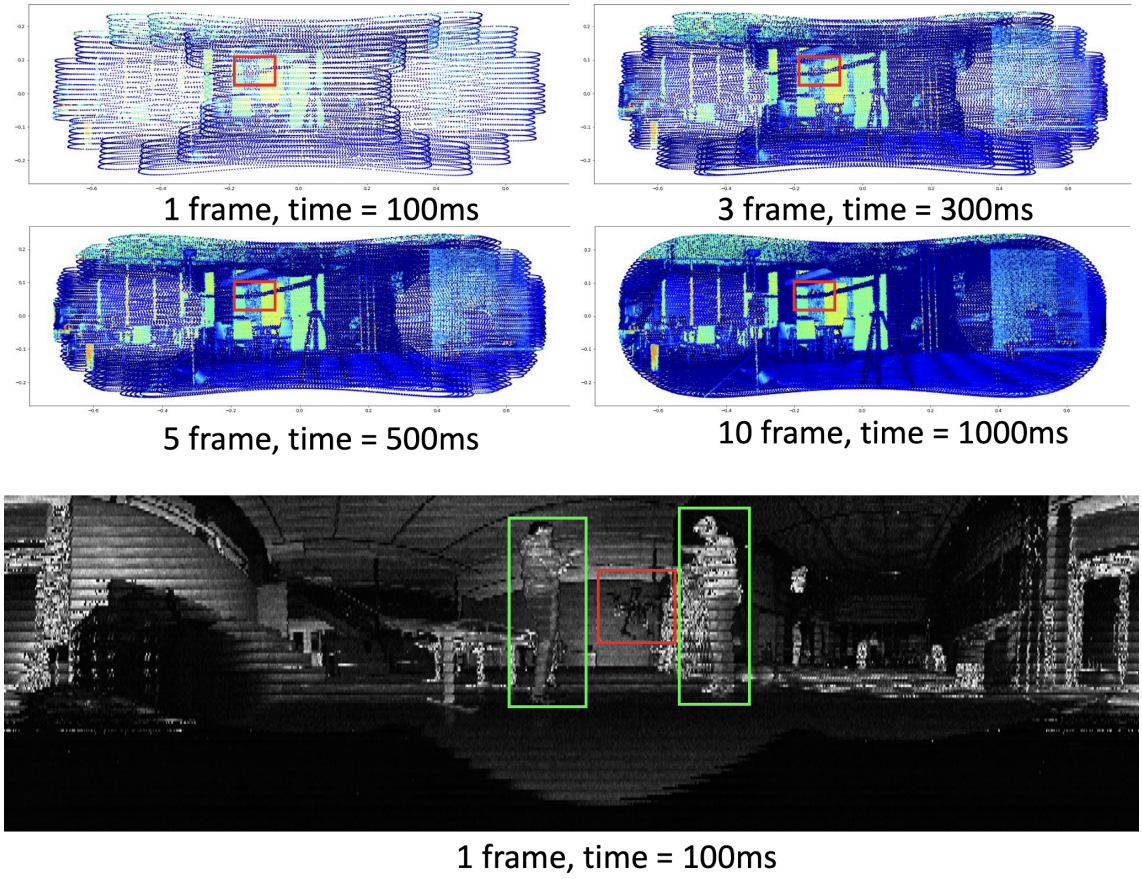


Figure 4.1: Converting the point cloud data from the Ouster OS0-128 and Livox Horizon to 2D signal images.

equipped with an Intel i5-9300HF processor, 16 GB DDR4 RAM, and 1 TB SSD storage, as depicted in Fig. 4.3. This setup provides the necessary computing power and data storage for processing the incoming data stream from the LiDAR in real-time.

4.1.2 Software Design

Before delving into the detailed description of the individual software modules upon which the localization algorithm is based, it is useful to provide an overview of the entire system. The basic requirements of the algorithm are to take raw point cloud data output by the Ouster LiDAR (with the 128-line laser radar being the preferred option) as input and calculate the output, namely the x, y,and z spatial coordinates of the unmanned aerial

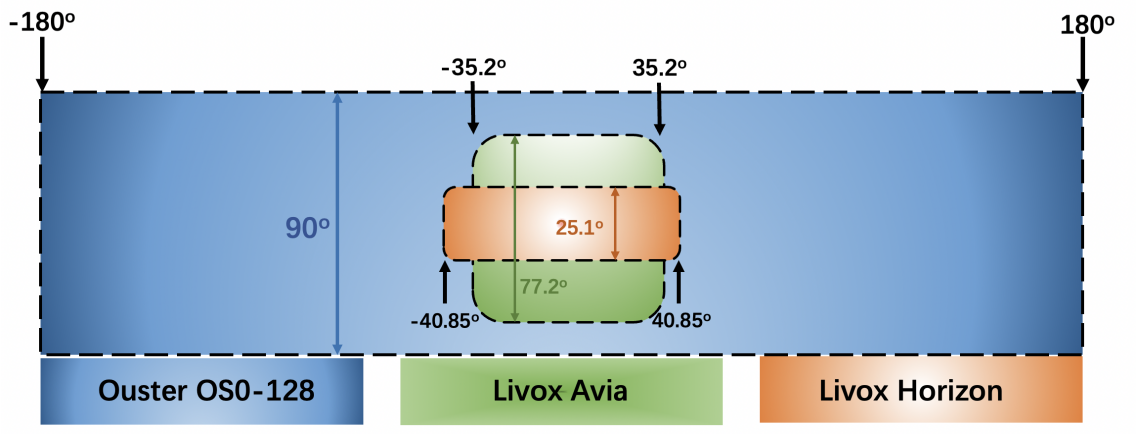


Figure 4.2: FoV of Livox Horizon, Livox Avia and Ouster OS0-128.

vehicle (UAV), recorded in the format of ROS messages as *geometry_msgs/PoseStamped*.

Fig. 4.4 provides a general understanding of the system and its inputs (raw point cloud data from the Ouster LiDAR) and outputs (the x, y, z spatial coordinates of the UAV). The core of the design is the effective fusion of the signal images and point cloud data output by the Ouster LiDAR. The raw point cloud data of the Ouster LiDAR placed at a certain height is first transformed to obtain the corresponding 2D signal image, and the image target recognition algorithm is run to obtain the target frame, which is then mapped to the point cloud data to obtain the point cloud data of the interested area. If the UAV is far away from the Ouster LiDAR, the image recognition algorithm is unable to find the corresponding target frame, it will find the target point cloud from the area near the spatial coordinates of the UAV in the previous frame. The middle position information is obtained from the point cloud data of the UAV. The effective fusion of signal images and point cloud data has effectively solved the processing speed and localization precision of point cloud data computation.

4.1.3 Object Detection with YOLOV5

The default model of YOLOv5s is trained on the COCO dataset, which primarily contains RGB images (e.g. 640x640) that are completely different from signal images

Figure 4.3: Setup for LiDAR-based drone tracking.

(e.g. 512/1024/2048x128). As a result, the model's performance on signal images is not as good as typical RGB images. Therefore, we will attempt to re-train the model using some custom data to improve its performance. To re-train the YOLOv5 model using LiDAR data, we need to construct our own dataset. In our design, we collected multiple OS0-128 PCAP files and utilized the Ouster Python SDK to extract the reflectance information of each frame scan. We then utilized OpenCV's *cv2.imwrite* function to save the reflectance data as grayscale images for data labeling. As shown in Fig. 4.5, our selected images depict drones of various sizes and angles. To improve network learning and enhance the generalization ability of our network models, we employed data augmentation techniques. Specifically, we utilized image rotation, image flipping, and brightness balancing as our data augmentation methods . Rotating and flipping images improves the detection performance and robustness of the network, while brightness balancing eliminates the impact of brightness deviation caused by environmental lighting changes and sensor differences on network performance [96]. After data enhancement, we obtained

