LiDAR Perception Systems for Autonomous Robots: from Calibration to Localization, Mapping, and Recognition

by

Jianhao JIAO

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in the Department of Electronic and Computer Engineering

December 2021, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

Jianhao JIAO

December 2021

LiDAR Perception Systems for Autonomous Robots: from Calibration to Localization, Mapping, and Recognition

by

Jianhao JIAO

This is to certify that I have examined the above PhD thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

Prof. Ming LIU, Thesis Supervisor

Prof. Albert WONG, Acting Head of Department

Thesis Examination Committee

1. Prof. Ming LIU (Supervisor)	Department of Electronic and Computer Engineering
2. Prof. Michael Yu WANG	Department of Electronic and Computer Engineering
3. Prof. Zhiyong FAN	Department of Electronic and Computer Engineering
4. Prof. Qiong LUO	Department of Conputer Science and Engineering
5. Prof. Hesheng WANG (External Examiner)	Department of Automation Shanghai Jiao Tong University

Department of Electronic and Computer Engineering
December 2021

Acknowledgments

This thesis would never have been completed without help from many people. First of all, I would like to thank my supervisor, Prof. Ming Liu, for his constructive guidance and inspiring encouragement throughout my Ph.D. study. He taught me how to find, analyze, and solve a problem, present research results, and build the proper research taste. I would keep his invaluable teaching in mind and learn from his spirit in both teaching and research. I will try to be a good supervisor like him for the rest of my life.

I want to thank Prof. Xiaofang Zhou, for being the chairperson of my thesis committee. Also, I would like to thank all other thesis committee members: Prof. Michael Yu Wang, Prof. Zhiyong Fan, Prof. Qiong Luo, and Prof. Hesheng Wang for providing advice and taking the time to join my defense committee.

I would thank colleagues from different university departments (i.e., ECE, OKT, CMO, CSO) for their support on the autonomous vehicle trial. I want to thank my colleagues from RAM-LAB, HKUST Robotics Institute, and UDI for their help and sharing in both technical thoughts and happiness in daily life. Especially, I thank Peng and Yang for the nights we worked together to overcome difficulties and modify the paper quality. I thank Prof. Lujia Wang for her kind suggestions for my research and life. I also thank Prof. Rui Fan, for his patient guidance during my first-year PhD. and guiding me to finish my first ICRA paper. I want to thank Dr. Haoyang Ye, Dr. Qinghai Liao, and Dr. Lei Tai for their patient guidance and cooperation on computer vision and robotic research.

Last but not least, I thank my father Mr. Chengquan Jiao, mother Mrs. Yuxing Li, and brother Mr. Jianhua Jiao, and his wife Mrs. Yanxia Liang, for their wise counsel and sympathetic ear. They always listen to me and support my choices. Finally, I thank my girlfriend, Miss Lu Fan, for her love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LiDAR Perception Systems for Autonomous Robots: from Calibration to Localization, Mapping, and Recognition

by Jianhao JIAO

Department of Electronic and Computer Engineering

The Hong Kong University of Science and Technology

Abstract

LiDARs have attached much attention from both academics and industry due to their active nature in measuring distance and robustness to lighting changes. The development of LiDAR technology has facilitated the wide applications of robots in complex environments, such as field surveillance, exploration, search-and-rescue, and autonomous driving. To cope with drawbacks of a single LiDAR in data sparsity and limited field of view, combining multiple LiDARs to maximize a robot's perceptual awareness of environments and obtain sufficient measurement is a straightforward but promising solution. But novel algorithms to unlock the potential of multiple LiDARs must be investigated.

In this thesis, I formulate the multi-LiDAR perception problem in a unified way. Three essential problems ranging from extrinsic calibration, simultaneous localization and mapping (SLAM), and recognition are addressed by proposing a coherent and complete perception solution using multiple LiDARs. Extensive simulated and real-world experiments on various robotic platforms have demonstrated the performance of this proposed solution.

I start by presenting an offline method that enables automatic dual-LiDAR extrinsic calibration using one-shot measurements. I further investigate the automatic calibration and propose a hybrid approach that takes advantages of motion features for initialization as well as appearance cues for refinement. Based on the above research, I propose a complete system that enables the flexible and online calibration as well as SLAM with multiple LiDARs. After that, I investigate the algorithm latency problem and propose

a greedy-based method for feature selection to accelerate the SLAM system. Moreover, I turn to study recognition problems and integrate the calibration methodology with learning approaches to propose a multi-LiDAR 3D object detector with the awareness of extrinsic perturbation for autonomous driving. Finally, I summarize this thesis and propose future research opportunities.

CHAPTER 1

INTRODUCTION

1.1 Background

Benefitting from the progress in mechatronics, sensory technology, and artificial intelligence (AI), the evolution of autonomous robots in recent years is surprising. We have seen that many types of robots, including service robots, drones, and self-driving cars, have appeared in people's hourses. Some of them have gradually changed our way of living or working. Quadrupedal robots such as the SpotMini[1] designed by *Boston Dynamics* have been used for surveillance, releasing laborers from repeated and dangerous works. Drones are applied in racing [37], cinematography [72], and rescue [43] tasks, which highly differs from their initial stage [94]. Another interesting example is the rapid development of self-driving cars [3, 9, 56, 89, 116, 173]. Fig. 1.1 presents robots in our daily life.

You may watch the American science fiction film called *WALL-E* [2]. This movie describes a virtual world that in the 29^{th} century, the Earth is destroyed into a garbage-strewn wasteland. Humanity is nowhere to be found and has been evacuated to other planets. Many mobile trash robots with binocular cameras are left to clean the Earth. All of them are highly intelligent and able to conduct repeated garbage cleaning works without any human supervision. One day, WALL-E, the only living robot, is visited by a probe sent by the robot called EVE, with whom he falls in love and pursues across the galaxy. I was impressed by the lovely story and these robots' human-like behavior in this movie. If the future mode of robots is imaged, both WALL-E and EVE should be two vivid examples.

However, there exists a large gap between our current stage and the production of highly autonomous robots. One of the difficulties is mainly the complexity of *sensing* and *understanding* the surrounding environment. As demonstrated with great success in cleaning robots [152], navigation in a 2D indoor environment with a robot equipped with

[1] `https://www.bostondynamics.com/spot`

[2] `https://en.wikipedia.org/wiki/WALL-E`

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

Figure 1.1: Examples of robot including (a) the quadrupedal robot, (b) the drone [203], (c) the mobile robot [118], and (d) the autonomous logistic vehicle [116].

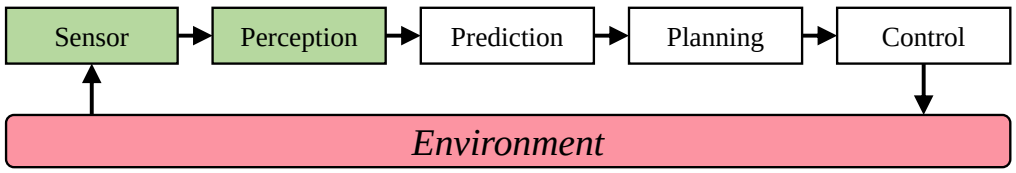
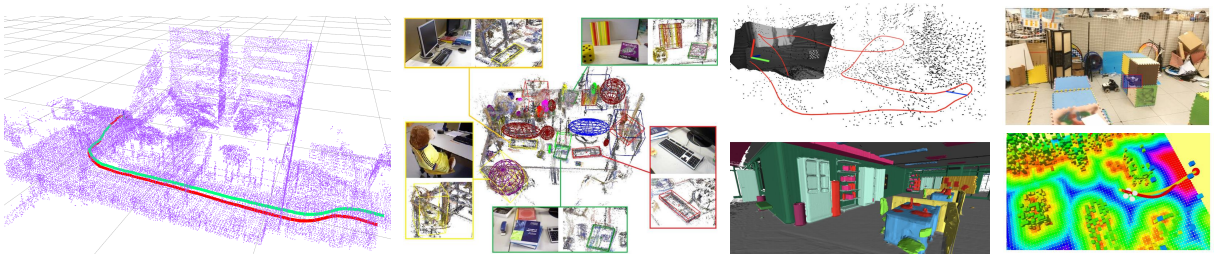


Figure 1.2: Pipeline of a typical robotic system [168]. Modules marked in green (sensor and perception) are focused on in this thesis.

wheel encoders and a laser scanner can be considered largely solved. However, it is still challenging for such robots to work in environments where several dynamic agents such as a human crowd are present. Two factors mainly cause this. First, dynamic environments tend to validate the general assumptions of existing localization and mapping systems which commonly use static landmarks to estimate the motion. Second, robots should be capable of real-time recognizing and predicting agents' motion to avoid collision while following their pre-planned trajectory.

Toward this direction, this thesis focuses on the *sensor* and *perception* problems. My goal is to explore possible solutions to improve the way of a robot to estimate its real-time *states* and construct a *model* (also called *map*) of the environment from sensory data. This naturally intersects with research problems such as sensor calibration, simultaneous localization, mapping (SLAM), and object recognition and tracking. A robot state consists of quantities, such as the pose (position and orientation) that describe the robot's motion over time. A state also includes the robot's velocity, sensor biases, kinematics, and calibration parameters in complex instances. A map is a representation of the environment in which the robot operates. It can be expressed in numerous formats, e.g., landmarks' positions or object bounding boxes, application-dependent. Fig. 1.3 shows some representative results of SLAM. In many high-level navigation tasks, like decision-making [160] and path planning [170], the robot's pose and the environmental model should be known as a priori (see Fig. 1.2).



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

Figure 1.3: Examples of SLAM outputs. Robot's trajectories are shown as red or green lines in (a) and (c). Maps are represented in different formats: (a) an aggregation of 3D LiDAR points [83], (b) object-oriented semantic map with 3D points [187], (c) 3D points (top) and semantic mesh (bottom) [154], and (d) Euclidean Signed Distance Field (ESDF) which is used for path planning (yellow and red lines) [62].

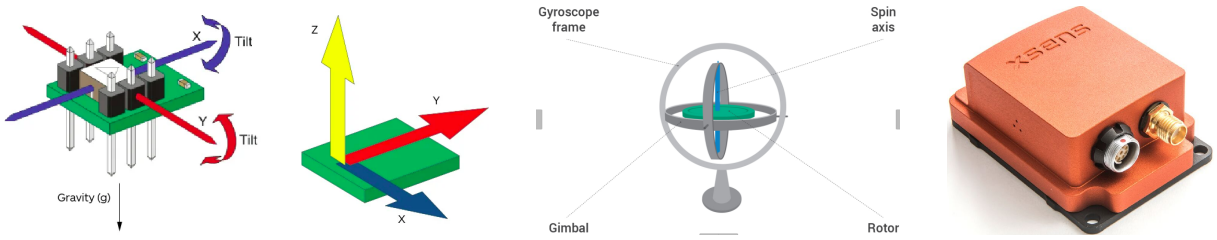
The characteristics of sensors often drive the development of perception algorithms. In other words, the design of an algorithm should take advantage of a specific sensor combination (e.g., multiple cameras/LiDARs, camera+IMU). Therefore, before going into too many details of scientific problems in this thesis, the subsequent Section 1.2 first summarizes the main features and perception algorithms of sensors that are commonly used on robotic platforms and autonomous vehicles. Among these sensors, LiDARs are investigated in this thesis due to their accuracy and robustness in measuring distance. Section 1.3 formulates the LiDAR perception problem and describes my motivation on researching multi-LiDAR perception. Section 1.4 explains how this thesis addresses three typical multi-LiDAR perception problems step by step with different assumptions: extrinsic calibration, SLAM, and object detection. Finally, Section 1.5 states the contributions.

1.2 Sensors for Robotics

Autonomous robots should be equipped with multiple types of sensors for providing diverse and redundant information from the environment and robot itself. It is useful to categorize sensors into two groups: *interoceptive* and *exteroceptive* sensors [6]. Their Definitions are given as follow:

- **Interoceptive**: being stimuli arising within the body.
- **Exteroceptive**: being activated by stimuli received by an organism from outside.

In robotics, typical interoceptive sensors are accelerometers and gyroscopes, and typical exteroceptive sensors are the GNSS, camera, and time-of-flight transmitter/ receiver



(a) (b) (c)

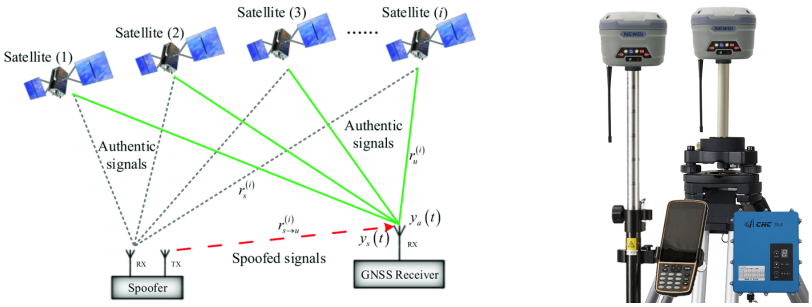
Figure 1.4: (a) An accelerometer. (b) A gyroscope. (c) The Xsens IMU [197].

(e.g., RGB-D camera, Radar, and LiDAR). In the following sections, I briefly review the main features and related works of these sensors.

1.2.1 IMU

An Inertial Measurement Unit (IMU) is an electronic device that is composed of accelerometers, gyroscopes, and sometimes magnetometers to measure a body's acceleration and angular rate (see Fig. 1.4). IMUs are typically used to maneuver aircraft including unmanned aerial vehicles (UAVs) and spacecraft including satellites and landers since they are seldom affected by external environments. Modern microelectromechanical (MEMS) IMUs are both cheap and tiny while keeping with promising performance. They are commonly used on robots.

A major disadvantage of using IMUs is that they typically suffer from accumulated error. The navigation system is continually integrating accelerometer data twice to compute the position, and gyroscope data once to track the orientation. The output high-frequency gyroscope and accelerometer measurements suffer from a Gaussian noise and bias [47]. Any errors, however small, are accumulated, yielding drift over time. As emphasized by Barfoot [6], "In most cases, the best state estimation concepts make use of both interoceptive and exteroceptive measurements." Therefore, IMUs are often fused with cameras [145] and LiDARs [197] in the Extended Kalman Filter (EKF) or optimization framework [6] for propagating high-rate motion prior [47]. Besides pose estimation and localization, IMU-centric sensor systems have been demonstrated with several novel applications, including object tracking [39, 148], semantic mapping [154], and temporal calibration [147, 149].



(a) (b)

Figure 1.5: (a) The concept of trilateration that enables a GPS receiver to calculate its position. (b) A typical RTK-GNSS.

1.2.2 GNSS

The Global Navigation Satellite System (GNSS) refers to a constellation of satellites providing space signals transmitting positioning and timing data to GNSS receivers. The receivers then use this data to determine location. By definition, GNSS provides global coverage. Through the receiver, any device can use the GNSS to locate its global position. Existing GNSS includes the China's BeiDou navigation satellite system (BeiDou), European Union's Galileo positioning system (GALILEO), the USA's NAVSTAR global positioning system (GPS), Russia's global navigation satellite system (GLONASS), and India's NavIC and Japan's Quasi-Zenith regional satellite system (QZSS) [42].

Like the IMU, tiny GNSS devices have been widely embedded into consumer-level devices such as phones and electronic wristbands. But the GNSS has several limitations: *1)* hard to initialize (needs enough statellites to be found); *2)* noisy and inaccurate measurements ($\geq 10cm$); and *3)* low frequency (typically $1-100$Hz). This is because, in practical applications, the GNSS commonly suffers from multiple sources of errors and influences, including the message transmission delay through the atmospheric lay, the reflection of signals on multiple surfaces (e.g., localization among buildings), ephemeris errors, and the uncertainties on the satellite's position [185]. Therefore, the ideal case is to use the GNSS in outdoor open-field environments. In recent years, differential GNSS (DGPS) and real-time kinematic GNSS (RTK-GNSS) were also developed to enhance the GNSS's accuracy and allow for localization within the order of decimeters or even centimeters in well-conditioned environments. Fig. 1.5 illustrates the working principle of GNSS and an example of RTK-GNSS.