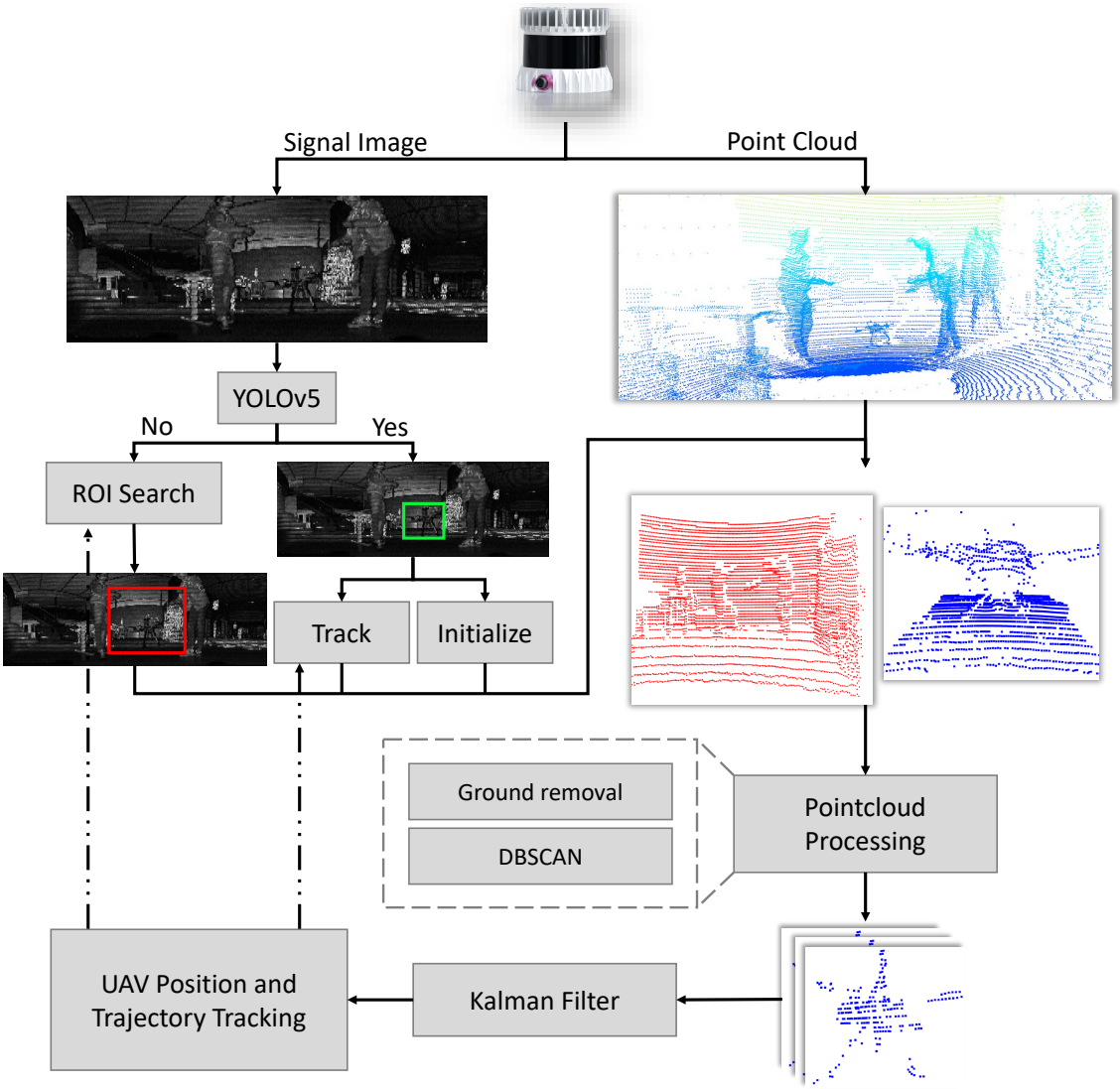


Figure 4.4: Flowchart of the Ouster LiDAR based drone tracking system.

a total of 920 images, which were divided into a training dataset, a validation dataset, and a test dataset in the ratio of 7:2:1. The results of our final training are illustrated in Fig. 4.6. Our method is based on the signal image of Ouster LiDAR and YOLOv5, and has achieved a high accuracy rate. When the epoch is 45, the box loss is 0.02, and the object loss is about 0.01, as shown in Fig. 4.6. The speed of our method is 30 frames per second (FPS), allowing for accurate detection of drones in the signal images of Ouster LiDAR. Our results indicate that the resolution of the training images plays a significant

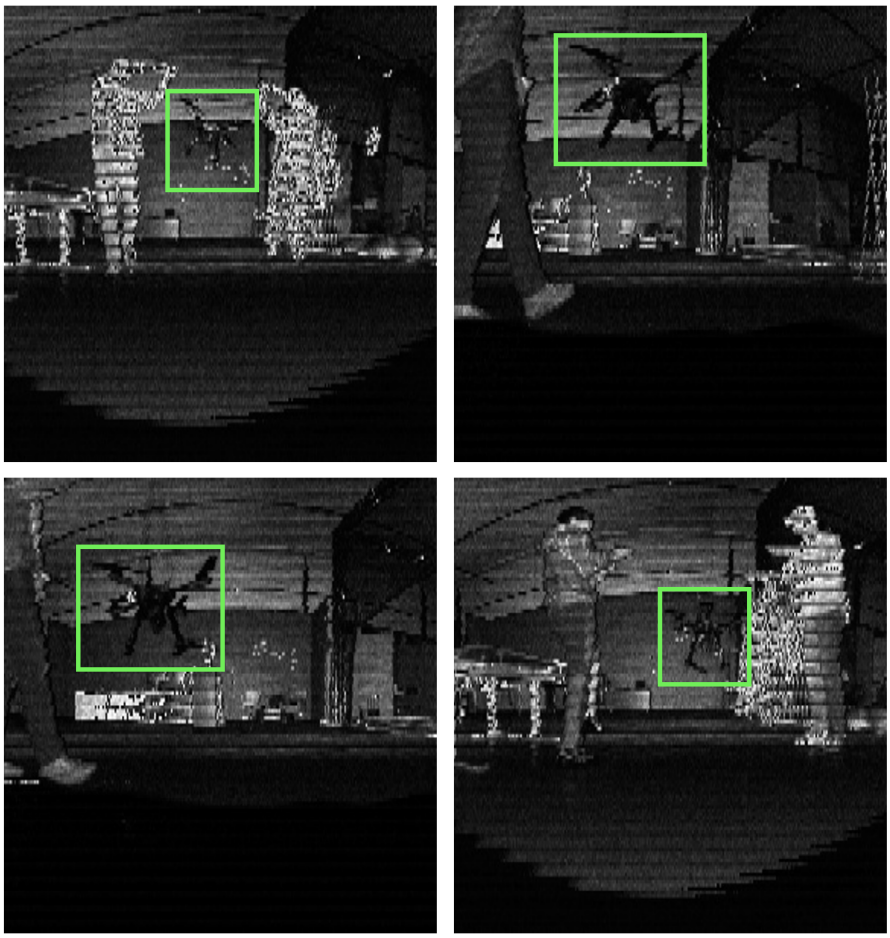


Figure 4.5: Traing data for YOLOV5 model for datecting UAV.

role in determining detection accuracy. We found that as the resolution of the input image increases, the detection accuracy also increases. Furthermore, the fusion strategy that we employed was an important factor in achieving these results.

4.1.4 Point Cloud Precessing

One of the key features of Ouster LiDAR is its 1:1 spatial correspondence of data points. This means that each pixel in the 2D structured data corresponds to a 3D point in the LiDAR data, without the need for any discretization or resampling. As a result, the perception process is not affected by unnecessary noise or artifacts, which helps to improve accuracy and significantly reduce computational requirements. This is why 2D