Fusing GNSS signals with IMU measurements via the EKF is a popular solution, and

therefore becomes a system called the Inertial Navigation System (INS). The INS has been commonly adopted by many self-driving cars [54]. Additionally, some robotic researchers also explored the potential of the GNSS on camera- or LiDAR-based systems and thus achieved the hybrid indoor-outdoor navigation systems [146, 200]. The global position provided by the GNSS is important for correcting drift of navigation systems.

1.2.3 Camera

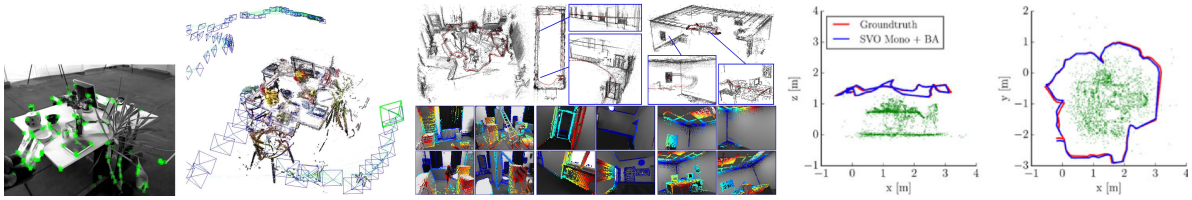
Modern cameras mimic the output of the human visual system. The front-end optics capture light reflected or emitted by an object in the 3D world through the optical center and project it onto the camera's 2D imaging plane. The intensities of the light are encoded and restored as an image, in which each element is called *pixel*. Images are extremely rich in texture, and their ability to provide both spatial and qualitative information has attracted researchers' attention on *computer vision problems* [4, 49, 174].

Interacting with the world requires robots to have the capability of 3D perception. Camera calibration is the first challenge to be solved, whose objective is to estimate the geometric model that describes the camera projection process. Intrinsic calibration estimates the *focal length*, *pixel origin*, and *distortion parameters* of a camera, while extrinsic calibration computes the relative *rotation* and *translation* among multiple cameras. In practical applications, both intrinsics and extrinsics are pre-computed by using the maker-based methods [209] (i.e., move a checkerboard with sufficient motion before cameras), and sometimes online optimized.

The next step is to infer the 3D environmental model given a series of images. This often involves the problem of visual odometry (VO) or visual SLAM (VSLAM). Depending on how to process images, related algorithms are categorized onto feature-based (indirect) methods, direct methods, and hybrid methods. Fig. 1.6 visualizes their results.

1) Typical feature-based SLAM frameworks are the ORB-SLAM series [20,128,129]. They use ORB features [156] with robust descriptors to boost short and mid-term data associations, build a covisibility graph to manage keyframes, and performs loop closure and relocalization with the DBoW2 library [52].

2) Typical direct method is the Direct Sparse Odometry (DSO) [41], which estimates the geometry and camera model by directly operating on the raw sensor measurements. It



(a) (b) (c)

Figure 1.6: Results of classical VSLAM systems: (a) ORB-SLAM [128], (b) DSO [41], and (c) SVO [48].

minimizes the *photometric error* that computes the intensity difference between corresponding pixels, instead of the *reprojection error* in feature-based methods. The direct method skips the keypoint extraction and matching step, which saves a high constant cost per frame. However, it may fail in large viewpoint changes, and optimizing the photometric residual easily gets stuck in a local minimum.

3) The Semidirect VO (SVO) [48] belongs to the hybrid domain, which estimates frame-to-frame motion with the sparse image alignment (direct manner) and refines the geometry and camera poses with the bundle adjustment (indirect manner).

Additionally, data-driven approaches open up a new door for designing novel visual systems in recent years [11,125,158,194]. For example, semantics from images can provide object types and properties, which offer depth prior and strong association cues.

The minimum setup of VSLAM is using a monocular camera, but the absolute scale cannot be estimated. An initialization phase based on physical-based prior is required to establish a map with sufficient points. In practice, since epipolar constraints can acquire the depth information [4], stereo or multi-camera solutions are more popular. Furthermore, RGB-D cameras that exploit the time-of-flight technique to obtain depth offer another option directly, as demonstrated in applications [130, 171].

However, traditional cameras are easily affected in situations of high dynamics, low texture or structure distinctiveness, and challenging illumination conditions (see Fig. 1.7). Novel solutions are needed to these challenges:

1) One direction is to fuse cameras with the IMU, leading to the visual-inertial system that the IMU helps to resolve the high dynamics and textureless issues [20, 103, 145, 154]. Alternatively, cameras can be fused with the LiDAR, in which the absolute depth information eliminates the scale ambiguity problem [198, 206, 220].

(a) (b)

Figure 1.7: Open challenges in tranditional computer vision [159]: (a) latency and motion blur and (b) high dynamic range.

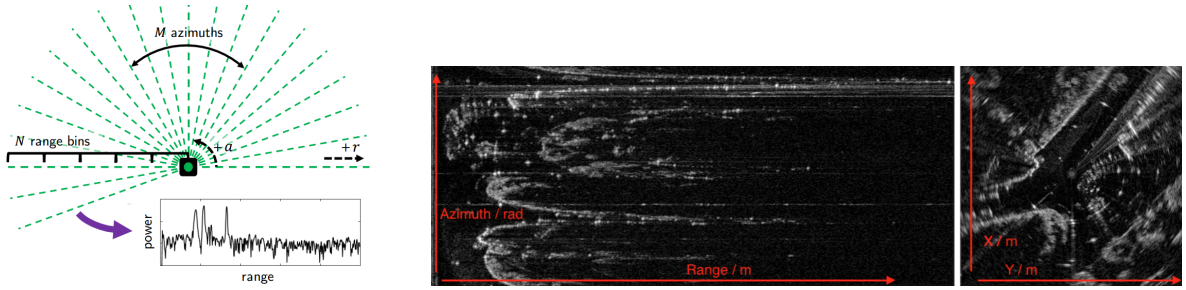
2) Another direction is to develop the bio-inspired event cameras. Different from standard cameras, which capture intensity images at a fixed rate, event cameras asynchronously capture the per-pixel *intensity changes* and output a stream of *events*. Each event is encoded with information, including the triggered time, pixel localization, and the sign of the intensity change. As summarized in [51], event cameras have great potential for several computer vision and robotic tasks, e.g., high-speed motion estimation and high dynamic range perception, which are commonly difficult to frame cameras. However, research on common vision problems with event cameras is preliminary. This is because event cameras work in a fundamentally different way from frame cameras. Existing vision-based solutions still lack a good way to handle events. Novel event-based algorithms must be investigated.

1.2.4 Radar

Radio Detection and Ranging (Radar) sensors emit and receive radio waves to determine objects' distance, angle, and velocity. The important advantage of the Radar is its reliability against extreme conditions such as rain, snow, dust, fog, or direct sunlight. Since the technology is rapidly becoming affordable and efficient, Radars are currently accessible to modern self-driving cars.

I am taking the Frequency-Modulated Continuous-Wave (FMCW) Radar[3] as an example. The FMCW Radar is a special Radar that provides a 360°-view of the scenes with several desirable features: high reliability, high resolution, and long-range. The Radar data can be viewed as a 2D image that can be processed with vision-based techniques, as shown in Fig. 1.8. Research on FMCW Radars has been active in recent years [75].

[3] `https://navtechradar.com/explore/fmcw-radar`



(a) (b)

Figure 1.8: (a) Bird-eye view of a FMCW scanning radar [22]. (b) Example sensor data from the Navtech CTS350-X radar [9].

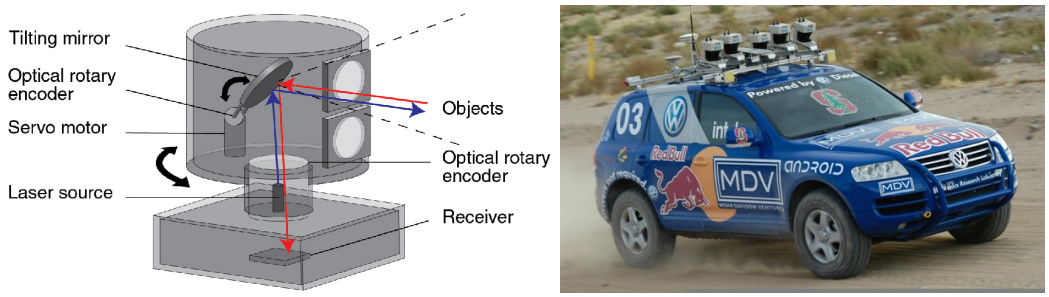
Cen et al. [22, 23] present a Radar-only odometry pipeline with efficient keypoint extraction and graph-based matching. The graph matching approach is robust to false-positive key points. Hong et al. [70] propose the complete graph-based Radar SLAM system: *RadarSLAM* that online manages the map points, detects loop candidates, and optimizes pose graphs to correct drift. Burnett et al. [18] provide solutions to compensate the motion distortion and Doppler effect in radar odometry, while this was often neglected by previous research. Directly aligning two consecutive Radar frames offer another option to solve the odometry problem. An example is demonstrated in [137] which explots the phase correlation. Regarding the object detection and place recognition tasks, FMCW Radars are also feasible, as presented in [16,50,175,184,199]. Available open datasets aush as Oxford Radar RobotCar dataset [9] and MulRan dataset [90] further spur research in this area.

Radars and LiDARs have similar working principles, but each uses different wavelengths of light. The longer wavelength used by Radar does not allow the detection of small objects. It is also challenging to determine the object's category (e.g., pedestrian or cars) from Radars. A complete navigation system should make the complementary strengths of Radars and LiDARs, as shown in [66].

1.2.5 LiDAR

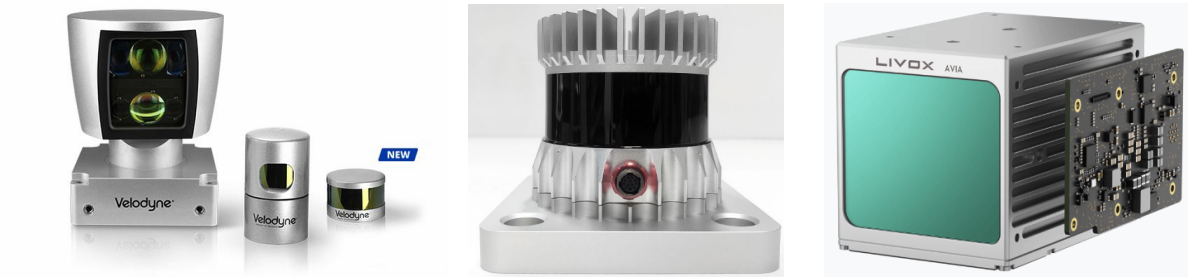
1.2.5.1 Overview of LiDARs

Light Detection and Ranging (LiDAR) sensors emit and receive lasers (an amplified light with a short wavelength) to determine the distance of an object. The intensity of the laser partially reflects the surface materials of the targeted object. This property can



(a) (b)

Figure 1.9: (a) The working principle of a traditional mechanical LiDAR. (b) The Stanley autonomous vehicle is equipped with multiple LiDARs [181].



(a) (b) (c)

Figure 1.10: (a) The Velodyne LiDAR. (b) The Ouster LiDAR. (c) The Livox LiDAR.

be used to recognize special targets such as the road border. Fig. 1.9a illustrates the working principle of LiDARs. Traditional mechanical scanning LiDARs (e.g., Velodyne HDL-64E) commonly align multiple lasers vertically and physically rotate them at a high rate (around 3600°-7200° per second) to obtain a 360° horizontal field of view. This can significantly enhance the perceptual view compared with single-beam LiDARs. Due to their active nature in measuring distance, LiDARs are robust to external light conditions and reliable measurements accuracy. Fig. 1.9a illustrates the LiDAR's working principle.

The application of autonomous driving accelerates the development of LiDAR technology. A pivotal occurrence should be the success of Stanley [181] that is the first autonomous vehicle completed the DARPA Grand Challenge in 2005 (see Fig. 1.9b). Stanley replied on LiDARs in 3D mapping and obstacle detection. This event has motivated researchers and engineers to improve both hardware and algorithms of LiDARs. In the past ten years, the complexity and cost of producing a LiDAR quadratically increased along with the number of used lasers. However, now, it is no longer an issue. More and more companies, such as *Velodyne*, *Ouster*, *Luminar*, *Robosense*, *Livox*, have joined the LiDAR market in recent years (see Fig. 1.10). Some of them resort to the solid-state design

to eliminate the moving mechanical parts. The decreasing weight, size, and price make LiDARs gradually available in robotic platforms. I have seen that most self-driving cars adopt LiDARs as their standard setup. LiDARs are also popular to mobile robots [111] and quadrupedal robots [101].

The output data of a LiDAR is a point cloud $\mathcal{P}$. Each point cloud contains a set of unordered 3D points: $\mathbf{p} = [x, y, z, \mathbf{s}]$ in the Cartesian coordinate system, where $\mathbf{s}$ indicates properties of a point, such as the intensity, ring id, and timestamp. Additionally, the range image [31] is another representation of a point cloud. Here, LiDAR is regarded as a spherical projection model. Each point is projected onto an image to create a valid pixel. The pixel value records the Euclidean distance from a point to the origin.

1.2.5.2 Algorithms in LiDAR-Based SLAM and Object Detection

This subsection briefly introduces typical LiDAR-based SLAM and object detection algorithms, which are two important tasks in LiDAR perception. Fig. 1.11 example results of LiDAR-based SLAM and object detection.

LiDAR-Based SLAM

The iterative closest point (ICP) algorithm is a classical solution to LiDAR-based motion estimation. Generally, many SLAM system are developed from ICP [124,140,161, 162], into which the methods including *measurement preprocessing*, *degeneracy prediction*, and *sensor fusion* are incorporated. These works have pushed the current LiDAR-based systems to become fast, robust, and feasible in large-scale environments.

As the front-end of a system, the measurement preprocessing encodes point clouds into a compact representation. Related algorithms are categorized into either dense or sparse methods. As a typical dense method, Suma [10] demonstrates the advantages of utilizing surfel-based maps for registration and loop detection. The range image is used. Its extended version [31] incorporates semantic constraints into the original cost function. Sparse methods prefer to extract keypoints from raw measurements. LOAM [205] selects distinct points from both edge lines and local planar patches. Its variants have used ground planes [163], semantics [28], probabilistic grid maps [77], good features [84], or dense scanners [15,110,138] to improve the performance in noisy or structure-less environments.

In the motion estimation part, different methods have been proposed to tackle the

(a) (b)

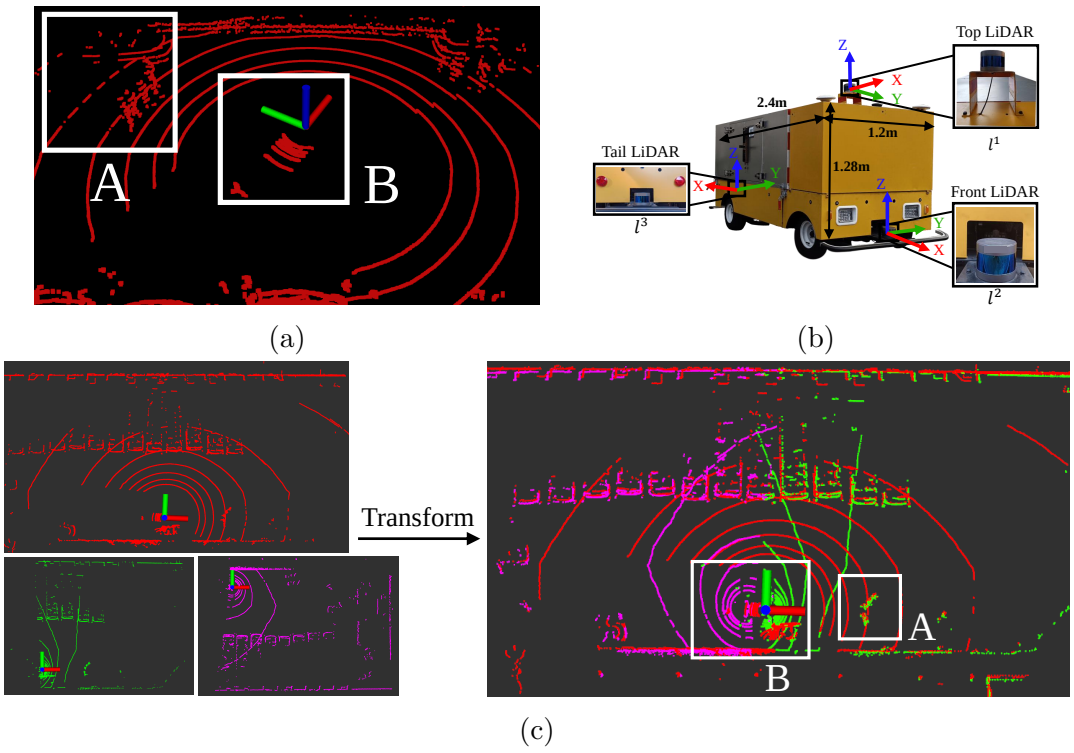
Figure 1.11: Example results of (a) SLAM [164] (b) 3D object detection [217].

degeneracy issue. Zhang et al. [204] define a *factor*, which is equal to the minimum eigenvalue of information matrices, to determine the system degeneracy. They also proposed a technique called *solution remapping* to update variables in well-conditioned directions. This technique was further applied to tasks including localization, registration, pose graph optimization, and inspection of sensor failure [69, 207, 214].

Utilizing multiple sensors to improve the motion-tracking performance of single-LiDAR odometry is promising. Most existing works on LiDAR-based fusion combine visual or inertial measurements. The simplest way to deal with multi-modal measurements is loosely coupled fusion, where each sensor's pose is estimated independently. For example, LiDAR-IMU fusion is usually done by the EKF [122, 183, 211]. Tightly coupled algorithms that optimize sensors' poses by jointly exploiting all measurements have become increasingly prevalent in the community. They are usually done by either the EKF [65,112,113,143,220] or optimization [99, 100, 121, 165, 197, 215].

LiDAR-Based Object Detection

The LiDAR-based object detectors are generally categorized into grid-based [105, 192, 217] and point-based methods [25, 142, 167]. Li et al. proposes 3D-FCN [105] to apply 3D volumetric CNN on voxelized point clouds. Zhou et al. proposes VoxelNet [217], which is an end-to-end network, to learn features. But both of these methods commonly suffer a high computational cost using dense convolution. In contrst, SECOND [192] utilizes the spatially sparse convolution [58] to replace the 3D dense convolution layers. Regarding point-based methods, Qi et al. proposed PointNet [25] to learn features from the raw unordered point set with a symmetric function. Furthermore, their follow-up work presents a set-abstraction block to capture local features [142]. Shi et al. presents PointRCNN [167] which exploits PointNet to learn point-wise features and extract foreground points for autonomous driving.



(a) (b) (c)

Figure 1.12: (a) The drawback of the single-LiDAR setup. Region A: measurement sparsity. Region B: occlusion. (b) The vehicle with a multi-LiDAR system tested in Chap. 3. (c) The bird-eye view of a merged point cloud which is transformed by point clouds from all LiDARs (indicated by different colors). Both the region A and B are compensated with additional points.

These detectors can be enhanced by exploiting visual information. MV3D [29] combines features from multiple views, including a bird-eye view and front view, to conduct classification and regression. AVOD [93] extends MV3D by generating object proposals through the features at all stages. On the other hand, several methods [141, 190] separate the process of detection into two stages. They first projected region proposals generated by the camera-based detectors into 3D space and then used PointNets to segment and classify objects on point clouds.

1.3 Research Problems

LiDARs are robust and provide valuable information for navigation. However, the limitations of LiDARs are also obvious: LiDAR commonly suffers from data sparsity, limited FOV, and occlusion. Single LiDAR-based methods are easily induced to fail in structureless environments. Fig. 1.12a displays a point cloud example captured by the top LiDAR of the autonomous vehicle. In block A, the pedestrians and vehicles are scanned

with a few points, making these objects challenging. About block B, points are gathering together because of occlusion. These issues arise a demand of deploying multiple LiDARs on a robot to maximize the perceptual awareness of environments and obtain sufficient measurements, like the setup in Fig. 1.12b. Compared with a single-LiDAR setup, the primary improvement of a multi-LiDAR system is the significant enhancement of the sensing range and density of measurements, as demonstrated in Fig. 1.12c. However, the multi-LiDAR perception is still an open problem, and few articles discuss it. Therefore, this thesis mainly focuses on SLAM and object detection, two important tasks in LiDAR perception. The extrinsic calibration of multiple LiDARs is also concerned.

In the following sections, I will first formulate perception problem for the single-LiDAR setup. For the multi-LiDAR case, this formulation should be modified with additional unknown variables, where several key challenges are involved.

1.3.1 Problem Formulation for the Single-LiDAR Case

Denoting $\mathcal{X}$ the unknown variable to be estimated given data $\mathcal{Z}$. From the optimization perspective, the optimal $\hat{\mathcal{X}}$ is estimated by minimizing the *objective function*:

$$\hat{\mathcal{X}} = \arg\min_{\mathcal{X}} f(\mathcal{X}, \mathcal{Z}), \tag{1.1}$$

where the specific definitions of $\mathcal{X}$ and $\mathcal{Z}$ are given in below sections.

1.3.1.1 SLAM

In LiDAR-based SLAM (L-SLAM), the variable $\mathcal{X}$ typically indicates the robot's poses, and $\mathcal{Z} = \{\mathbf{z}_k\}$ represents a set of measurements, e.g., raw point clouds or features from LiDARs. Each measurement can be expressed as a function of $\mathcal{X}$, i.e., $\mathbf{z}_k = h_k(\mathcal{X}) + \boldsymbol{\epsilon}_k$, where $h_k(\cdot)$ is a known function (the *observation* model) and $\boldsymbol{\epsilon}_k$ is random measurement noise. SLAM is commonly formulated as a maximum a *posteriori* (MAP) estimation problem [19]. Assume that the measurement noise $\boldsymbol{\epsilon}_k$ is an independent, zero-mean Gaussian noise with the covariance matrix $\boldsymbol{\Sigma}_k$, the objective is defined as

$$f(\mathcal{X}, \mathcal{Z}) \triangleq \sum_{\mathbf{z}_k \in \mathcal{Z}} ||h_k(\mathcal{X}) - \mathbf{z}_k||^2_{\boldsymbol{\Sigma}_k}, \tag{1.2}$$

where $||\mathbf{e}||^2_{\boldsymbol{\Sigma}} = \mathbf{e}^\top \boldsymbol{\Sigma}^{-1} \mathbf{e}$ is the Mahalanobis distance. The minimization of (1.2) is commonly solved via iterative methods sucha as Gauss-Newton or the Levenberg-Marquardt.

These methods locally linearize the objective function by computing the Jacobian as $\mathbf{J} = \partial f / \partial \mathcal{X}$ on the operation point. Given an initial guess, $\mathcal{X}$ is iteratively optimized until convergence.

1.3.1.2 Object Detection

In LiDAR-based object detection tasks [201], the output is a set of objects' bounding boxes $\mathcal{B}$. Each box $\mathbf{b}$ is parameterized as $[cls, x, y, z, l, , w, h, \gamma]$, where cls is the class (e.g., car, pedestrian) of a bounding box, $[x, y, z]$ denotes a box's bottom center, $[l, w, h]$ represent the sizes along the $x-$, $y-$, and $z-$ axes respectively, as well as γ for the rotation of the 3D bounding box along the $z-$ axis. Object detection is a typical pattern recognition problem, which can be solved by *supervised learning* algorithms. To make the notation compatible with the computer vision community, I substitute $\mathcal{X}$ and $\mathcal{Z}$ with $\boldsymbol{\theta}$ and $\mathcal{D}$ respectively.

In learing-based methods, *training* and *inference* are two essential stages. The training process is to teach a model $G_{\boldsymbol{\theta}}(\cdot)$ to learn from the *training set* $\mathcal{D}_{train}$. This is done by minimizing the objective function $f(\boldsymbol{\theta}, \mathcal{Z}_{train})$, where $\boldsymbol{\theta}$ is the model parameter. After getting the parameter $\hat{\boldsymbol{\theta}}$ that best fits the training set, the model can make predictions of data from the *testing set* $\mathcal{D}_{test}$, i.e., $\mathcal{B}_{pred} = G_{\hat{\boldsymbol{\theta}}}(\mathcal{D}_{test})$. Note that $\mathcal{D}_{train}$ contains labeled (or called ground-truth) boxes with corresponding object point clouds, while $\mathcal{D}_{test}$ only consists of raw point clouds.

1.3.2 Problem Formulation for the Multi-LiDAR Case

The major aspect of formulating the multi-LiDAR perception problem is to consider the *extrinsics* that indicate the spatial offset between the reference LiDAR and the target LiDAR. With these extrinsics, point clouds from all LiDARs can be correctly transformed into a unified coordinate system and jointly used to improve the perception results. But extrinsics are often unknown and should be calibrated (called *extrinsic calibration*). Therefore, problem (1.1) is reformulated for the multi-LiDAR case as

$$\hat{\mathcal{X}}, \hat{\mathcal{Y}} = \underset{\mathcal{X},\ \mathcal{Y}}{\arg\min}\ f(\mathcal{X}, \mathcal{Y}, \mathcal{Z}), \tag{1.3}$$

where $\mathcal{Y} = \{\mathbf{y}_i : i = 1, \ldots, I\}$ indicate a set of extrinsics of a multi-LiDAR system and I is the number of LiDARs.

1.3.3 Challenges in Multi-LiDAR Perception

Solving the problem (1.3) is challenging since several issues have to be considered:

1.3.3.1 Extrinsic Calibration

1) *Accurate, Flexible, and Automatic Calibration*: To make use of multi-LiDAR measurements for a new robotic platform, recovering the extrinsics is the first priority. However, previous multi-LiDAR calibration methods present two major drawbacks: *1)* they rely exclusively on an additional sensor and *2)* their success depends on the quality of initialization provided by users. The practical issue that LiDARs share limited overlapping FOV should also be dealt with. The requirement of developing an accurate, automatic, and flexible extrinsic calibration method is put forward.

2) *Modeling the Extrinsic Perturbation*: The extrinsic perturbation always exists due to factors such as vibration, temperature drift, and calibration error. Especially, wide baseline stereo cameras or vehicle-mounted multi-LiDAR systems suffer even more extrinsic deviations than the normal ones. The small extrinsic perturbation is detrimental to the measurement accuracy even with a small change and thus exists as a systematical error. It should be modeled for subsequent SLAM and object detection modules.

1.3.3.2 SLAM

1) *Pose Drift*: After the extrinsic calibration, an approach that fully utilizes multiple LiDARs to reduce the pose drift in SLAM is required. The increasing constraints from multi-LiDAR measurements should help to prevent degeneration in state estimation using single LiDAR and improve the localization accuracy.

2) *Uncertainty*: Mapping a 3D environment using estimated motions and extrinsics commonly suffer uncertainty. A typical phenomenon is that noisy points appear on the global map. This issue affects the reconstruction quality, which will further limit the localization accuracy. A method to model such uncertainties and reject noisy map points must be investigated.

3) *Algorithm Latency*: L-SLAM comprises two major computational tasks: data association (i.e., feature matching) and optimization using the Gauss-Newton algorithm. To

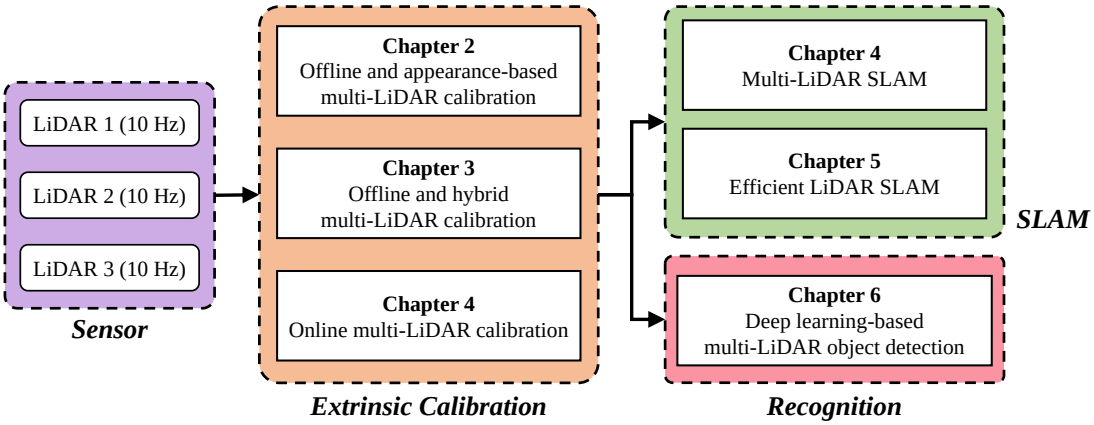


Figure 1.13: Thesis overview.

enforce accuracy, most SLAM systems tend to exploit thousands of features to solve a large NLS problem, while inducing a high latency on onboard processors with limited computation resources. This issue is critical if the scale of SLAM becomes large or more LiDARs are involved. A demand for accelerating current SLAM systems while preserving their performance is raised.

1.3.3.3 Object Detection

1) *Efficient Multi-LiDAR Fusion*: Efficiently fusing multi-LiDAR data using learning-based approaches for object detection is an unexplored topic. The open question: *"Can a multi-LiDAR object detector perform better accuracy than single-LiDAR methods?"* should be answered. Current data fusion methods all have pros and cons [29]. Exhaustive efforts to benchmark possible fusion schemes for the best detection algorithm are required.

1.4 Thesis Overview

This thesis divides the complete problem (1.3) into several subproblems in which one or more challenges are involved. These subproblems are addressed step by step with different assumptions, as summarized in Fig. 1.13.

I start the thesis by proposing two offline multi-LiDAR extrinsic calibration methods: the appearance-based method and the hybrid method. The word "offline" indicates that the robot's poses are pre-computed and the extrinsics are optimized afterward. The appearance-based method (Chap. 2) considers the simplest problem format:

$\hat{\mathcal{Y}} = \arg\min_{\mathcal{Y}} f(\mathcal{Y}, \mathcal{Z}, \mathbf{I})$ where $\mathcal{X}$ is identical and $\mathcal{Z}$ are one-shot point clouds of all LiDARs. It exploits the geometric constraints of three planar surfaces to estimate the extrinsics and is good at accuracy. However, it requires a calibration target (i.e., a wall corner) to provide distinct data correspondences. In contrast, the hybrid method (Chap. 3) takes advantages of motion features for initialization and appearance cues for refinement. It solves the problem: $\hat{\mathcal{Y}} = \arg\min_{\mathcal{Y}} f(\mathcal{Y}, \mathcal{Z}, \mathcal{X})$ by assuming $\mathcal{X}$ to be pre-computed using the state-of-the-art (SOTA) SLAM system. And $\mathcal{Z}$ are all historical point clouds. It releases the needs of calibration targets and initial guesses given by users. But the hybrid method requires that LiDARs' views should have large overlapping regions to guarantee enough constraints.

In Chap. 4, I propose an online system for multi-LiDAR extrinsic calibration and SLAM to address the complete problem: $\hat{\mathcal{X}}, \hat{\mathcal{Y}} = \arg\min_{\mathcal{X},\mathcal{Y}} f(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$. Both $\mathcal{X}$ and $\mathcal{Y}$ are simultaneously optimized over time, and $\mathcal{Z}$ are historical measurements in a sliding window. They are the major differences from the offline methods. With manual intervention, the system can start with several extrinsic-uncalibrated LiDARs, automatically calibrate their extrinsics, and provide accurate poses as well as a globally consistent map. The extrinsic perturbation is also modeled as a zero-mean Gaussian noise with covariance $\mathbf{\Xi}$ and its effect is reduced in the system. In Chap. 5, I accelerate the data association and optimization process by introducing a greedy-based method for feature selection. Only the optimal subset of features $\mathcal{S} \in \mathcal{Z}$ is used to optimize robot's poses and extrinsics. Assume that the noisy extrinsics are known and only the SLAM problem is focused on, the problem is reformulated as $\hat{\mathcal{X}} = \arg\min_{\mathcal{X}} f(\mathcal{X}, \mathcal{Y}, \mathbf{\Xi}, \mathcal{S})$.