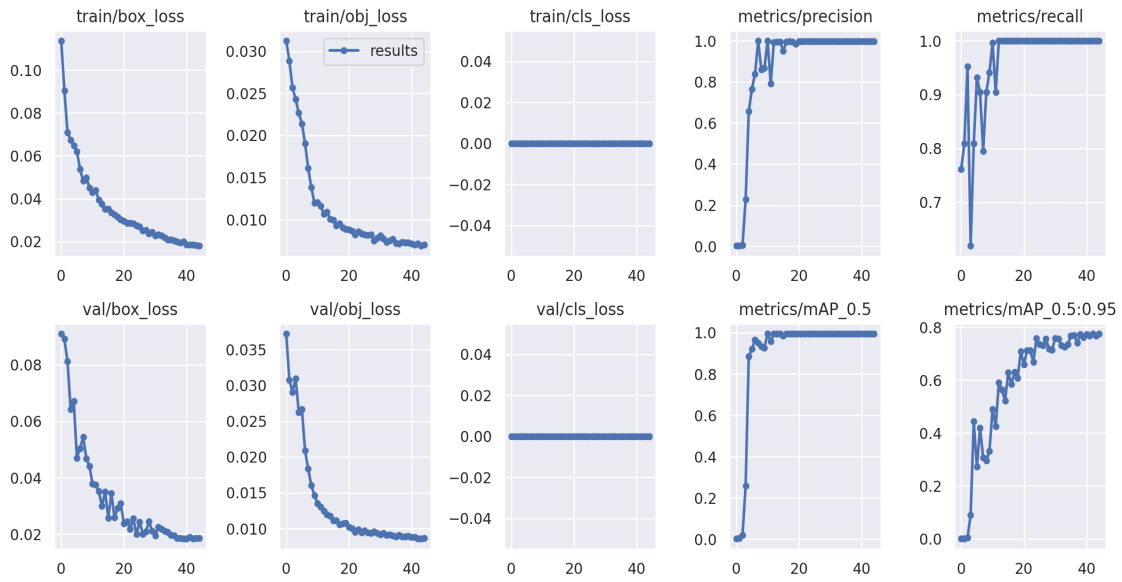


Figure 4.6: Plots of box loss, objectness loss over the training epochs for the training set.

algorithms can directly use the 3D data output by the Ouster LiDAR.

In the previous section, we obtained the target frame of the Ouster LiDAR signal image, and subsequently mapped the 2D target frame to the 3D point cloud. With the assistance of the Ouster SDK, we were able to easily complete this step. Our unmanned aerial vehicle (UAV) was able to successfully extract the region of interest from the point cloud at different distances, as shown in the first two columns of Fig. 4.7. We observed that the number of point clouds in the region of interest was 2904, 714, and 6160 points at near, medium, and long distances, respectively. This is in obvious contrast to the number of point clouds in the full range, which is 261440. The small number of point clouds also lays the foundation for the next step.

In the context of this project, it is imperative to adopt a dependable method to segregate the point cloud representing the drone from the environmental context. Due to the drone's physical attributes, including size and structural elements, the point cloud data derived from it can vary with the distance from the LiDAR sensor. Particularly at substantial distances, the 3D structure of the drone's point cloud data exhibits a wide range

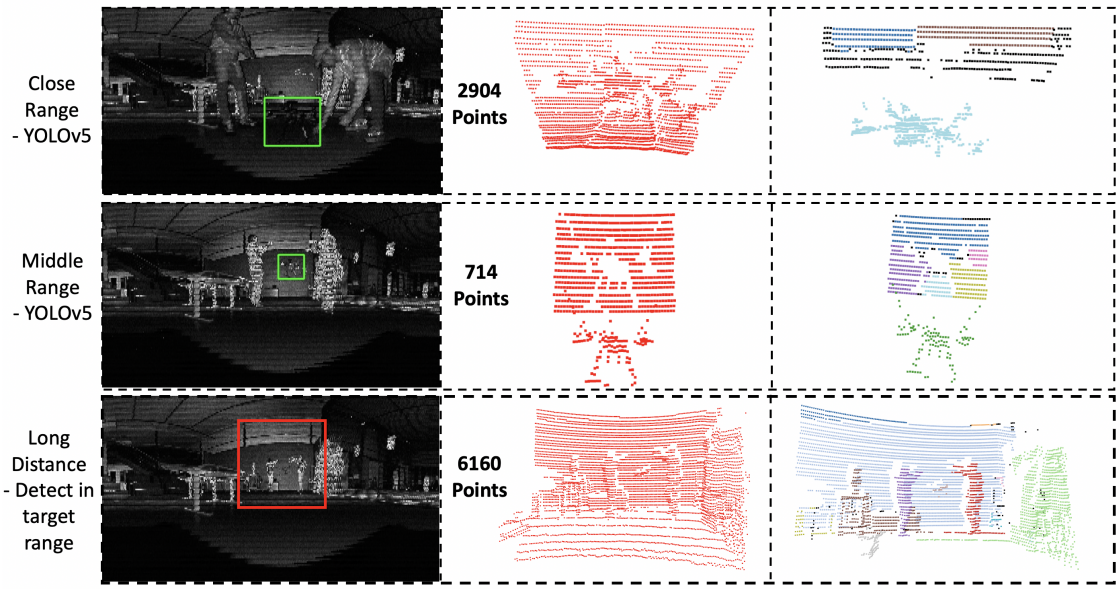


Figure 4.7: The first column shows the signal image of Ouster LiDAR, the top and middle are the output of YOLOV5 detection, and the bottom is the exploration range outside the detection range of YOLOV5. The second and third columns correspond to the original point cloud and clustered point cloud of the region of interest.

of variations, as represented in Fig. 4.8.

One could contemplate the use of model-based clustering for this project, integrating a high-capacity model, for instance, a deep neural network with an extensive number of layers. However, this approach is not deemed practical due to its time-intensive training requirements, imposing a significant burden on users to develop their unique models tailored to the task. The region-growing method, as referenced in [80], while having its advantages, necessitates manual interaction for the procurement of seed points. Moreover, it is susceptible to over-segmentation and noise sensitivity, rendering it less suitable for this project. This limitation becomes more pronounced when considering point cloud data derived from long-range drones, which is susceptible to being misinterpreted as noise. Given these considerations, the DBSCAN algorithm, known for its proficiency in distinguishing high-density clusters from low-density clusters, was selected for this project. The decision to adopt the DBSCAN algorithm was guided by its demonstrated

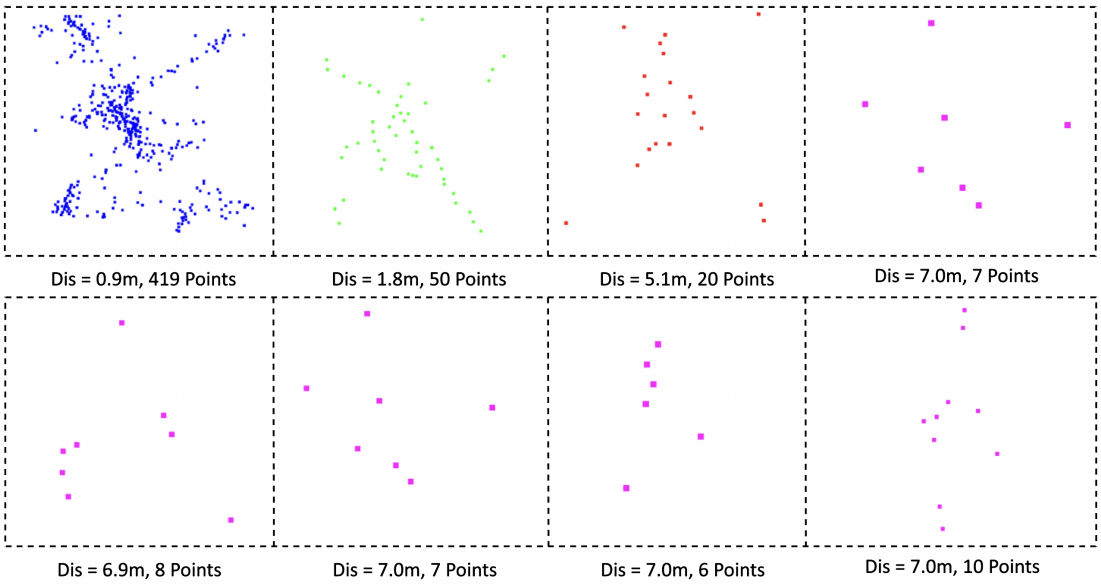


Figure 4.8: The point cloud data of drones at different distances, the bottom line shows the point cloud data of 4 consecutive frames of drones at long distances.

capacity to proficiently handle the inherent variability and noise within the point cloud data of the drone, as evidenced in Fig. 4.9 and Fig. 4.8. Given this ability, it is proposed that DBSCAN is expected to yield optimal results within the parameters of this particular project. It is pivotal to highlight the critical role of the appropriate ϵ value (distance threshold) selection when deploying the DBSCAN algorithm, as noted in the study by *E. Schubert* [82] et al. Through empirical testing, it was determined that an ϵ value of 0.3, within an indoor environment, was effective in segregating the drone's point cloud data. This outcome aligns with the findings of *E. Schubert* et al. [82]. Furthermore, it was observed that at $\epsilon = 0.3$, the best point cloud clustering results were achieved when the neighborhood sample number threshold (MinPts) was in a linear relationship of 1:1.2 with the number of points representing the drone. This correlation is illustrated in the third column of Fig. 4.7.