In Chap. 6, the multi-LiDAR system is applied in 3D object detection. I integrate my past research with SOTA learning approaches to propose a multi-LiDAR object detector to predict the states of objects from point clouds. Assume that the noisy extrinsics are known, the problem: $\hat{\mathcal{X}} = \arg\min_{\mathcal{X}} f(\mathcal{X}, \mathcal{Y}, \mathbf{\Xi}, \mathcal{Z})$ is addressed. Finally, Chap. 7 concludes this thesis and discusses future research.

1.5 Thesis Contributions

This thesis contributes to several aspects of LiDAR perception. Herein, I summarize the concrete contributions of each chapter in this thesis:

1) **Chap. 2** introduces an offline method that enables automatic dual-LiDAR calibration using three linearly independent planar surfaces. These surfaces provide strong data correspondences between LiDARs and sufficient geometric constraints. The calibration problem is formulated as a typical registration problem and is solved accurately.

2) **Chap. 3** presents an offline method that automatically calibrates multi-LiDAR extrinsics by combining a hand-eye-based method for initialization with an appearance-based module for refinement. This chapter also deals with several practical issues, such as filtering outliers and determining the best extrinsics.

3) **Chap. 4** follows the initialization-refinement paradigm in **Chap. 3** to propose an online procedure to achieve flexible multi-LiDAR extrinsic calibration. The extrinsics are optimized along with the motion estimates. A general criterion is introduced to monitor the convergence and terminate the calibration process. Moreover, the above three calibration methods are quantitatively compared on three robotic platforms.

4) **Chap. 4** also proposes a system called *M-LOAM* that solves multi-LiDAR SLAM by two algorithms: odometry and mapping.

   (a) The odometry is composed of a sliding window-based estimator to fully exploit information from multiple LiDARs. This module can be explained as small-scale frame-to-map registration, further reducing the drift accumulated by frame-to-frame registration.

   (b) The mapping contains a two-stage approach that captures and propagates uncertainties from sensor noise, degenerate pose estimation, and extrinsic perturbation. This module enables the mapping process with an awareness of uncertainty. It helps the system maintain the consistency of a global map and boost the robustness of a system for long-duration navigation tasks.

   M-LOAM's performance has been validated under extensive experiments on handheld devices and autonomous vehicles, covering various scenarios from indoor offices to outdoor urban roads.

5) **Chap. 5** proposes a general and straightforward feature selection algorithm for L-SLAM systems. This algorithm evaluates the environments' degeneracy and adaptively changes the number of selected features to avoid ill-posed estimation. The feature

selection method is integrated into the multi-LiDAR SLAM system in **Chap. 4**, leading to both lower latency and better performance.

6) **Chap. 6** focuses on multi-LiDAR object detection for autonomous driving. I propose a two-stage network called *MLOD* to predict objects' states: the stage-1 network generates object proposals, while the stage-2 network handles the extrinsic perturbation and refines the proposals. This is the first work to systematically study the multi-LiDAR object detection with the awareness of extrinsic perturbation.

Part I

Calibration

CHAPTER 2

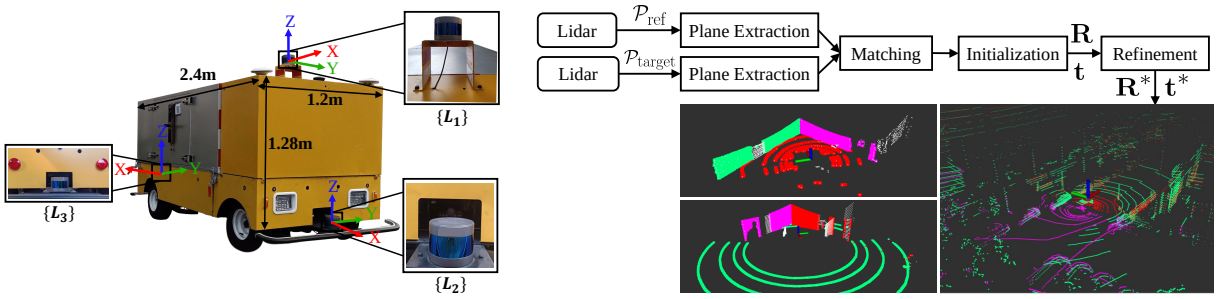
A NOVEL DUAL-LIDAR CALIBRATION ALGORITHM USING PLANAR SURFACES

Multiple LiDARs are used on mobile robots for rendering a broad view to enhance the performance of perception systems. However, precise calibration of multiple LiDARs is challenging since the feature correspondences in scan points are sparse for providing enough constraints. To address this problem, existing methods require fixed calibration targets in scenes or rely exclusively on additional sensors. In this thesis, we present a novel method that enables automatic LiDAR calibration without these restrictions. Three linearly independent planar surfaces appearing in surroundings is utilized to find correspondences. Two components are developed to ensure the extrinsic parameters to be found: a closed-form solver for initialization and an optimizer for refinement by minimizing a nonlinear cost function. Simulation and experimental results demonstrate the accuracy of our calibration approach with the rotation and translation errors smaller than $\mathbf{0.05}rad$ and $\mathbf{0.1}m$ respectively.

2.1 Introduction

Accurate extrinsic calibration has gained importance for vehicles which are equipped with a large number of sensors. Traditional manual calibration techniques, which require fixed calibration targets [55, 107, 109, 188, 216], suffer limited flexibility and tend not to scale well to multi-sensor configurations.

In this thesis, we focus on the automatic calibration methods of 3D LiDARs. With the development of mobile platforms, LiDARs have become one of the most popular sensors for perceiving the environments. Thanks to their accuracy and stability in measuring distance, they are used in many applications [163, 202, 217]. However, much recent work prefers the configuration with multiple LiDARs rather than a single LiDAR because it can render a richer view of environments and offer denser measurements. Several problems such as occlusion and sparsity can be avoided. On any mobile platform containing multiple



(a) Our mobile platform (b) Pipeline of the proposed method

Figure 2.1: Multiple LiDARs in the mobile platform are calibrated using our proposed method. In Fig. 2.1b, the top one visualizes the extracted planes. The bottom one visualizes the calibration results, where the red, pink and pink dots represent the points perceived by different LiDARs.

LiDARs, it is of paramount importance that sensors can be calibrated automatically. When the calibration is finished, all measurements will be correctly projected into a unified coordinate system, as shown in Fig. 2.1b.

Over the past years, automatic methods for calibrating sensors e.g., LiDAR to camera [178], multi-camera [68], and camera to IMU [195] have been proposed. However, few efforts have investigated multi-LiDAR calibration since this problem is challenging; as Choi et al. [33] explained, "searching for correspondences among scan points is difficult". [53] and [64] are two specific multi-LiDAR calibration methods, but several drawbacks are presented. Firstly, they rely exclusively on an additional sensor. Secondly, their success will depend on the quality of initialization provided by users. Thirdly, both of them assume that mobile platforms should undergo efficient motion.

To tackle these issues, we propose a novel approach for calibrating dual LiDARs without any additional sensors and artificial markers. This method assumes that three linearly independent planar surfaces forming a wall corner shape are provided as the calibration targets. Through matching these planes, our method can successfully acquire the unknown extrinsic parameters with two steps: a closed-form solution for initialization and an optimizer for refinement by minimizing a defined cost function. This method has been demonstrated to calibrate three LiDARs on our mobile platform. An overview of the method is shown in Fig. 2.1b. In solving calibration with poor human intervention, we make two significant contributions in this thesis:

- We make it possible to use objects with unknown size in the outdoor environment as the calibration target.

- We demonstrate that our method is efficient in applications since the extrinsic parameters can be obtained immediately with one-shot measurement.
- We release the implementation and experimental data.[1]

2.2 Related Work

2.2.1 Calibration of Multi-Lidar Systems

In recent years, Gao and Spletzer [53] proposed an algorithm to calibrate multiple LiDARs using point constraints provided by retro-reflective tapes in scenes. He et al. [64] demonstrated a technique to extract geometric features among point clouds, which enables an offline algorithm to calibrate multiple 2D LiDARs in arbitrary scenes. Shortly after that, their approach was improved in a challenging scenario: an underground parking lot, where GPS is not available [63]. However, such methods rely on an additional localization module, making the calibration process complicated.

Artificial landmarks are prevalently used to find correspondences among sensor data. Xie et al. [189] provided a general solution to jointly calibrate multiple cameras and LiDARs in the presence of a pre-built environment with apriltags. Steder et al. [155] proposed a tracking-based method to calibrate multiple 2D LiDARs using a moving object which appears in their overlapping areas. Based on it, Quenzel et al. [150] calibrated the same sensors with an additional verification step. However, these approaches require artificial markers to be placed in scenes. In this thesis, we exploit common planar surfaces as the calibration target inspired by [33], but our approach differs from it by releasing the orthogonal assumption of these surfaces to achieve outdoor calibration. These planar surfaces, as shown in experiments, can be easily found from urban environments.

2.2.2 Calibration of Other Multi-Sensor Systems

There exist several published papers on LiDAR-camera, multi-camera and camera-IMU calibration. One of the first work to solve online camera and LiDAR calibration is [104]. In this method, edge features in images are associated with LiDAR measurements using depth discontinuities. The extrinsic parameters are optimized by minimizing a cost

[1] `https://github.com/ram-lab/LiDAR_appearance_calibration`

function. Different metrics based on Gradient Orientation Measure (GOM) [178], Mutual Information (MI) [135], and line-plane constraints [216] were also proposed. However, all of them require initialization provided by users. In our proposed method, we introduce an algorithm to automatically initialize the extrinsic parameters by exploiting the geometric constraints of planar surfaces.

Developed from hand-eye calibration using the structure-from-motion techniques, motion-based approaches have been implemented to solve the calibration. Heng et al. [68] proposed CamOdoCal, an automatic algorithm for four-camera calibration without the assumption of overlapping fields of view. They decouple the calibration process into initialization and refinement. In initialization, a rough estimate of extrinsic parameters is computed by combining visual odometry with the vehicle's egomotion. To refine the estimates, a bundle adjustment is used to optimize all of the cameras' poses and feature data. This pipeline is employed in our method. However, CamOdoCal was explicitly designed for vision sensors, which may not be feasible in various sensor configurations. In contrast, Taylor and Nieto released a system [176, 177] to calibrate multiple heterogeneous sensors and their time offset. Generally, motion-based methods can work for a variety of configurations and can be integrated into several SLAM systems [145]. However, the calibration accuracy of motion-based methods is limited due to the drift of computed odometry, which needs to be refined using the appearance cues from surrounding environments.

2.3 Methodology

Our approach makes use of three linearly independent planar surfaces to calibrate a dual-LiDAR system. To achieve it, we firstly introduce a robust algorithm to extract planes from scan points. The geometric structure of these planes is then utilized to acquire the extrinsic parameters. We define $\{\}^{l_k}$ as a 3D coordinate system with its origin at the geometric center of the k^{th} LiDAR. The $x-$, $y-$ and $z-$ axes are pointing forward, left and upward respectively. In this thesis, we consider $\{\}^{l_1}$ as the reference frame, and $\{\}^{l_k}$ as the target frame. The point clouds perceived by a LiDAR is denoted by $\mathcal{P}$, and the coordinates of a point in $\mathcal{P}$ is represented as $\mathbf{p}_n = [x_n, y_n, z_n]^\top$. Detailed notations are listed in Table 2.1 and are visualized in Fig. 2.2.

Table 2.1: Annotation table.

Notation	Explanation
$\{\}^{l_1}$ / $\{\}^{l_k}$	Reference / Target coordinate system
Π_i / Π_i'	i^{th} planar surfaces in $\{\}^{l_1}$ / $\{\}^{l_k}$
$\mathbf{o}$ / $\mathbf{o}'$	Intersection point in $\{\}^{l_1}$ / $\{\}^{l_k}$
$\boldsymbol{\beta}_i$ / $\boldsymbol{\beta}_i'$	Coefficients of Π_i / Π_i'
$\mathbf{n}_i$ / $\mathbf{n}_i'$	Unit normal vector of Π_i / Π_i'

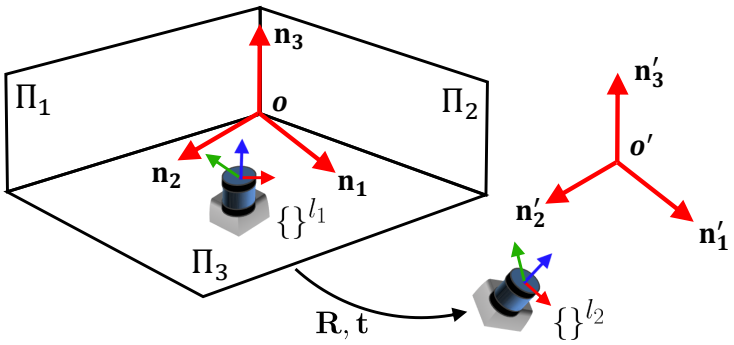


Figure 2.2: A diagram of the notations. Red, green, and blue arrows denote the $x-$, $y-$, and $z-$ axes of each LiDAR coordinate system respectively.

2.3.1 Plane Extraction

Denoting $\boldsymbol{\beta}_i = \left[\beta_{(i,0)}, \beta_{(i,1)}, \beta_{(i,2)}, \beta_{(i,3)}\right]^\top$ the coefficients of Π_i, the distance between $\mathbf{p}_n$ and Π_i [see Fig. 2.3] is computed as follows:

$$f_i(\mathbf{p}_n) = \left|\beta_{(i,0)} x_n + \beta_{(i,1)} y_n + \beta_{(i,2)} z_n + \beta_{(i,3)}\right|. \tag{2.1}$$

To fit a planar model from a series of discrete points, we employ the random sample consensus (RANSAC) algorithm. By randomly selecting N points from $\mathcal{P}$, the planar coefficients are acquired by solving a least-squares problem [44]:

$$\boldsymbol{\beta}_i^* = \arg\min_{\boldsymbol{\beta}_i} \sum_{n=1}^{N} f_i^2(\mathbf{p}_n), \tag{2.2}$$

where the parameter vector $\boldsymbol{\beta}_i$ will be updated iteratively until an optimal model is acquired with maximum inlier points. To determine whether a point is an inlier, its square distance to a plane is computed. To extract three models, the RANSAC algorithm is executed separately at three times. At each time, points belonging to former extracted models are ignored. Finally, we can obtain three groups of planar coefficients which are

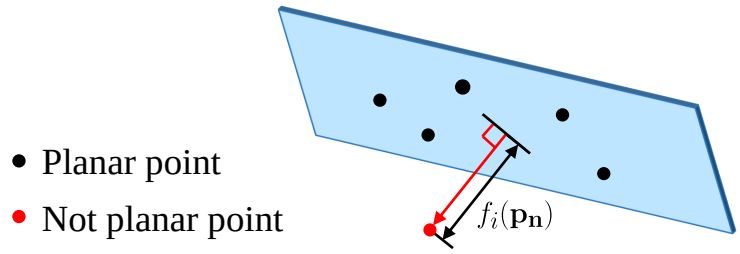


Figure 2.3: Point to Plane distance.

denoted by $\boldsymbol{\beta}_1$, $\boldsymbol{\beta}_2$, and $\boldsymbol{\beta}_3$ respectively to describe the planar surfaces. Hence, we can compute $\mathbf{o} = [o_x, o_y, o_z]^\top$ by solving a set of linear systems

$$\begin{bmatrix} \boldsymbol{\beta}_{(1,0)} & \boldsymbol{\beta}_{(1,1)} & \boldsymbol{\beta}_{(1,2)} & \boldsymbol{\beta}_{(1,3)} \\ \boldsymbol{\beta}_{(2,0)} & \boldsymbol{\beta}_{(2,1)} & \boldsymbol{\beta}_{(2,2)} & \boldsymbol{\beta}_{(2,3)} \\ \boldsymbol{\beta}_{(3,0)} & \boldsymbol{\beta}_{(3,1)} & \boldsymbol{\beta}_{(3,2)} & \boldsymbol{\beta}_{(3,3)} \end{bmatrix} \begin{bmatrix} o_x \\ o_y \\ o_z \\ 1 \end{bmatrix} = \mathbf{0}. \tag{2.3}$$

After computing $\boldsymbol{\beta}_i$, the unit normal vectors $\mathbf{n}_1$, $\mathbf{n}_2$ and $\mathbf{n}_3$ can be represented up to scale. According to our assumption of linear independence, there exist three non-zero scalars a, b and c that satisfy the following equation:

$$a\mathbf{n}_1 + b\mathbf{n}_2 + c\mathbf{n}_3 = -\mathbf{o}, \tag{2.4}$$

where we can fix the directions of normal vectors to make a, b and c positive.

Fig. 2.1b shows an example of the extracted planes, where the color values represent their extraction order. We can observe that the corresponding planes do not have the same order. By utilizing the wall corner shape that is a geometric prior (see Fig. 2.2), we find that the orders of planar surface can be rearranged for correct data association. Without loss of generality, we set Π_1 and Π_2 as the **left** and **right** plane respectively, and Π_3 as the **bottom** plane. Their normal vectors should follow the right-hand rule:

$$(\mathbf{n}_2 \times \mathbf{n}_1) \cdot \mathbf{n}_3 > 0. \tag{2.5}$$

Following the above steps, we can correctly match the corresponding planes between two LiDARs.

2.3.2 Initialization Using Closed-Form Solution

We can formulate the calibration of dual-LiDAR as a nonlinear optimization problem by minimizing the distance between corresponding planes. But the defined cost function (2.8) is non-convex. To avoid local minima, the parameters should be firstly initialized.

According to Section 2.3.1, we already have two sets of fitted planes Π and Π' with known normal vectors. Consequently, their relative rotation $\mathbf{R}$ can be thus computed for initialization by introducing the Kabsch algorithm [85]. The Kabsch algorithm is an effective approach that provides a least-squares solution to calculate the rotation between a pair of vector sets. We use $\mathbf{P}$, $\mathbf{Q} \in \mathbb{R}^3$ to indicate two 3×3 matrices. The elements at the ith column of $\mathbf{P}$ are defined as $(\mathbf{n}_i - \mathbf{o})$, and those of $\mathbf{Q}$ are defined as $(\mathbf{n}'_i - \mathbf{o}')$. We also denote $\mathbf{H} = \mathbf{P}^\top \mathbf{Q}$ the cross-covariance matrix. By calculating the singular value decomposition (SVD) of $\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, $\mathbf{R}$ can be computed as:

$$\mathbf{R} = \mathbf{V} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{pmatrix} \mathbf{U}^\top, \text{ where } d = \det(\mathbf{V}\mathbf{U}^\top). \tag{2.6}$$

On the contrary, the relative translation $\mathbf{t}$ is computed using the plane intersections:

$$\mathbf{t} = \mathbf{o}' - \mathbf{o}. \tag{2.7}$$

2.3.3 Nonlinear Optimization

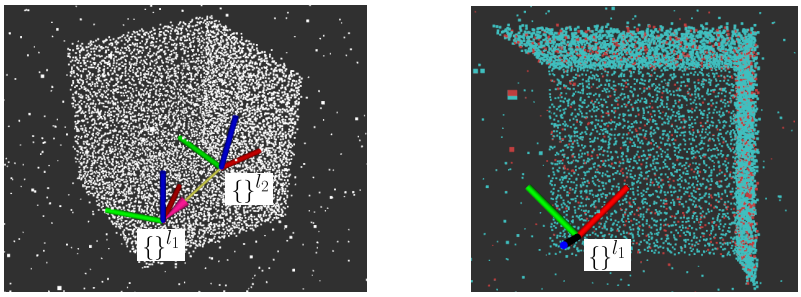
The initial solution is further refined via a nonlinear optimization. By defining a cost function to describe the euclidean distance between Π_i and Π'_i, it can be computed as a sum of the squired distance between a point $\mathbf{p}_n$ and its corresponding plane, i.e., $f_i^2(\mathbf{p}_n)$. Therefore, we can write down the cost function and adopt a Levenberg-Marquardt algorithm for the nonlinear optimization:

$$\begin{aligned} \mathbf{R}^*, \mathbf{t}^* &= \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{3} \mathcal{F}(\mathbf{R}, \mathbf{t}, \Pi'_i, \Pi_i) \\ &= \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{3} \Big[\sum_{\mathbf{p}' \in \Pi'_i} f_i^2 \left(\mathbf{R}\mathbf{p}' + \mathbf{t}\right) + \sum_{\mathbf{p} \in \Pi_i} f_i'^2 \left(\mathbf{R}'\mathbf{p} + \mathbf{t}'\right) \Big]^2, \end{aligned} \tag{2.8}$$

where $f'_i(\cdot)$ is the counterpart of f_i, and $\mathbf{R}' = \mathbf{R}^{-1}$ as well as $\mathbf{t}' = -\mathbf{R}^{-1}\mathbf{t}$ are a rotation matrix and a translation vector from $\{\}^{l_2}$ to $\{\}^{l_1}$ respectively.

2.4 Experiment

To evaluate the proposed extrinsic calibration method, we test it with different configurations of dual LiDARs. Experiments are presented with synthetic data and real



(a) Sythetic data

(b) Calibrated and merged point cloud

Figure 2.4: (a) An example of the synthetic data. (b) The red points and cyan points indicate the point cloud perceived by l_1 and l_2 respectively. We transform the cyan points from $\{\}^{l_1}$ to $\{\}^{l_2}$ using the results provided by our proposed method.

sensor data. All the resulting values are compared against the ground truth or the values provided by four methods in terms of accuracy.

2.4.1 Implementation Details

We adopt `pcl`[2] to preprocess point clouds and implement the RANSAC-based plane fitting. `Eigen`[3] library is applied to implement the Kabsch algorithm, and `Ceres Solver`[4] is used to solve the nonlinear optimization problem. In the optimization, we set the maximum iteration as 1000 and stopping tolerance as $1e^{-3}$.

2.4.2 Experiments in Synthetic Data

To verify the performance of the proposed algorithm, we randomly generate 9500 scan points (7500 planar points and 2000 noisy points) in a $10m \times 10m \times 10m$ space. The planar points are generated evenly on three planar surfaces, which are subjected to zero-mean Gaussian noise with a standard deviation of $0.1m$. The rotation angles α on z-axis between Π_1 and Π_2 are set at intervals of $(60°, 120°)$, while Π_3 is set on the bottom, which is orthogonal to Π_1 and Π_2. The noisy points are distributed in the space, which are subjected to zero-mean Gaussian distribution with a standard deviation of 5m. l_1 is set arbitrarily where all the planar surfaces can be observed. Rotations from $\{\}^{l_1}$ to $\{\}^{l_2}$ are randomly generated within $(0°, 20°)$, $(0°, 20°)$, $(0°, 360°)$ on $x-$, $y-$ and $z-$ axis

[2] `http://pointclouds.org`

[3] `http://eigen.tuxfamily.org`

[4] `http://ceres-solver.org`

Table 2.2: The ground truth of two simulated configurations.

Configuration	Rotation $[rad]$	Translation $[m]$
1	2.7337, -0.3946, -0.1809	0.8766, 0.4672, 1.0474
2	-0.5174, 0.1277, 0.1222	1.3785, -1.3929, 1.3020

Table 2.3: The calibration results on simulated data.

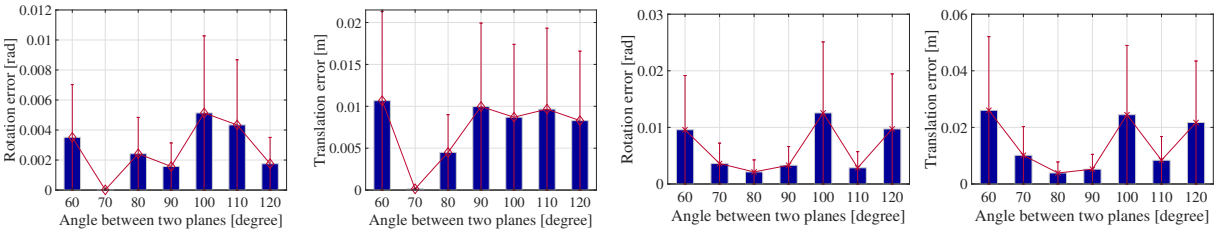
Configuration	α $[degree]$	Rotation Error $[rad]$		Translation Error $[m]$	
		mean	std.	mean	std.
1	60	0.0035	0.0035	0.0107	0.0161
	70	0	0	0.0001	0
	80	0.0024	0.0056	0.0045	0.0085
	90	0.0016	0.0040	0.0100	0.0243
	100	0.0051	0.0107	0.0087	0.0178
	110	0.0043	0.0063	0.0097	0.0161
	120	0.0018	0.0051	0.0083	0.0243
2	60	0.0096	0.0104	0.0260	0.0349
	70	0.0036	0.0083	0.0101	0.0274
	80	0.0021	0.0064	0.0039	0.0109
	90	0.0033	0.0071	0.0052	0.0101
	100	0.0126	0.0170	0.0245	0.0339
	110	0.0029	0.0057	0.0084	0.0163
	120	0.0097	0.0139	0.0217	0.0352

respectively, and translations are generated within $(-1.5, 1.5)m$ respectively. An example of the sensor configuration and the generated points is visualized in Fig. 2.4a. In our experiments, we randomly select two configurations with different $\mathbf{R}$ and $\mathbf{t}$ [see Table 2.2] as the ground truth to compare with the resulting values.

The difference in rotation is measured by the angle difference between the ground truth $\mathbf{R}_{\text{gt}}$ and the resulting rotation $\mathbf{R}_{\text{res}}$, which is calculated as $e_r = \| \log(\mathbf{R}_{\text{gt}}\mathbf{R}_{\text{res}}^{-1})^{\vee} \|_2$.[5] The difference in translation is computed using vector subtraction as $e_t = \|\mathbf{t}_{\text{gt}} - \mathbf{t}_{\text{res}}\|_2$.

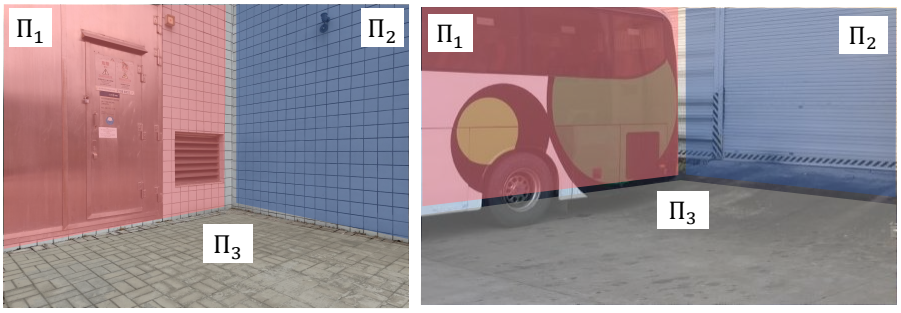
For each group of $\mathbf{R}$ and $\mathbf{t}$, we performed 10 trials on the noisy data and computed the mean as well as the standard deviation of the rotation and translation errors. In Fig. 2.5, blue bars and red lines indicate the mean and standard deviation respectively. Detailed calibration results are shown in Table 2.3. An example of the calibrated point cloud is shown in Fig. 2.4b. In summary, we can see that the rotation and translation errors are tiny on the synthetic data. This proves that the proposed method can successfully

[5]The operator $\boldsymbol{\phi} = \log(\mathbf{R})^{\vee}$ is defined to associate $\mathbf{R}$ in SO(3) to its rotation angle $\boldsymbol{\varphi} \in \mathbb{R}^3$ on the axis.



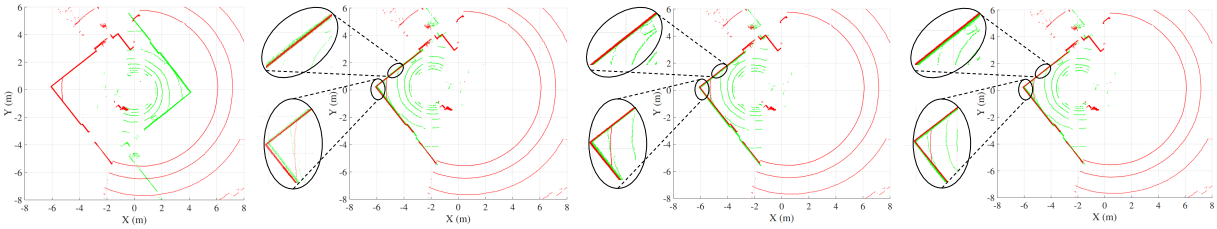
(a) Configuration 1 (b) Configuration 2

Figure 2.5: Performance analysis using rotation and translation errors on two simulated configurations.



(a) Easy scenario. (b) Hard scenario.

Figure 2.6: The calibration environments which can be found in outdoors. They all form a wall corner shape. We extract three linear independent planar surfaces (Π_1, Π_2, and Π_3) for calibration.



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

Figure 2.7: Top views of (a) the uncalibrated point clouds, point clouds calibrated by (b) W/O refinement, (c) proposed method, and (d) groundtruth in the *easy* scenario. The point cloud of l_1 and l_3 are denoted by red and green dots respectively.

calibrate the extrinsic parameters.

2.4.3 Experiments in Real Data

We calibrate a sensor system which consists of three 16-beam RS-Lidars[6] on our vehicle. As presented in Fig. 2.1a, these LiDARs are mounted at the front (l_1), top (l_2), and tail (l_3) position respectively. Especially, l_3 is mounted with approximately 180° rotation

[6] `https://www.robosense.ai/rsLiDAR/rs-LiDAR-16`

offset in yaw. In later sections, we denote $l_1 \ominus l_i$ the configuration between l_1 and l_i. The surrounding buildings and ground as the scan planar surfaces are used for calibration. We select two calibration environments in outdoors with two levels (*easy* and *hard*) for calibration, which are shown in Fig. 2.6.

Calibration is performed in the case of two different configurations: $l_1 \ominus l_2$, as a standard setup, and $l_1 \ominus l_3$, as a challenging setup. We take three methods for comparisons. The former two methods are developed based on the proposed one, but some steps are modified, while the last method is based on the motion-based techniques:

- W/O refinement: The refinement step is removed.
- ICP refinement: The nonlinear optimization refinement is replaced by a point-to-plane ICP [140].
- Motion-based: The motion of LiDARs are estimated using the LeGO-LOAM [163], and the approach in [176] is implemented to initialize the extrinsic parameters. In refinement, we use the approaches in [134] based on ground surface to obtain the translation on $z-$ axis.

2.4.3.1 Easy Scenario

Since the precise extrinsic parameters of the multi-LiDAR system are unknown, we use the values provided by the manufacturer as the ground truth to evaluate these methods. The calibration results on different configurations are listed in Table 2.4 and 2.5. The estimated extrinsic parameters of the proposed algorithm are quite close to the ground truth. Regarding the relative rotation and translation errors, our method achieved $[0.02rad, 0.067m]$ of $l_1 \ominus l_2$ and $[0.02rad, 0.084m]$ of $l_1 \ominus l_3$ respectively. We observe that larger errors are caused by ICP refinement and Motion-based methods. Regarding the ICP approach, failure is caused due to the wrong matching of corresponding planes. About the Motion-based approach, the drift and uncertainty of the estimated motion would significantly reduce the accuracy of the calibration results. We can also see that the proposed method performs better than the w/o refinement method since the nonlinear optimization could further reduce the noise. During the calibration, the time for optimization is around $27.3s$ and $35.8s$ of the two configurations.

Table 2.4: Estimated extrinsic parameters and errors on $l_1 \ominus l_2$.

Method	Rotation $[rad]$	Error $[rad]$	Translation $[m]$	Error $[m]$
Ground truth	0.01, 0.10, 0.04	—	0.377, -0.033, -1.232	—
W/O refinement	-0.00, -0.11, -0.02	0.22	0.315, 0.051, -1.234	0.103
ICP refinement	-0.09, -0.43, 0.05	0.53	1.219, -0.067, -1.560	0.904
Motion-based	0.23, -0.05, -0.00	0.23	0.212, 0.489, -1.190	0.547
Proposed	0.00, 0.09, 0.03	**0.02**	0.336, 0.003, -1.191	**0.067**

Table 2.5: Estimated extrinsic parameters and errors on $l_1 \ominus l_3$.