After clustering the point clouds of the region of interest, we need to distinguish the point clouds of the drone from the clustered clusters. For point cloud tracking, the judgment of the first frame is crucial, unless the algorithm provides sample point clouds

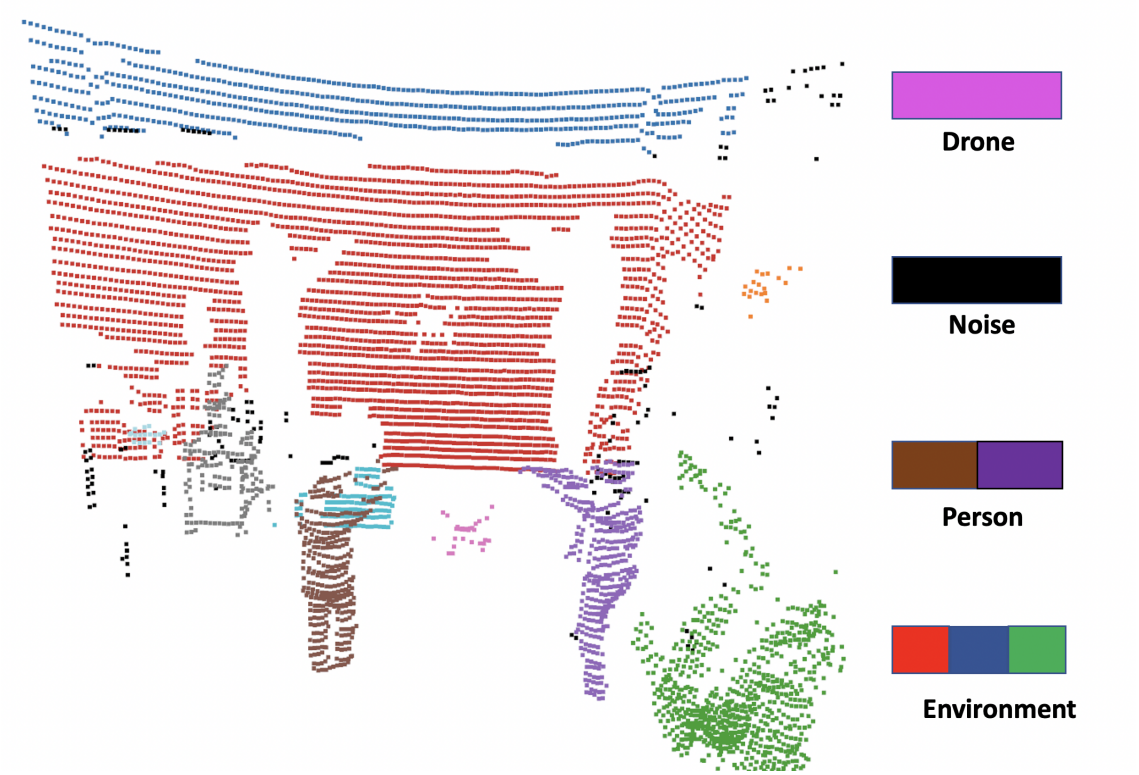


Figure 4.9: Segment the point cloud data from the drone from the environment with DBSCAN.

as a comparison object. In this design, we use the first target box determined by YOLOv5 as the starting point, and use the distance of the center point of the cluster from the Ouster LiDAR as a criterion, as outlined in Algorithm 2, $\mathcal{P}_{roi}$ represents the point cloud of the region of interest, while $\mathcal{P}_{drone}$ represents the point cloud of the final output drone. The cache data, $\mathcal{D}_{cache}$, stores the distance information of different clusters from the Ouster LiDAR. The first step in the process involves the removal of ground points from the point cloud of the input region of interest. This is followed by the application of the DBSACN point cloud clustering operation on the output result. Subsequently, the cluster closest to the Ouster LiDAR is selected as the point cloud of the drone for the final output. The resulting effect is illustrated in Fig 4.10.

Algorithm 3 is designed to extract the UAV point cloud when the UAV is within the detection range of YOLOv5. The algorithm uses three inputs: the raw point cloud of the current frame, $\mathcal{P}^t_{raw}$, the signal image of the current frame, $\mathcal{S}^t$, and the UAV point cloud

Algorithm 2: Get UAV initial position.

Input:

ROI pointcloud: $\mathcal{P}_{roi}$

Output:

Drone pointcloud: $\mathcal{P}_{drone}$

Point cloud clusters label : $\mathcal{L} \leftarrow DBSCAN(\mathcal{P}_{roi})$;

Distance cache of point cloud clusters: $\mathcal{D}_{cache} = []$;

$\mathcal{P}'_{roi}$ = RemoveGround($\mathcal{P}_{drone}$)

for $\mathcal{P}_i$ *in* $\mathcal{P}'_{roi}(\mathcal{L})$ **do**

    $\mathcal{D}_i$ = Distance ($\mathcal{P}_i$, (0,0,0))

    $\mathcal{D}_{cache}$.append($\mathcal{D}_i$)

Drone point cloud index : idx = $\mathcal{L}$ ($\mathcal{D}_{cache}$ = Min ($\mathcal{D}_{cache}$))

$\mathcal{P}_{drone}$ = $\mathcal{P}'_{roi}$(idx)

of the previous frame, $\mathcal{P}_{auv}^{t-1}$. The steps involved in Algorithm 3 are as follows:

1. Step 1: Perform YOLOv5 object detection for each $\mathcal{S}^t$.

2. Step 2: Perform object extraction on the point cloud within the ROI range. The ROI range is defined based on the output of the YOLOv5 object detection.

3. Step 3: Use the number and distance of the UAV point cloud in the previous frame of object extraction as a reference. This helps to confirm that the extracted points correspond to the UAV.

4. Step 4: If the object extraction fails, the result predicted by KF shall prevail. This means that the Kalman filter (KF) output is used to estimate the position of the UAV.

Algorithm 3: UAV tracking in middle range

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` ($\mathcal{P}_{raw}^{t}$, $\mathcal{P}_{uav}^{t-1}$, $\mathcal{ROI}_{yolo}^{t}$) **:**

- $\mathcal{P}_{roi}^{t}$ = $\mathcal{P}_{raw}^{t}$ ($\mathcal{ROI}_{yolo}^{t}$)
- $\mathcal{P}^{t} \longleftarrow ground\_removal(\mathrm{P}_{roi}^{t})$
- $\mathcal{P}_{i}^{t} \longleftarrow DBSCAN(\mathrm{P}^{t})$, 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 ;
      - $\mathcal{P}_{uav}^{t} \longleftarrow$ P;
    - **else**
      - flag = 0 ;
  - **else**
    - flag = 0 ;
- **return** $\mathcal{P}_{uav}^{t}$, flag;

foreach *new* $\mathcal{S}^{t}$ **do**

- $\mathcal{ROI}_{yolo}^{t} \longleftarrow YOLOv5(\mathrm{S}^{t})$
- $\mathcal{P}_{uav}^{t}$, 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* ($\mathbf{P}_{uav}^{t}$) ;

By using YOLOv5 object detection and object extraction, the algorithm can accurately extract the UAV point cloud when it is within range. Additionally, using the previous frame's UAV point cloud as a reference helps to improve the accuracy of the extraction

process. If the object extraction fails, the algorithm relies on the KF output to estimate the position of the UAV, ensuring that the algorithm always provides a result.The resulting effect is illustrated in Fig 4.10.

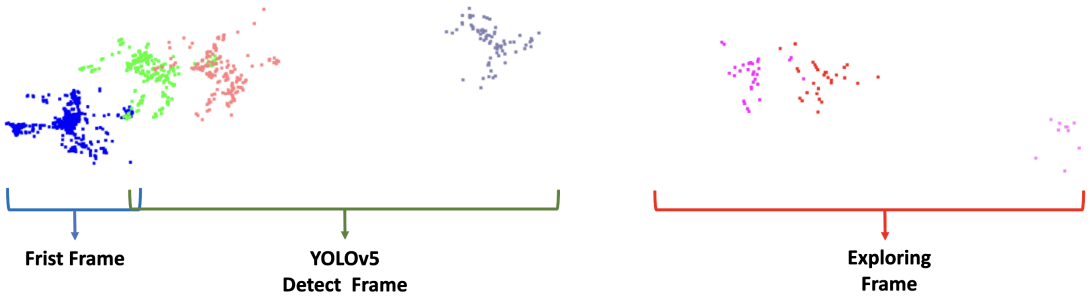


Figure 4.10: UAV point cloud cluster separated from the environment.