Method	Rotation $[rad]$	Error $[rad]$	Translation $[m]$	Error $[m]$
Ground truth	-0.016, -0.02, 3.14	—	-1.964, 0.041, -1.138	—
W/O refinement	-0.05, -0.02, 3.13	0.04	-1.921, 0.097, -0.802	0.342
ICP refinement	-0.03, -0.13, 3.13	0.16	-1.959, 0.047, -0.384	0.754
Motion-based	-0.00, 0.00, -2.65	0.50	-1.779, -0.467, -1.111	0.547
Proposed	0.00, -0.00, 3.14	**0.02**	-1.910, 0.053, -1.074	**0.084**

To evaluate the calibration results qualitatively, we transform the point cloud in $\{l_3\}$ to $\{l_1\}$ using the extrinsic parameters provided by ground truth, W/O refinement, and Proposed respectively. The top view of these fused point cloud can be seen in Fig. 2.7. We observe that the point clouds calibrated by the Proposed approach have litter uncertainty on the planar surfaces. Therefore, our algorithm can successfully calibrate dual LiDARs in real data with low error.

2.4.3.2 Hard Scenario

In the following, we study the performance of our approach in the *hard* scenario. This scenario is more challenging because there are several objects on these planes that may influence the plane extraction and optimization results. In this experiment, the configuration $l_1 \ominus l_3$ is calibrated. The relative rotation and translation errors compared with the ground truth are $[0.03rad, 0.09m]$, while the optimization time is around $106.4s$. The fused point clouds are shown in Fig. 2.8, where the planar surfaces are registered well without much offset. We conclude that the extrinsic parameters can be recovered in this scenario.

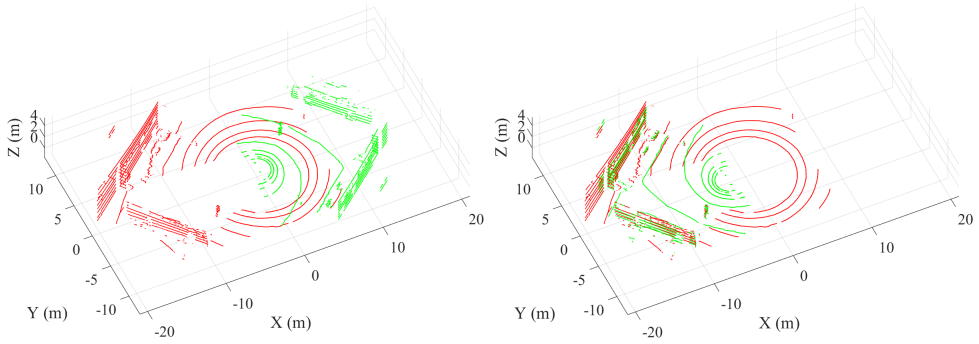


Figure 2.8: Top views of the uncalibrated point clouds (left) and point clouds (right) calibrated by the proposed method in the *hard* scenario. The point cloud of l_1 and l_3 are denoted by red and green dots respectively..

2.4.4 Discussion

We have three certain assumptions in this method: LiDARs are horizontally mounted; they share large overlapping fields of view with each other and three planar surfaces are provided for calibration. Therefore, the proposed method may fail in several cases. For instance, if LiDARs are mounted at an arbitrary orientation, a wrong initialization will be caused. Another case is that if planar surfaces are wrongly detected, their correspondences will be mismatched.

2.5 Conclusion and Future Work

In this thesis, we have presented an automatic algorithm for calibrating a dual-LiDAR system without any additional sensors, artificial landmarks, or information about the motion provided by sensors. The RANSAC-based model fitting approach is used to extract three linearly independent planar surfaces from scan points. Linear constraints for initialization are provided by the geometric structure of these planar surfaces, and a final nonlinear optimization is then used to refine the estimates. Our proposed method has been demonstrated to recover the extrinsic parameters of a dual-LiDAR system with rotation and translation error smaller than $0.05rad$ and $0.1m$ in different testing conditions.

It would be beneficial to determine the parameters in more general cases, e.g., sensors have non-overlapping fields of view, or they are arbitrarily mounted on a vehicle. Such problems may be solved by developing a simultaneous localization and mapping system, where the extrinsic parameters are jointly optimized with the odometry and map within a unified framework.

CHAPTER 3

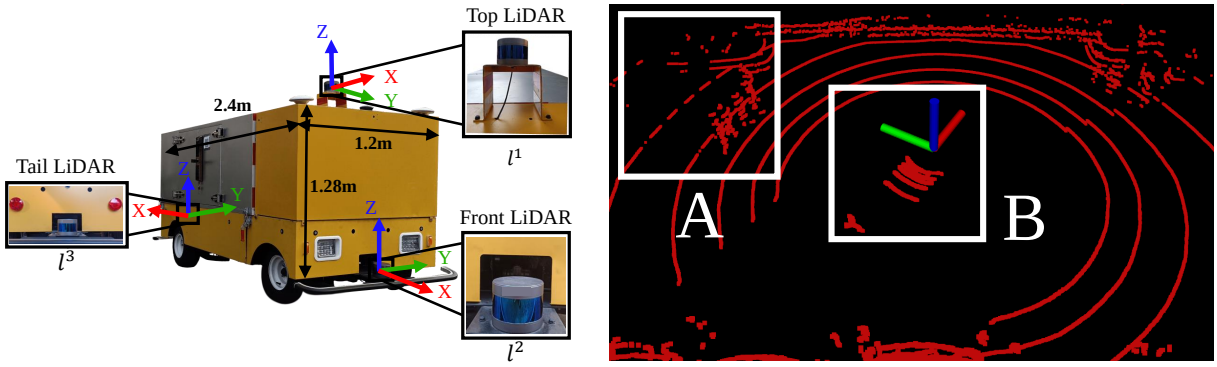
AUTOMATIC CALIBRATION OF MULTIPLE 3D LIDARS IN URBAN ENVIRONMENTS

Multiple LiDARs have progressively emerged on autonomous vehicles for rendering a rich view and dense measurements. However, the lack of precise and flexible calibration methods negatively affect potential applications of multiple LiDARs. In this thesis, we propose a novel system that enables automatic multi-LiDAR calibration method without any calibration target, prior environment information, and manual initialization. Our approach starts with a hand-eye calibration by aligning the motion of each sensor. The initial transformations are then refined by an appearance-based method by minimizing a point-to-plane residual function. Experimental results on simulated and real-world data demonstrate the reliability and accuracy of our calibration approach. The proposed approach can calibrate a multi-LiDAR system with the rotation and translation errors less than $\mathbf{0.04}rad$ and $\mathbf{0.1}m$ respectively for a mobile platform.

3.1 Introduction

Accurate extrinsic calibration has become increasingly essential for a series of applications using multiple sensors. Numerous research work has been studied on [68, 176, 216]. Over the past decades, LiDARs have appeared as a dominant sensor in mobile robotics for their active nature of providing accurate and stable distance measurements. They have been widely utilized in mapping [10] and object detection [201]. However, LiDARs do not provide a high spatial resolution of measurements and are also sensitive to occlusion. These drawbacks limit their potential applications in robotic systems. Fig. 3.1 (bottom) displays two examples, which is a point cloud captured by the top LiDAR. In block A, the pedestrians and vehicles are scanned with a few points, making the detection of these objects challenging. About block B, points are gathering together because of occlusion. Therefore, employing the multi-LiDAR setup on self-driving cars is necessary.

Traditional calibration techniques for multiple sensors are done by either placing markers in scenes or hand-labeled correspondences. However, these approaches suffer from im-



(a) (b)

Figure 3.1: (a) Our vehicle consists of a multi-LiDAR system with unknown extrinsic parameters. (b) A point cloud captured by l^1. The white boxes indicate two drawbacks presented in a single LiDAR configuration: (A) measurement sparsity and (B) occlusion.

practicality and limited scalability to the multi-LiDAR configuration. Additionally, there are surprisingly few discussions on calibrating multiple 3D LiDARs. The majority of current approaches involve one or more of the following assumptions: prior knowledge about the structure of environments [80], usage of additional sensors [64], and user-provided initial values of the extrinsic parameters [104]. It is also not trivial to register sparse point clouds accurately without initial guess. Inspired by the progress of the hand-eye calibration, we find that extrinsic parameters can be recovered from individual motion of each LiDAR. Moreover, the geometric features in environments also form constraints to sensors' relative transformations. Hence, we can conclude that the complementary usage of these approaches is a prospective solution to calibrate multiple LiDARs.

In this thesis, we proposed a novel system that allows automatic calibration of multiple LiDARs in urban environments. It consists of three components: **motion estimation** of each sensor, motion-based **initialization**, and appearance-based **refinement**. We show a variety of experiments to demonstrate the reliability and accuracy of this approach. The contributions of this thesis are summarized as follows:

- A pipeline to automatically calibrate the extrinsic parameters of multiple LiDARs which releases the assumptions of calibration targets, prior knowledge about surroundings, and initial values given by users.
- The usage of the motion-based method for initialization and appearance-based method for refinement.
- Extensive experiments on simulated and real-world data.

- Implementation has been released.[1]

3.2 Related Work

Besides multi-LiDAR calibration, there are also extensive discussions on the calibration among LiDARs, cameras, and IMUs. Some of them are related to our method. In this section, we categorize them as appearance-based and motion-based approaches.

3.2.1 Appearance-based approaches

Appearance-based approaches that recover the spatial offset using appearance cues in surroundings are considered as a category of registration problem. Data association of multiple sources of data is the key challenge Calibration targets (i.e., chessboards and high-reflective markers) that are commonly detected by sensors have been prevalently used to acquire correct correspondences. Our previous work [80] exploited three planar surfaces from a wall corner to calibrate multiple LiDARs. Regarding the LiDAR-camera setup, Zhou et al. [216] used a chessboard to establish line and planar correspondences between point clouds and images to optimize the extrinsics. Liao et al. [108] considered the polygon as the calibration target that is more handy to acquire and proposed to minimize the vertex, edge, and inner-point constraints. However, these methods require targets to be observable to sensors, which is not flexible in non-structural scenarios. In contrast, our method only utilizes features such as edges and small planar patches, which is a minimum requirement of outdoor environments. These features are common and have been exploited in outdoor SLAM systems [163, 205].

Automatic target-less calibration schemes in arbitrary scenes has led the trend recently. He et al. [64] extracted geometric features among scan points to achieve robust registration. This work was further extended to a challenging scenario [63]. Levinson and Thrun [104] first put forward an online calibration for a camera-LiDAR system. This is accomplished by aligning 3D points with depth discontinuity onto 2D image edge marks. However, their success highly relies on the initialization given by users. Compared with them, our approach can initialize the extrinsics from the sensor's motion, which enables calibration without careful measuring.

[1]Code is available at `https://github.com/ram-lab/MLC`

3.2.2 Motion-based approaches

Motion-based approaches treat calibration as a well-researched hand-eye calibration problem [71], where the extrinsic parameters are computed by combining the motions of all available sensors. The hand-eye calibration problem is usually referred to solve $\mathbf{X}$ in $\mathbf{AX} = \mathbf{XB}$, where $\mathbf{A}$ and $\mathbf{B}$ are the motions of two sensors, and $\mathbf{X}$ is their relative transformation. As described in [36], this problem has been well formulated and explored since the 1980s. The ongoing research focuses on the calibration of multiple sensors in outdoor environments. Heng et al. [68] proposed CamOdoCal, a versatile algorithm with a bundle adjustment to calibrate four cameras. Taylor et al. [176] provided a more general solution to multi-modal sensors. Several state-of-the-art visual-inertial navigation systems adopted the motion-based approaches for online calibration [145, 196]. Temporal calibration is also a key issue in multi-sensor fusion. As present in [147], motion-based approaches can also be utilized to estimate temporal offset between a typical interoceptive and exteroceptive sensor (i.e., an IMU and a camera). Although motion-based methods have been extensively developed, their accuracy is sensitive to the accumulated drifts of estimated motion. In contrast, our method takes advantages of geometric features to improve the calibration of multiple LiDARs.

3.3 Overview

The notations are defined as follows. We denote $[0, K]$ the time interval during calibration, and define $\{l_k^i\}$ as the sensor coordinate system of the i^{th} LiDAR at timestamp k. The $x-$, $y-$ and $z-$ axes of coordinate systems are pointing forward, left and upward respectively. We denote I the number of LiDARs to be calibration, and l^1 the reference LiDAR. The transformation from $\{a\}$ to $\{b\}$ are denoted by $\mathbf{T}_b^a$. It can be decomposed into the rotational and translational component ($\mathbf{R}_b^a$ and $\mathbf{t}_b^a$). $\mathbf{T}_{l^i}^{l^1}$ are the unknown transformations from $\{l^1\}$ to $\{l^i\}$ $(i \neq 1, i \leq I)$ to be solved. LiDARs are synchronized via. the hardware trigger, meaning that the corresponding multi-sensor data can be associated at a high success rate. Before the calibration process, we are able to manually drive the robot platform with sufficient motion excitation over a planar surface. With these notations and assumptions, the calibration problem is defined as:

Problem: Given a sequence of point clouds during calibration, computing the ex-

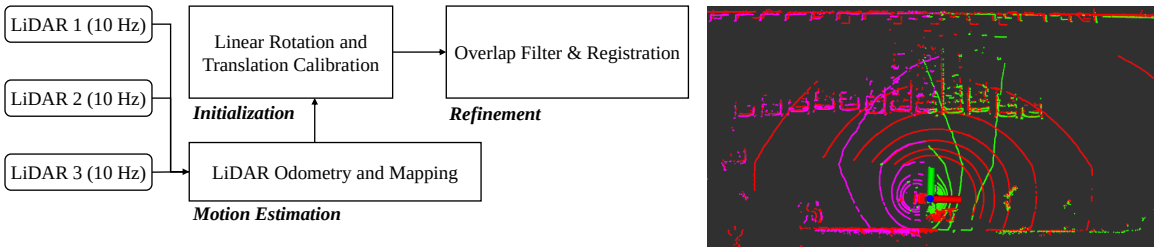


Figure 3.2: This figure illustrates the full pipeline of the proposed approach (left) and the fused point clouds after calibration (right). Note that the red, green, and purple point clouds are captured by the top LiDAR, front LiDAR, and tail LiDAR respectively.

trinsics of a multi-LiDAR system by combining motion information with surrounding appearance.

As illustrated in Fig. 3.2, the pipeline of our proposed calibration system consists of three phases: **motion estimation**, **motion-based initialization**, and **appearance-based refinement**. The motion estimation phase takes point clouds as input, and results in the incremental motions of each LiDAR within a time interval $[k-1, k]$ (see Section 3.4.1). The initialization phase roughly computes $\mathbf{T}_{l^i}^{l^1}$ using a least-squares solution (see Section 3.4.2). Finally, the refinement phase utilizes the appearance cues in surroundings to register different LiDARs (see Section 3.4.3).

3.4 Methodology

3.4.1 Motion Estimation

To calculate incremental motions between consecutive frames of each LiDAR, the LeGO-LOAM (a LiDAR SLAM system) [163] is used. LeGO-LOAM registers edge and planar features to estimate sensors' motion. In our implementation, the individual transformations of all the sensors and extracted ground points are used to provide constraints for the extrinsic parameters. Details are introduced in Section 3.4.2 and Section 3.4.3.