When the UAV exceeds the object detection range, the number of point clouds of the UAV is small, and the structure is large, making it difficult to detect the UAV accurately using YOLOv5. In such scenarios, we need to expand the ROI by referring to the coordinates of the UAV in the previous frame to detect the UAV accurately. This process is illustrated in the third row of the first column in Fig. 4.7.

The input to the Algorithm 4 consists of point cloud data from the current frame, denoted as the raw point cloud of the current frame, $\mathcal{P}_{raw}^{t}$, the signal image of the current frame, $\mathcal{S}^{t}$, and the UAV point cloud of the previous frame, $\mathcal{P}_{auv}^{t-1}$. Using the center point of $\mathcal{P}_{auv}^{t-1}$ as a reference, the ROI of the current frame is determined, allowing the specific ROI of the point cloud to be identified, and $\mathcal{P}^{t}$ obtained. To reduce the computational burden, the ground is removed from $\mathcal{P}^{t}$, and clusters with a similar number of points to $\mathcal{P}_{auv}^{t-1}$ are filtered out to obtain a new cluster. Then, the point cloud closest to the target point cloud is selected as the point cloud of the UAV. This process helps to identify the UAV point cloud accurately while reducing the computational complexity of the algorithm. The final effect of this process is shown in Fig. 4.10. Overall, expanding the ROI based on the previous frame's coordinates and filtering out similar clusters in the point cloud of the current frame provides an effective way to detect the UAV accurately when it is beyond

the detection range of YOLOv5.

Algorithm 4: UAV tracking with searching ROI

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(`$\mathcal{P}_{raw}^{t}$, $\mathcal{P}_{uav}^{t-1}$`)`**:**

- $\mathcal{ROI}^{t} \longleftarrow KF\left(get\_center\left(\mathcal{P}_{uav}^{t-1}\right)\right)$
- $\mathcal{P}^{t} \longleftarrow ground\_removal\left(\mathcal{P}_{ROI}^{t}\right)$
- $\mathcal{P}_{i}^{t} \longleftarrow DBSCAN\left(\mathcal{P}^{t}\right),\ i \in (0, R)$
- **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**

- $\mathcal{P}_{uav}^{t}$, 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* ($\mathbf{P}_{uav}^{t}$) ;

4.2 Implementation

This section describes in detail the software and hardware settings and experimental steps of the UAV tracking experiment.

4.2.1 UAVs Used for Experiments

During the experiment, a drone model called Holybro X500 V2 (refer to Figure 4.1) was used. It is a versatile quadcopter frame. Compared to other quadcopters such as DJI S1000 and DJI Matrice 300, Holybro X500 V2 has a smaller size, suitable for indoor safe flight, and smaller weight and projection area under Ouster LiDAR, making it a better choice to test this design and its robustness. Table 4.2 shows the technical specifications of the three drones Holybro X500 V2, DJI S1000, and DJI Matrice 300.

Table 4.2: Comparison of technical specifications of different UAVs.

Technical specifications	Holybro X500 V2	DJI S1000	DJI Matrice 300
Maximum takeoff weight	2.5 kg	9.0 kg	11.0 kg
Flight Time	18 min	55 min	15 min
Radio Control Range	300 m	8 km	300 m
Battery	4S LiPo	12S LiPo	6S LiPo

Assembling the Holybro model requires us to install the general hardware architecture of the drone and various components (flight controller, telemetry radio, RC receiver, GNSS sensor, power supply, etc.). In addition, we must overcome the inherent challenges of complex hardware design, such as installing different cameras, choosing the best location for the onboard computer and sensors, and ensuring that the newly assembled drone is ready to fly. In this experiment, we assembled Holybro according to the official recom-

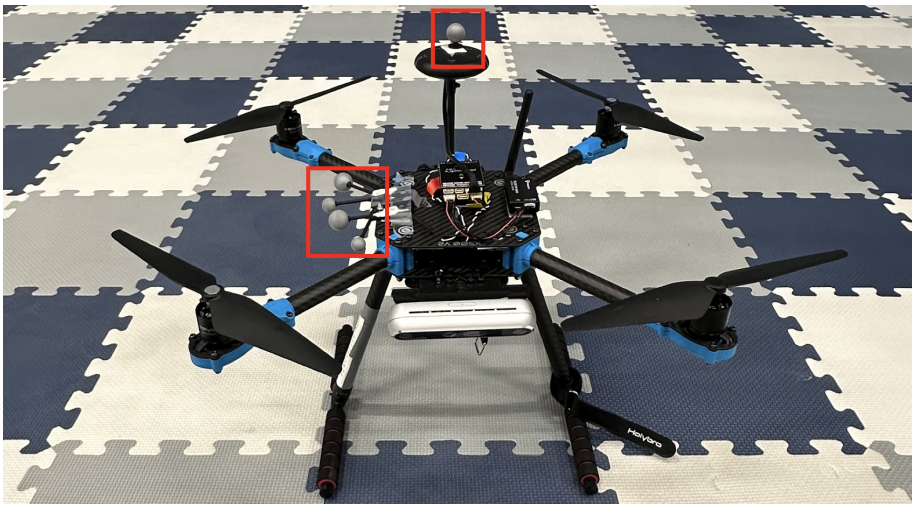


Figure 4.11: Holybro X500 V2 quadcopter equipped with onboard computer, the object identified by the red frame is Optitrack Marker.

mended configuration and carried Realsense T265 LiDAR camera and OptiTrack Marker, as shown in Fig. 4.11.

4.2.2 Collect Ground Truth with MoCAP System

In order to validate and evaluate the effectiveness of our design, we need to obtain ground truth and compare it with the results of the experiment. In this experiment, we chose to use the OptiTrack MoCAP system to obtain ground truth. The OptiTrack system typically produces measurement errors of less than 0.2 millimeters, and in smaller measurement areas, the OptiTrack system typically produces errors of 0.1 millimeters or less. Each motion capture camera emits infrared light. The infrared light is reflected by the markers on the UAV (refer to Fig. 4.11) and is sent back as a two-dimensional image to each camera (refer to Fig. 4.12). Fig. 4.12 shows the OptiTrack MoCAP system as well as the motion capture camera from the perspective of a brother angle. Finally, the OptiTrack markers will be published by the OptiTrack MoCAP system's operating software Motive [1] in the ROS message format *PoseStamped*.

[1]https://github.com/ros-drivers/MoCAP_optitrack

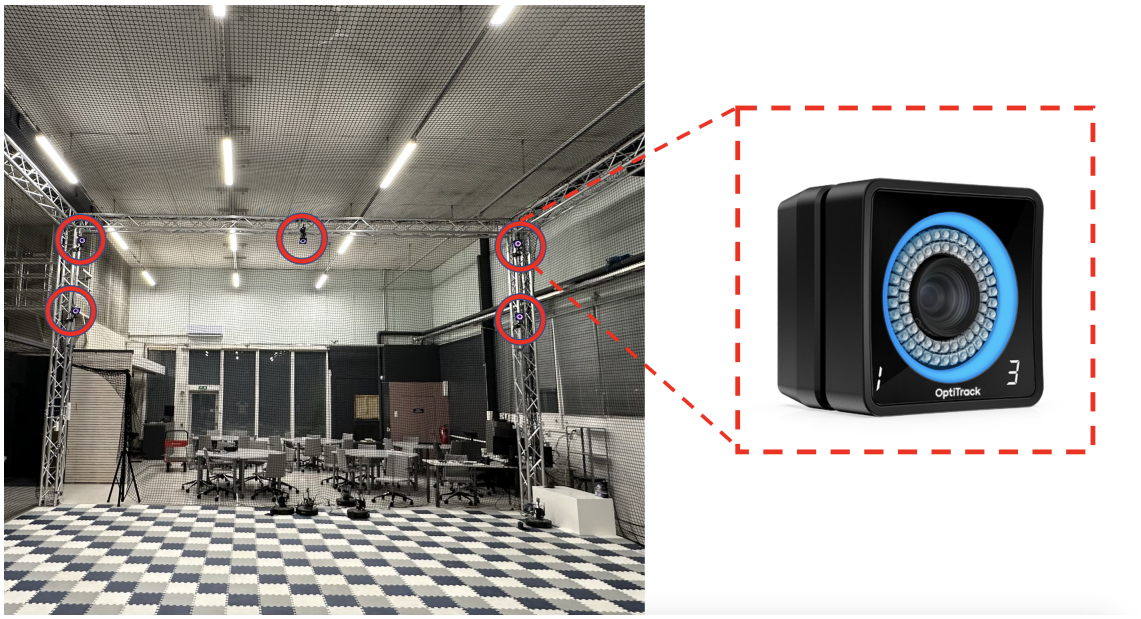


Figure 4.12: OptiTrack MoCAP system (left) and OptiTrack Prime camera (right).

While collecting ground truth data, it is necessary to concurrently execute our drone tracking program. This allows for the simultaneous acquisition of two sets of data, with corresponding timestamps, which can then be saved in the Rosbag format for further analysis in chapter 4.3.

We collected three sequences, denoted as *Sequence 1*, *Sequence 2*, and *Sequence 3*, all of which were captured in the environment shown in Fig. 4.12. Details of the three sequences are provided in Table 4.3, where the trajectory of the UAV in *Sequence 1* is a relatively smooth elliptical path, and the average flight speed of the UAV is relatively slow. The farthest distance of the UAV from the Ouster OS0-128 LiDAR is 7.0 meters. The trajectory of the UAV in *Sequence 2* is a spiral ascent, and the average flight speed of the UAV is relatively fast. The farthest distance of the UAV from the LiDAR is 6.3 meters. *Sequence 3* has a longer duration than *Sequence 2*, and the average flight speed of the UAV is slower than that in Sequence 2. However, the farthest distance of the UAV in *Sequence 3* is greater than that in *Sequence 2*, reaching 8.0 meters.

Table 4.3: Details of sequences that use for our experiment.

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

4.2.3 Setup for Ground Truth Evaluation and Run-time 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 perfor-

mance, the Lenovo Legion Y7000P equipped with 16GB RAM, 6-core Intel i5-9300H (2.40GHz) 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 (1.43GHz), 4GB RAM, and 128-core Maxwell GP

4.3 Results

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

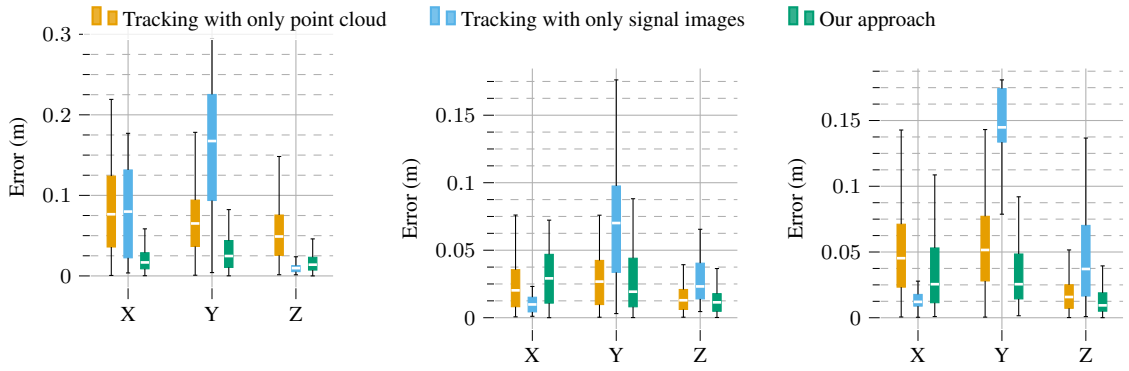
4.3.1 UAV in the Ouster LiDAR Point Cloud

The first parameter to analyze is the number of points that reflect from the UAV at different distances. Our analysis, shown in Fig. 4.8, 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.

4.3.2 Trajectory Validation

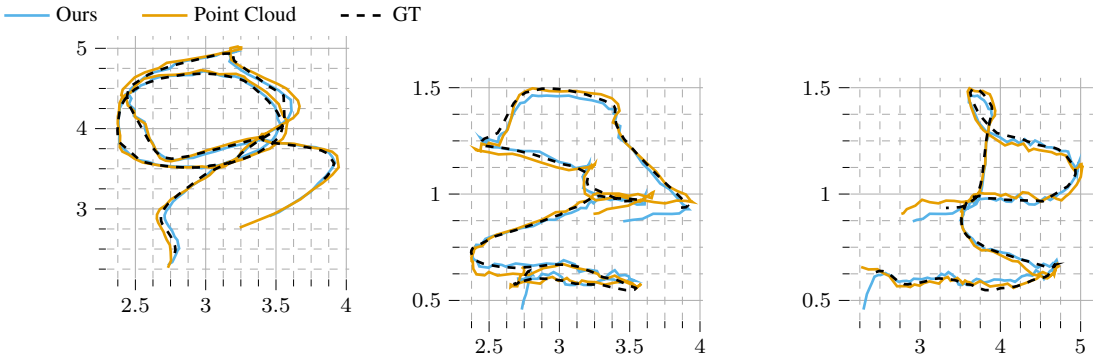
We also show a quantitative analysis of the APE based on ground truth, with the main results summarized in Fig. 4.13. To ensure the trajectories are compared under the same coordinates, we utilize the coordinates of the ground truth as the reference coordinates and

convert all trajectories generated by the three UAV tracking methods to these coordinates.



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

Figure 4.13: Absolute position error (APE) value of three data sequences.



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

Figure 4.14: Comparison of estimated trajectories with the point cloud tracking method and our proposed method from three different projections.

Table 4.4 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.

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

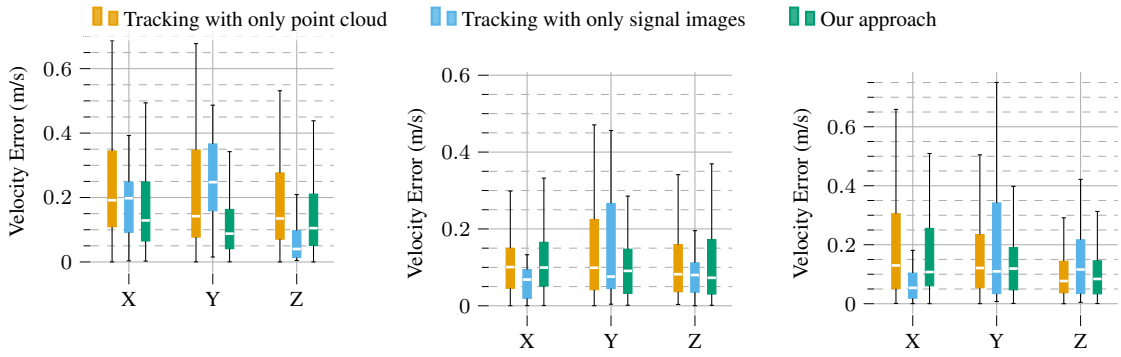
Table 4.4: Performance(Detectable distance, frame rate, and APE error) and initial conditions comparison of selected tracking methods.

	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 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 Fig. 4.14, with more consistent behavior.

4.3.3 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. Fig. 4.15 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 4.15: Velocity estimation error for each linear component in the three data sequences.

4.3.4 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 [2]. 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 4.5.

Table 4.5: 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.

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

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 Conclusion and Future Works

5.1 Conclusion

We provide LiDAR datasets covering the characteristics of various environments (indoor, outdoor, forest), and systematically evaluate 5 open source SLAM algorithms in terms of LiDAR Odometry, and power consumption. The experiments have covered 9 sequences across 2 computing platforms. By including the Nvidia Jetson Xavier platform, it provides further references for the application of various SLAM algorithms on computationally resource-constrained devices such as drones. Overall, we found that in both indoor and outdoor environments, the spinning LiDAR-based FLIO exhibited good performance with low power consumption, which we believe is due to the ability of the spinning LiDAR to obtain a full view of the environment . However, in the forest environment, the LIOL algorithm based on solid-state LiDAR has the best performance in terms of accuracy and mapping quality, although it has the highest power consumption due to the sliding window optimization.