3.4.2 Motion-Based Initialization

With l^1 as the reference sensor, we present a method of calibrating l^1 with l^i pairwise. To simplify the notations, we replace $\{l^1\}, \{l^i\}$ with $\{a\}, \{b\}$ to indicate the coordinate system of the reference sensor and the target sensor respectively. The constant transformation of two LiDARs can be initialized by aligning their estimated motion. Fig. 3.3

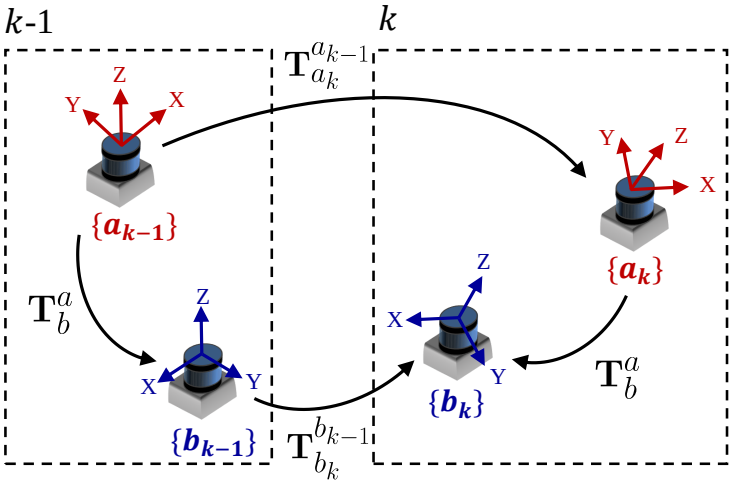


Figure 3.3: The transformations between different LiDARs at $[k-1, k]$.

depicts the relationship between the motion of two LiDARs and their relative transformation. As the vehicle moves, the extrinsic parameters can be recovered using these poses for any k:

$$\mathbf{T}_{a_k}^{a_{k-1}}\mathbf{T}_b^a = \mathbf{T}_b^a\mathbf{T}_{b_k}^{b_{k-1}}, \tag{3.1}$$

where (3.1) can be decomposed in terms of its rotation and translation components with the following equations:

$$\mathbf{R}_{a_k}^{a_{k-1}}\mathbf{R}_b^a = \mathbf{R}_b^a\mathbf{R}_{b_k}^{b_{k-1}}, \tag{3.2}$$

$$(\mathbf{R}_{a_k}^{a_{k-1}} - \mathbf{I}_3)\mathbf{t}_b^a = \mathbf{R}_b^a\mathbf{t}_{b_k}^{b_{k-1}} - \mathbf{t}_{a_k}^{a_{k-1}}. \tag{3.3}$$

The method described in [68] is used to solve these two equations. Based on (3.2), the pitch-roll rotation can be calculated directly using the estimated rotations, while the yaw rotation, the translation can be computed using (3.3).

3.4.2.1 Outlier Filter

The poses of sensors have two constraints that are independent of the extrinsic parameters, which were presented as the screw motion in [26]:

$$\theta_{a_k}^{a_{k-1}} = \theta_{b_k}^{b_{k-1}}, \qquad \mathbf{r}_{a_k}^{a_{k-1}} \cdot \mathbf{t}_{a_k}^{a_{k-1}} = \mathbf{r}_{b_k}^{b_{k-1}} \cdot \mathbf{t}_{b_k}^{b_{k-1}}, \tag{3.4}$$

where θ denotes the angle of a rotation matrix $\mathbf{R}$, and $\mathbf{r}$ is the corresponding rotation axis[2]. The screw motion residuals include rotation and translation residuals, which are calculated as: $|\theta_a - \theta_b|$, $\|\mathbf{r}_a \cdot \mathbf{t}_a - \mathbf{r}_b \cdot \mathbf{t}_b\|^2$. We adopt the screw motion residuals to evaluate

[2]The rotation angle and axis are calculated using the $\log(\cdot)^\vee$ operator such that $\boldsymbol{\phi} = \log(\mathbf{R})^\vee, \boldsymbol{\phi} = \theta\mathbf{r}$.

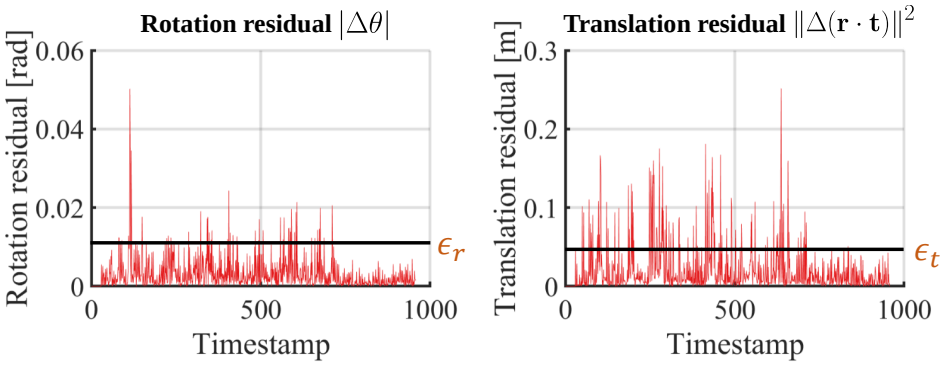


Figure 3.4: An example of the screw motion residuals in rotation and translation of a set of estimated motion. ϵ_r, ϵ_r are the thresholds to filter the outliers.

the performance of the previous motion estimation phase. Fig. 3.4 shows these residuals in a real-world example, where we find the estimated motion is very noisy. Hence, the outliers are filtered if both their rotational and translational residuals are larger than the thresholds: ϵ_r and ϵ_t.

3.4.2.2 Pitch-roll rotation computation

We reformulate (3.2) using the Hamilton quaternion [169] ($\mathbf{q} = [q_w, q_x, q_y, q_z]^\top$) to solve the rotation as below:

$$\begin{aligned} \mathbf{q}_{a_n}^{a_{n-1}} \otimes (\mathbf{q}_b^a)_{yx} = (\mathbf{q}_b^a)_{yx} \otimes \mathbf{q}_{b_n}^{b_{n-1}} &\Rightarrow \left[\mathbf{Q}_1(\mathbf{q}_{a_n}^{a_{n-1}}) - \mathbf{Q}_2(\mathbf{q}_{b_n}^{b_{n-1}})\right] \cdot (\mathbf{q}_b^a)_{yx} \\ &\Rightarrow \ \mathbf{Q}_n^{n-1} \cdot (\mathbf{q}_b^a)_{yx} = \mathbf{0}, \end{aligned} \tag{3.5}$$

where

$$\mathbf{Q}_1(\mathbf{q}) = \begin{bmatrix} q_w\mathbf{I}_3 + [\mathbf{q}_{xyz}]_\times & \mathbf{q}_{xyz} \\ -\mathbf{q}_{xyz}^\top & q_w \end{bmatrix}, \quad \mathbf{Q}_2(\mathbf{q}) = \begin{bmatrix} q_w\mathbf{I}_3 - [\mathbf{q}_{xyz}]_\times & \mathbf{q}_{xyz} \\ -\mathbf{q}_{xyz}^\top & q_w \end{bmatrix}, \tag{3.6}$$

are matrix representations for left and right quaternion multiplication, $[\mathbf{q}_{xyz}]_\times$ is the skew-symmetric matrix of $\mathbf{q}_{xyz} = [q_x, q_y, q_z]^\top$, and $\otimes$ is the quaternion multiplication operator. With N pairs of filtered rotations, we are able to formulate an over-constrained linear system as follows:

$$\begin{bmatrix} \mathbf{Q}_1^0 \\ \vdots \\ \mathbf{Q}_N^{N-1} \end{bmatrix}_{4N\times 4} \cdot (\mathbf{q}_b^a)_{yx} = \mathbf{Q}_N \cdot (\mathbf{q}_b^a)_{yx} = \mathbf{0}. \tag{3.7}$$

Using SVD to decompose $\mathbf{Q}_N = \mathbf{USV}^\top$, $(\mathbf{q}_b^a)_{yx}$ is computed by the weighted sum of

$\mathbf{v}_3$ and $\mathbf{v}_4$:

$$(\mathbf{q}_b^a)_{yx} = \lambda_1 \mathbf{v}_3 + \lambda_2 \mathbf{v}_4, \tag{3.8}$$

where $\mathbf{v}_3$ and $\mathbf{v}_4$ are the last two column vectors of $\mathbf{V}$, and λ_1, λ_2 are two scalars. Therefore, $(\mathbf{q}_b^a)_{yx}$ can be obtained by solving the following equations:

$$x_{(\mathbf{q}_b^a)_{yx}} y_{(\mathbf{q}_b^a)_{yx}} = -z_{(\mathbf{q}_b^a)_{yx}} w_{(\mathbf{q}_b^a)_{yx}} \quad \text{and} \quad x^2_{(\mathbf{q}_b^a)_{yx}} + y^2_{(\mathbf{q}_b^a)_{yx}} + z^2_{(\mathbf{q}_b^a)_{yx}} + w^2_{(\mathbf{q}_b^a)_{yx}} = 1, \tag{3.9}$$

where $x_\mathbf{q}, y_\mathbf{q}, z_\mathbf{q}, w_\mathbf{q}$ are the elements of a quaternion.

3.4.2.3 Yaw rotation and translation computation

Due to the planar motion assumption, the translational offset on $z-$ axis is unobservable. We set $t_z = 0$ and rewrite (3.3) by removing the third row as follows:

$$\mathbf{R}_1 \begin{bmatrix} t_x \\ t_y \end{bmatrix} - \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) \\ \sin(\gamma) & \cos(\gamma) \end{bmatrix} \mathbf{t}_1 = -\mathbf{t}_2, \tag{3.10}$$

where t_x and t_y are unknown translations along $x-$ and $y-$ axes, and γ is the unknown rotation angle around $z-$ axis. $\mathbf{R}_1$ is the 2×2 upper-left submatrix of $(\mathbf{R}_{a_n}^{a_{n-1}} - \mathbf{I}_3)$, $\mathbf{t}_1 = [t_{11}, t_{12}]^\top$ are the first two elements of $\mathbf{R}_b^a \mathbf{t}_{b_n}^{b_{n-1}}$, and $\mathbf{t}_2$ denote the first two elements of $\mathbf{t}_{a_n}^{a_{n-1}}$. We can rewrite (3.11) as a matrix-vector equation:

$$\underbrace{\begin{bmatrix} \mathbf{R}_1 & \mathbf{J} \end{bmatrix}}_{\mathbf{G}_{2\times 4}} \begin{bmatrix} t_x \\ t_y \\ -\cos(\gamma) \\ -\sin(\gamma) \end{bmatrix}, \quad \text{where } \mathbf{J} = \begin{bmatrix} t_{11} & -t_{12} \\ t_{12} & t_{11} \end{bmatrix}. \tag{3.11}$$

We then construct a linear system from (3.11) with the filtered motion:

$$\underbrace{\begin{bmatrix} \mathbf{G}_1^0 \\ \vdots \\ \mathbf{G}_N^{N-1} \end{bmatrix}}_{\mathbf{A}_{2N\times 4}} \underbrace{\begin{bmatrix} t_x \\ t_y \\ -\cos(\gamma) \\ -\sin(\gamma) \end{bmatrix}}_{\mathbf{x}_{4\times 1}} = -\underbrace{\begin{bmatrix} (\mathbf{t}_2)_1^0 \\ \vdots \\ (\mathbf{t}_2)_N^{N-1} \end{bmatrix}}_{\mathbf{b}_{2N\times 1}}, \tag{3.12}$$

where $\mathbf{x}$ is obtained by applying the least-squares approach.

3.4.3 Appearance-Based Refinement

This section combines the coarse initial results with the sensor measurements to refine the extrinsic parameters. Three steps are proposed. Firstly, to recover the unknown t_z, the

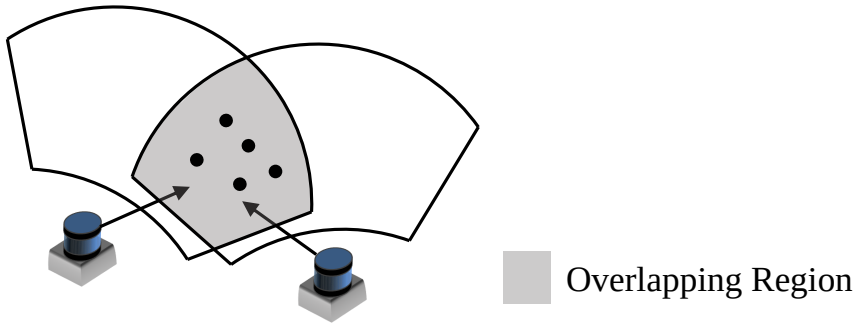


Figure 3.5: The FOV of each LiDAR and overlapping region are visualized.

ground points are utilized. And then we estimate a set of transformations $\mathcal{T}_a^b = \{(\mathbf{T}_b^a)_k\}$ from $\{a\}$ to $\{b\}$ at each timestamp by registering $\mathcal{P}^{a_k}$ and $\mathcal{P}^{b_k}$. To improve the registration accuracy, we also apply an overlap filter to retain the points that lie in the overlapping regions of two sensors.

3.4.3.1 Ground planes alignment

In Section 3.4.1, we have extracted K pairs of ground points $\mathcal{G}^{a_k}, \mathcal{G}^{b_k}$ of the reference and target LiDARs with a height filter. Since the segmented ground points are noisy, we additionally implement the random sample consensus (RANSAC) plane fitting algorithm to reject outliers. Denoting $\mathbf{c}_k^a, \mathbf{c}_k^b$ as the centroids of $\mathcal{G}^{a_k}, \mathcal{G}^{b_k}$ after filtering, we use the mean value of $\left(\mathbf{c}_k^a - \mathbf{R}_b^a \mathbf{c}_k^b\right)_z$ at each timestamp to determine t_z.

3.4.3.2 Overlap Filter

Precise registration between LiDARs is challenging since their overlapping field of view (FOV) is both limited and unknown. To tackle this issue, we employ an overlap filter to retains points that lie within the counterpart LiDARs' FOV, as the gray area depicted in Fig. 3.5. We denote the original point cloud captured by a and b at k as $\mathcal{P}^{a_k}, \mathcal{P}^{b_k}$ respectively. A point $\mathbf{p} \in \mathcal{P}^{b_k}$ can be transformed from $\{b\}$ to $\{a\}$ using the initial $\mathbf{T}_b^a$ as: $\widetilde{\mathbf{p}} = \mathbf{R}_b^a \mathbf{p} + \mathbf{t}_b^a$. We employ the KD-Tree searching method to realize the overlap filter to construct $\mathcal{S}^{a_k}, \mathcal{S}^{b_k}$:

$$\begin{aligned} \mathcal{S}^{a_k} &= \left\{ \forall \mathbf{p}_1 \in \mathcal{P}^{a_k} : d(\mathbf{p}_1, \widetilde{\mathbf{p}}_2) < r, \exists \mathbf{p}_2 \in \mathcal{P}^{b_k} \right\}, \\ \mathcal{S}^{b_k} &= \left\{ \forall \mathbf{p}_2 \in \mathcal{P}^{b_k} : d(\widetilde{\mathbf{p}}_2, \mathbf{p}_1) < r, \exists \mathbf{p}_1 \in \mathcal{P}^{a_k} \right\}, \end{aligned} \tag{3.13}$$

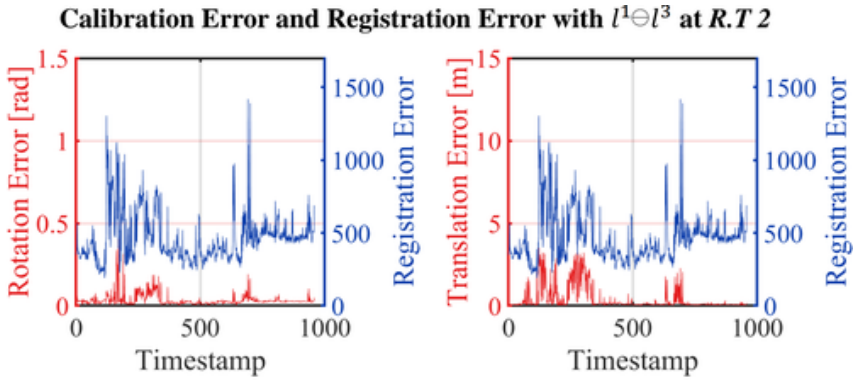


Figure 3.6: Calibration errors and registration errors with $l^1 \ominus l^3$ at *R.T 2*.