Additionally, we present 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.

5.2 Future Works

Finally, we aim to extend our dataset further to provide more refined and challenging sequences and open-source it in the future. In this thesis, our benchmark tests only focus on SLAM algorithms based on spinning LiDAR and solid-state LiDAR. In the future, we will add benchmark tests based on cameras and even SLAM algorithms based on multiple sensor fusions.

With regards to our LiDAR-as-camera based drone tracking method, we aim to explore its potential application to multiple drone tracking in the future. Another direction for improvement is to integrate the LiDAR-based tracking into the navigation of the UAV and incorporate the onboard state estimation of the UAV into the tracking algorithm, thus expanding the scope of application for this design.

References

[1] Q. Li, J. P. Queralta, T. N. Gia, Z. Zou, and T. Westerlund, "Multi-sensor fusion for navigation and mapping in autonomous vehicles: Accurate localization in urban environments", *Unmanned Systems*, vol. 8, no. 03, pp. 229–237, 2020.

[2] N. Varney, V. K. Asari, and Q. 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*, 2020, pp. 186–187.

[3] J. Yang, Z. Kang, S. Cheng, Z. Yang, and P. H. 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*, vol. 13, pp. 1055–1067, 2020.

[4] D. Van Nam and K. Gon-Woo, "Solid-state lidar based-slam: A concise review and application", in *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, IEEE, 2021, pp. 302–305.

[5] L. Qingqing, Y. Xianjia, J. P. Queralta, and T. Westerlund, "Adaptive lidar scan frame integration: Tracking known mavs in 3d point clouds", in *2021 20th International Conference on Advanced Robotics (ICAR)*, IEEE, 2021, pp. 1079–1086.

[6] K. Li, M. Li, and U. D. Hanebeck, "Towards high-performance solid-state-lidar-inertial odometry and mapping", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5167–5174, 2021.

[7] J. P. Queralta, L. Qingqing, F. Schiano, and T. Westerlund, "Vio-uwb-based collaborative localization and dense scene reconstruction within heterogeneous multi-robot systems", *arXiv preprint arXiv:2011.00830*, 2020.

[8] J. Lin and F. 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)*, 2020, pp. 3126–3131. DOI: 10.1109/ICRA40945.2020.9197440.

[9] Q. Li, X. Yu, J. P. Queralta, and T. Westerlund, "Multi-modal lidar dataset for benchmarking general-purpose localization and mapping algorithms", *arXiv preprint arXiv:2203.03454*, 2022.

[10] Y. Xianjia, S. Salimpour, J. P. Queralta, and T. Westerlund, "Analyzing general-purpose deep-learning detection and segmentation models with images from a lidar as a camera sensor", *arXiv preprint arXiv:2203.04064*, 2022.

[11] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. 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)*, IEEE, 2020, pp. 4353–4360.

[12] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments", *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 642–657, 2019.

[13] M. Kaess, A. Ranganathan, and F. Dellaert, "Isam: Incremental smoothing and mapping", *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[14] M. Fallon, H. Johannsson, M. Kaess, and J. J. Leonard, "The mit stata center dataset", *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1695–1699, 2013.

[15] X. Weng and K. Kitani, "A baseline for 3d multi-object tracking", *arXiv preprint arXiv:1907.03961*, vol. 1, no. 2, p. 6, 2019.

[16] Y. Wang, K. Kitani, and X. Weng, "Joint object detection and multi-object tracking with graph neural networks", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 13 708–13 715.

[17] S. Giancola, J. Zarzar, and B. Ghanem, "Leveraging shape completion for 3d siamese tracking", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1359–1368.

[18] Y. Zhang, H. Niu, Y. Guo, and W. He, "3d single-object tracking with spatial-temporal data association", in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 264–269.

[19] L. LiDAR, *Livox horizon manual*, https://www.livoxtech.com/3296f540ecf545assets/, ,Last accessed on 2022-12-12, 2020.

[20] P. Zhang, *Xpeng announces its lidar supplier as livox, a company incubated by drone maker dji*, https://cnevpost.com/2021/01/01/xpeng-announces-its-lidar-supplier-as-livox/, 2021.

[21] J. Hecht, "Lidar for self-driving cars", *Optics and Photonics News*, vol. 29, no. 1, pp. 26–33, 2018.

[22] R. Thakur, "Scanning lidar in advanced driver assistance systems and beyond: Building a road map for next-generation lidar technology", *IEEE Consumer Electronics Magazine*, vol. 5, no. 3, pp. 48–54, 2016. DOI: 10.1109/MCE.2016.2556878.

[23] O. Inc., *Os1 mid-range high resolution imaging lidar, ouster os-1 data sheet*, https://data.ouster.io/downloads/datasheets/datasheet-revd-v2p1-os1.pdf, Last accessed on 2022-11-09, 2021.

[24] O. Inc., *Ouster-sdk*, `https://ouster.com/developers/ouster-sdk/`, Last accessed on 2022-10-16, 2022.

[25] D. Rozenberszki and A. L. Majdik, "Lol: Lidar-only odometry and localization in 3d point cloud maps", in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 4379–4385.

[26] W. Zhen, S. Zeng, and S. Soberer, "Robust localization and localizability estimation with a rotating laser scanner", in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 6240–6245.

[27] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 3144–3150.

[28] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.", in *Robotics: Science and Systems*, vol. 2, 2014.

[29] T. Shan and B. 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)*, IEEE, 2018, pp. 4758–4765.

[30] Q. Li, P. Nevalainen, J. Peña Queralta, J. Heikkonen, and T. Westerlund, "Localization in unstructured environments: Towards autonomous robots in forests with delaunay triangulation", *Remote Sensing*, vol. 12, no. 11, p. 1870, 2020.

[31] P. Nevalainen, P. Movahedi, J. P. Queralta, T. Westerlund, and J. Heikkonen, "Long-term autonomy in forest environment using self-corrective slam", in *New Developments and Environmental Applications of Drones*, Springer, 2022, pp. 83–107.

[32] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

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

[34] J. L. *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)*, IEEE, 2020, pp. 3126–3131.

[35] J. Lin and F. 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)*, IEEE, 2022, pp. 10 672–10 678.

[36] T.-M. Nguyen, M. Cao, S. Yuan, Y. Lyu, T. H. Nguyen, and L. Xie, "Viral-fusion: A visual-inertial-ranging-lidar sensor fusion approach", *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 958–977, 2021.

[37] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset", *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[38] A. S. Huang, M. Antone, E. Olson, *et al.*, "A high-rate, heterogeneous data set from the darpa urban challenge", *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1595–1601, 2010.

[39] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset", *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[40] M. Wigness, S. Eum, J. G. Rogers, D. Han, and H. Kwon, "A rugd dataset for autonomous navigation and visual perception in unstructured outdoor environments", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 5000–5007.

[41] H. Caesar, V. Bankiti, A. H. Lang, *et al.*, "Nuscenes: A multimodal dataset for autonomous driving", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition(CVPR)*, 2020, pp. 11 621–11 631.

[42] P. Xiao, Z. Shao, S. Hao, *et al.*, "Pandaset: Advanced sensor suite dataset for autonomous driving", in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2021, pp. 3095–3101.

[43] J. Yin, A. Li, T. Li, W. Yu, and D. Zou, "M2dgr: A multi-sensor and multi-scenario slam dataset for ground robots", *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2266–2273, 2021.

[44] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of michigan north campus long-term vision and lidar dataset", *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2016.

[45] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking", in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, 2016, pp. 445–461.

[46] C. Martínez, P. Campoy, I. Mondragón, and M. A. Olivares-Méndez, "Trinocular ground system to control uavs", in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3361–3367. DOI: 10.1109/IROS.2009.5354489.

[47] M. Vrba and M. Saska, "Marker-less micro aerial vehicle detection and localization using convolutional neural networks", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2459–2466, 2020. DOI: 10.1109/LRA.2020.2972819.

[48] W. Kong, D. Zhang, X. Wang, Z. Xian, and J. 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, 2013, pp. 2963–2970. DOI: 10.1109/IROS.2013.6696776.

[49] P. Li, D. Wang, L. Wang, and H. Lu, "Deep visual tracking: Review and experimental comparison", *Pattern Recognition*, vol. 76, pp. 323–338, 2018.

[50] A. Carrio, S. Vemprala, A. Ripoll, S. Saripalli, and P. Campoy, "Drone detection using depth maps", in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2018, pp. 1034–1037.

[51] S. Dogru and L. Marques, "Drone detection using sparse lidar measurements", *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3062–3069, 2022. DOI: 10.1109/LRA.2022.3145498.

[52] J. Razlaw, J. Quenzel, and S. Behnke, "Detection and tracking of small objects in sparse 3d laser range data", *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2967–2973, 2019.

[53] M. Hammer, M. Hebel, B. Borgmann, M. Laurenzis, and M. Arens, "Potential of lidar sensors for the detection of UAVs", in *Laser Radar Technology and Applications XXIII*, M. D. Turner and G. W. Kamerman, Eds., International Society for Optics and Photonics, vol. 10636, SPIE, 2018, p. 1 063 605. DOI: 10.1117/12.2303949. [Online]. Available: https://doi.org/10.1117/12.2303949.

[54] M. Hammer, M. Hebel, M. Laurenzis, and M. 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*, G. S. Buller, R. C. Hollins, R. A. Lamb, and M. Mueller, Eds., International Society for Optics and Photonics, vol. 10799, SPIE, 2018, 107990S. DOI: 10.1117/12.2325702. [Online]. Available: https://doi.org/10.1117/12.2325702.

[55] I. Guvenc, F. Koohifar, S. Singh, M. L. Sichitiu, and D. Matolak, "Detection, tracking, and interdiction for amateur drones", *IEEE Communications Magazine*, vol. 56, no. 4, pp. 75–81, 2018.

[56] S.Hengy *et al.*, "Multimodal uav detection: Study of various intrusion scenarios", in *Electro-Optical Remote Sensing XI*, International Society for Optics and Photonics, vol. 10434, 2017, 104340P.

[57] J. Peña Queralta *et al.*, "Autosos: Towards multi-uav systems supporting maritime search and rescue with lightweight ai and edge computing", *arXiv preprint arXiv:2005.03409*, 2020.

[58] T. Rouček, M. Pecka, P. Čıéžek, *et al.*, "Darpa subterranean challenge: Multi-robotic exploration of underground environments", in *International Conference on Modelling and Simulation for Autonomous Systesm*, Springer, 2019, pp. 274–290.

[59] M. Petrlıék, T. Báča, D. Heřt, M. Vrba, T. Krajnıék, and M. Saska, "A robust uav system for operations in a constrained environment", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2169–2176, 2020.

[60] L. Qingqing, J. P. Queralta, T. N. Gia, and T. 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*, 2019, pp. 22–26.

[61] W. Kong, D. Zhang, X. Wang, Z. Xian, and J. 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*, 2013, pp. 2963–2970. DOI: 10.1109/IROS.2013.6696776.

[62] Y. Gui, P. Guo, H. Zhang, *et al.*, "Airborne vision-based navigation method for uav accuracy landing using infrared lamps", *Journal of Intelligent & Robotic Systems*, vol. 72, Nov. 2013. DOI: 10.1007/s10846-013-9819-5.

[63] E. J. Markvicka, J. M. Rogers, and C. Majidi, "Wireless electronic skin with integrated pressure and optical proximity sensing", in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 8882–8888. DOI: 10.1109/IROS45743.2020.9340787.

[64] E. N. Mortensen, H. Deng, and L. Shapiro, "A sift descriptor with global context", in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, IEEE, vol. 1, 2005, pp. 184–190.

[65] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features", in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, Springer, 2006, pp. 404–417.

[66] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Ieee, vol. 1, 2005, pp. 886–893.

[67] R. Girshick, "Fast r-cnn", in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[68] L. O. Chua and T. Roska, "The cnn paradigm", *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 3, pp. 147–156, 1993.

[69] W. Lan, J. Dang, Y. Wang, and S. Wang, "Pedestrian detection based on yolo network model", in *2018 IEEE international conference on mechatronics and automation (ICMA)*, IEEE, 2018, pp. 1547–1551.

[70] W. Liu, D. Anguelov, D. Erhan, *et al.*, "Ssd: Single shot multibox detector", in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, 2016, pp. 21–37.

[71] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints", in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 734–750.

[72] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection", in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6569–6578.

[73] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.

[74] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[75] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 918–927.

[76] Z. Qin, J. Wang, and Y. Lu, "Monogrnet: A geometric reasoning network for monocular 3d object localization", in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8851–8858.

[77] C. Yan and E. Salman, "Mono3d: Open source cell library for monolithic 3-d integrated circuits", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 3, pp. 1075–1085, 2017.

[78] ultralytics, *Yolov5*, `https://github.com/ultralytics/yolov5`, ,Last accessed on 2022-12-02, 2020.

[79] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.", in *kdd*, vol. 96, 1996, pp. 226–231.

[80] A.-V. Vo, L. Truong-Hong, D. F. Laefer, and M. Bertolotto, "Octree-based region growing for point cloud segmentation", *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 104, pp. 88–100, 2015.

[81] C. Chen, G. Li, R. Xu, T. Chen, M. Wang, and L. Lin, "Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4994–5002.

[82] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: Why and how you should (still) use dbscan", *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[83] M.-S. Yang, C.-Y. Lai, and C.-Y. Lin, "A robust em clustering algorithm for gaussian mixture models", *Pattern Recognition*, vol. 45, no. 11, pp. 3950–3961, 2012.

[84] Y. Zhang, M. Li, S. Wang, *et al.*, "Gaussian mixture model clustering with incomplete data", *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 1s, pp. 1–14, 2021.

[85] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp.", in *Robotics: science and systems*, Seattle, WA, vol. 2, 2009, p. 435.

[86] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[87] Z. Yan, L. Sun, T. Krajnıék, and Y. Ruichek, "Eu long-term dataset with multiple sensors for autonomous driving", in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 10 697–10 704.

[88] M. Lixia, A. Benigni, A. Flammini, C. Muscas, F. Ponci, and A. Monti, "A software-only ptp synchronization for power system state estimation with pmus", *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 5, pp. 1476–1485, 2012.

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

[90] J. Lin and F. 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)*, IEEE, 2020, pp. 3126–3131.

[91] K. Li, M. Li, and U. D. Hanebeck, "Towards high-performance solid-state-lidar-inertial odometry and mapping", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5167–5174, 2021.

[92] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

[93] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems", in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, IEEE, 2012, pp. 573–580.

[94] H. Sier, L. Qingqing, Y. Xianjia, J. P. Queralta, Z. Zou, and T. Westerlund, "A benchmark for multi-modal lidar slam with ground truth in gnss-denied environments", *arXiv preprint arXiv:2210.00812*, 2022.

[95] Y. Zhu, C. Zheng, C. Yuan, X. Huang, and X. Hong, "Camvox: A low-cost and accurate lidar-assisted visual slam system", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5049–5055. DOI: 10.1109/ICRA48506.2021.9561149.

[96] J. Ma, Y. Li, Y. Chen, *et al.*, "Estimating above ground biomass of winter wheat at early growth stages using digital images and deep convolutional neural network", *European Journal of Agronomy*, vol. 103, pp. 117–129, 2019.

[97] K. Marneweck *et al.*, "Why data-over-sound is an integral part of any iot engineer's toolbox: Chirp + arm = frictionless low power connectivity", 2019.

[98] qorvo, *Dwm1001*, https://www.qorvo.com/products/p/DWM1001-DEV, [Online] - Last access: 2022-11-23.

[99] G. Welch, G. Bishop, *et al.*, "An introduction to the kalman filter", 1995.

[100] Y. Wang, C. Wang, H. Zhang, Y. Dong, and S. Wei, "Automatic ship detection based on retinanet using multi-resolution gaofen-3 imagery", *Remote Sensing*, vol. 11, no. 5, p. 531, 2019.

[101] G. Pandey, J. R. McBride, and R. M. Eustice, "Ford campus vision and lidar data set", *The International Journal of Robotics Research*, vol. 30, no. 13, pp. 1543–1552, 2011.

[102] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration", in *European conference on computer vision*, Springer, 2016, pp. 766–782.

[103] O. Elmakis, T. Shaked, and A. Degani, "Vision-based uav-ugv collaboration for autonomous construction site preparation", *IEEE Access*, vol. 10, pp. 51 209–51 220, 2022. DOI: 10.1109/ACCESS.2022.3170408.