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are points, $d(\cdot, \cdot)$ is the Euclidean distance between two points, and r is a threshold. We define the overlap factor as:

$$\Omega_k = \frac{|\mathcal{S}^{a_k}|}{|\mathcal{P}^{a_k}|} \cdot \frac{|\mathcal{S}^{b_k}|}{|\mathcal{P}^{b_k}|}, \tag{3.14}$$

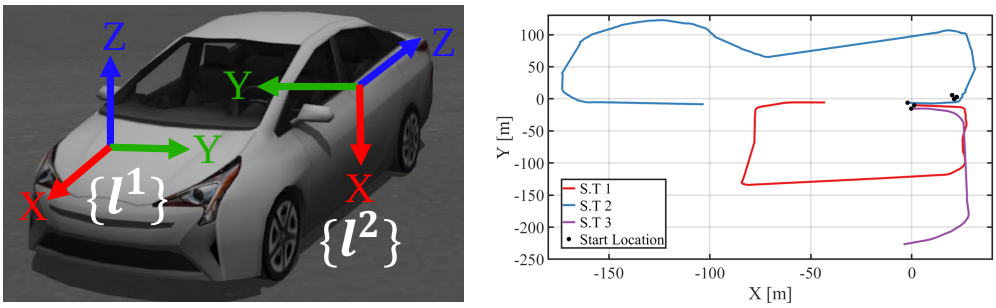
where $|\cdot|$ is the size of a set. The point cloud pairs with $\Omega_k > 0.8$ are selected as the inputs for the following registration step.

3.4.3.3 Registration

To estimate the relative transformations, the point-to-plane Iterative Closest Point (ICP) [5] is used. After registering all the point cloud pairs captured at different timestamps, we obtain a set of transformations $\mathcal{T}_b^a$. But some of these transformations outliers. By analyzing the registration errors and calibration errors computed with the ground truth of each element, we find that their values are correlated. An example of these errors over timestamp is plotted in Fig. 3.6, where the curves of registration error and calibration error have similar trends, especially the positions of the peak. This illustrates that a good registration brings precise extrinsics. Thus, we apply this observation to deal with $\mathcal{T}_b^a$. Only a series of transformations with the minimum registration error are selected as candidates, and the refinement results are computed as their mean values.

3.5 Experiment

In this section, we divide the evaluation into two separate steps. Firstly, the initial calibration experiments are presented with simulated data and real sensor sets. Then we



(a) (b)

Figure 3.7: (a) The simulated platform and (b) the three trajectories which the platform follows. The rotation offset is about $[0, 3.14, 1.57]$ rads in roll, pitch, and yaw respectively. The corresponding translation are about $[-2.5, 1.5, 0]$ meters along $x-, y-$, and $z-$ axes respectively.

Table 3.1: The initial calibration results with simulated data.

σ^2	Trajectory	Rotation Error [rad]		Translation Error [m]	
		Kabsch	Proposed	Kabsch	Proposed
σ_1^2	*S.T 1*	0.10	**0.01**	**0.22**	0.28
	S.T 2	0.80	**0.00**	0.66	**0.28**
	S.T 3	0.08	**0.01**	0.80	**0.48**
σ_2^2	*S.T 1*	1.37	**0.07**	2.13	**1.25**
	S.T 2	0.94	**0.04**	1.80	**1.20**
	S.T 3	1.17	**0.02**	1.66	**1.44**

test the refinement on real sensor data and demonstrate that the initial results can be improved using appearance cues.

3.5.1 Implementation Details

We use the ICP library [140] to process point clouds. With empirically setting parameters: $\epsilon_r = 0.01$, $\epsilon_t = 0.01$ and $r = 10$, our method can obtain promising results. Since the success of motion-based calibration highly depends on the quality of motion which a vehicle undergoes, we design several paths with different rotations and scales to test our proposed algorithm. These paths include three simulated trajectories (*S.T 1*-*S.T 3*) on simulation and two real trajectories (*R.T 1*-*R.T 2*) on real sensor data. In the experiments, two platforms with different sensor setups are used.

3.5.1.1 Simulation

The simulation software[3] is publicly available. For testing, we manually mount two sensors at different positions on the vehicle platform, as shown in Fig. 3.7a (left). The rotation offset between them is approximately $[0, 3.14, 1.57]$ rad in roll, pitch, and yaw, respectively. The corresponding translation are approximately $[-2.5, 1.5, 0]$ meters along $x-, y-$, and $z-$ axes respectively. We can thus acquire the positions of these sensors in the form of ground truth at 5 Hz. The refinement is not tested in simulation because this platform does not provide stable point clouds without time distortion.

3.5.1.2 Real Sensor

While our approach is not limited to a particular number of sensors, we are interested in calibrating between the reference LiDAR and two target LiDARs based on our platform. As shown in Fig. 3.1, three 16-beam RS-LiDARs[4] are rigidly mounted on the vehicle. The setup of this multi-LiDAR system has significant transformation among sensors. Especially, l^3 is mounted with approximately 180 degrees rotation offset in yaw. In later sections, we use $l^1 \ominus l^i$ to represent the configuration between l^1 and l^i. Since we do not know the precise extrinsic parameters, we use the parameters (shown in Table 3.2) provided by the manufacturer to evaluate our proposed algorithm.

In evaluation, the error in rotation is measured by the angle difference between the ground truth and the estimated rotation, which is calculated as $e_r = \| \log(\mathbf{R}_{\text{gt}}\mathbf{R}_{\text{est}}^{-1})^{\vee} \|_2$. Similarly, the error in translation is computed using vector subtraction as $e_t = \|\mathbf{t}_{\text{gt}} - \mathbf{t}_{\text{est}}\|_2$. The translation error on $z-$ axis will not be counted of the initialization results because of the planar movement assumption.

3.5.2 Performance of Initialization

We take the modified Kabsch algorithm [176] that operates at matrix representation for comparison.

[3] `https://github.com/osrf/car_demo/`

[4] `https://www.robosense.ai/rslidar/rs-lidar-16`

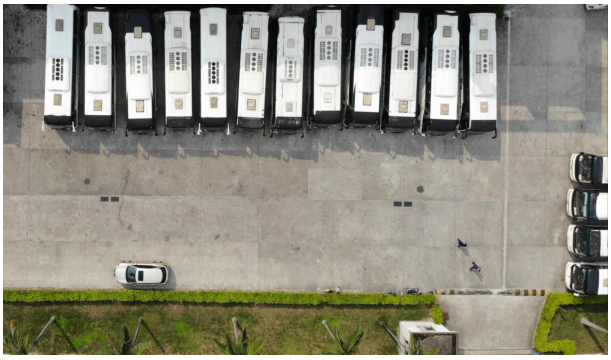


Figure 3.8: The testing environment for our calibration experiments.

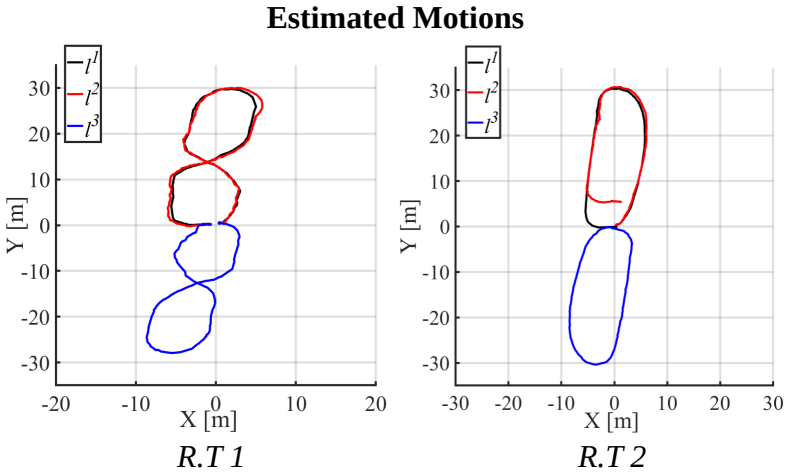


Figure 3.9: The black, red, and blue lines indicate the estimated motionof l_1, l_2, l_3 respectively at *R.T 1-R.T 2*.

3.5.2.1 Simulation

The simulated trajectories *(S.T 1-S.T 3)* are visualized in Fig. 3.7b, where the third one is considered as the most challenging one since it has few rotations. To test the robustness of our proposed algorithm, the sensor's motion are added with zero-mean Gaussian noise with the variance σ^2. σ^2 is set to two values: $\sigma_1^2 = 0.0001$ and $\sigma_2^2 = 0.001$ for evaluation. For each value, all simulated motion is tested. The calibration results are shown in Table 3.1. The proposed algorithm can successfully initialize the rotation offset with low error and outperform the Kabsch algorithm in most of the cases. The translation error with σ_2^2 is larger than 1m, meaning that our initialization approach is not robust in noisy data.

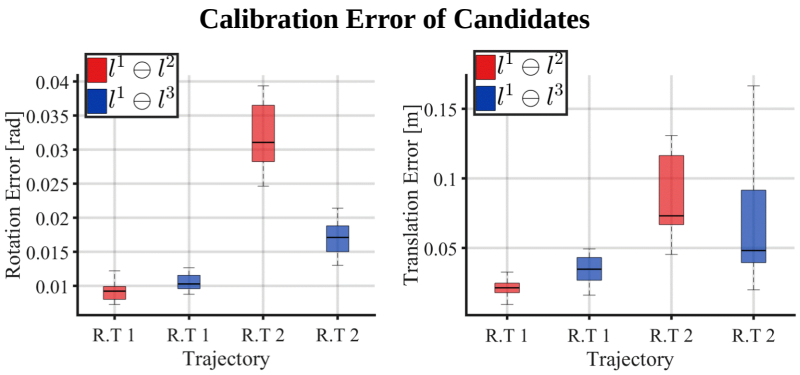


Figure 3.10: The calibration errors in each case of the candidate. The means of them are small at *R.T 1*, but large at *R.T 2*.

Table 3.2: The calibration ground truth of our multi-lidar system.

Configuration	Rotation $[rad]$			Translation $[m]$		
	x	y	z	x	y	z
$l^1 \ominus l^2$	0.01	0.08	0.03	0.42	0.00	-1.26
$l^1 \ominus l^3$	-0.02	0.01	-3.11	-2.11	0.06	-1.18

3.5.2.2 Real Sensor

We carry out a real sensor experiment to validate the proposed method. Two trajectories (*R.T 1*-*R.T 2*) are designed in an urban environment (shown in 3.8), and we drive the vehicle to follow them. After estimating the individual motion of each LiDAR, we use the results to initialize the calibration. The estimated motion is depicted in Fig. 3.9, where we can observe that the calculated trajectory of l^2 at *R.T 2* drifts. The initialization results are presented in Table 3.3. Our method performs well in recovering the rotation offset ($< 0.15rad$), but fail in calculating the translation offset ($> 0.5m$) in all cases. With the above results in simulation and real-world environments, we can conclude that the initialization phase can provide coarse estimates to the extrinsic parameters. For precise results, an additional refinement step is required.

3.5.3 Performance of Refinement

In our experiments, we select 10 transformations from $\mathcal{T}_b^a$ as the candidates for the optimal refinement results. We plot their calibration errors in each case in Fig. 3.10. The detailed calibration results are shown in Table 3.4, where all the rotation and translation errors are less than $0.04rad$ and $0.1m$, respectively. Compared with the result in Table

Table 3.3: The initialization results.

Confguration	Trajectory	Rotation Error $[rad]$		Translation Error $[m]$	
		Kabsch	Proposed	Kabsch	Proposed
$l^1 \ominus l^2$	*R.T 1*	0.94	**0.06**	1.60	**0.47**
	R.T 2	0.73	**0.03**	4.42	**1.95**
$l^1 \ominus l^3$	*R.T 1*	1.29	**0.14**	**1.23**	1.26
	R.T 2	0.60	**0.08**	**1.36**	2.04

Table 3.4: The refinement results.

Configuration Trajectory		$l^1 \ominus l^2$ *R.T 1*	$l^1 \ominus l^2$ *R.T 2*	$l^1 \ominus l^3$ *R.T 1*	$l^1 \ominus l^3$ *R.T 2*
Rotation $[rad]$	x	0.01	0.01	-0.02	-0.02
	y	0.08	0.07	0.02	0.00
	z	0.04	0.04	-3.14	-3.13
	error	**0.01**	0.03	**0.01**	0.02
Translation $[m]$	x	0.43	0.46	-2.13	-2.07
	y	0.00	-0.01	0.09	0.08
	z	-1.26	-1.27	-1.18	-1.20
	error	**0.01**	0.08	0.03	0.04

3.3, the refinement phase can improve the estimated parameters.

3.5.4 Discussion

Since the proposed calibration method achieve accurate results, it do not perform well in the below cases. Firstly, the initialization highly relies on the accuracy of estimated motion, especially when the translational offset is imprecise. Secondly, We assume that LiDARs' views should have overlapping regions, Finally, the registration between point clouds may fail in several feature-less environments.

3.6 Conclusion and Future Work

In this thesis, we presented a novel system for automatically calibrating of a multi-LiDAR system without any extra sensors, calibration target, or prior knowledge about surroundings. Our approach makes use of the complementary strengths of motion-based and appearance-based calibration methods. The individual motion of each LiDAR is estimated by an odometry algorithm. These poses are then utilized to initialize the

extrinsic parameters. Finally, the results are refined by exploiting appearance cues in sensors' overlap. The performance of our method is demonstrated through a series of simulated and real-world experiments with reliable and accurate calibration results. There are several possible extensions to this work: (1) online, active multi-LiDAR calibration, (2) releasing the overlapping requirement, and (2) applications including localization and 3D object detection based on a multi-LiDAR system.

Part II

Localization and Mapping

CHAPTER 4

ROBUST ODOMETRY AND MAPPING FOR MULTI-LIDAR SYSTEMS WITH ONLINE EXTRINSIC CALIBRATION

Combining multiple LiDARs enables a robot to maximize its perceptual awareness of environments and obtain sufficient measurements, which is promising for simultaneous localization and mapping (SLAM). This thesis proposes a system to achieve robust and simultaneous extrinsic calibration, odometry, and mapping for multiple LiDARs. Our approach starts with measurement preprocessing to extract edge and planar features from raw measurements. After a motion and extrinsic initialization procedure, a sliding window-based multi-LiDAR odometry runs onboard to estimate poses with an online calibration refinement and convergence identification. We further develop a mapping algorithm to construct a global map and optimize poses with sufficient features together with a method to capture and reduce data uncertainty. We validate our approach's performance with extensive experiments on ten sequences (4.60km total length) for the calibration and SLAM and compare them against the state-of-the-art. We demonstrate that the proposed work is a complete, robust, and extensible system for various multi-LiDAR setups. The source code, datasets, and demonstrations are available at: `https://ram-lab.com/file/site/m-loam`.

4.1 Introduction

4.1.1 Motivation

Simultaneous Localization and Mapping (SLAM) is essential to a wide range of applications, such as scene reconstruction, robotic exploration, and autonomous driving [6, 19, 180]. Approaches that use only a LiDAR have attracted much attention from the research community due to their accuracy and reliability in range measurements. However, LiDAR-based methods commonly suffer from data sparsity and limited vertical field of view (FOV) in real-world applications [79]. For instance, LiDARs' points distribute

loosely, which induces a mass of empty regions between two nearby scans. This characteristic usually causes state estimation to degenerate in structureless environments, such as narrow corridors and stairs [197]. Recently, owing to the decreasing price of sensors, we have seen a growing trend of deploying multi-LiDAR systems on practical robotic platforms [3, 9, 56, 89, 116, 173]. Compared with a single-LiDAR setup, the primary improvement of multi-LiDAR systems is the significant enhancement on the sensing range and density of measurements. This benefit is practically useful for self-driving cars since we have to address the critical blind spots created by the vehicle body. Thus, we consider multi-LiDAR systems in this thesis.

4.1.2 Challenges

Despite their great advantages in environmental perception, a number of issues affect the development of SLAM using a multi-LiDAR setup.

4.1.2.1 Precise and Flexible Extrinsic Calibration

Recovering the multi-LiDAR transformations for a new robotic platform is complicated. In many cases, professional users have to calibrate sensors in human-made surroundings [81] carefully. This requirement increases the cost to deploy and maintain a multi-LiDAR system for field robots.

It is desirable that the system can *self-calibrate* the extrinsics in various environments online. As shown in [106, 145, 196], benefiting from the simultaneous estimation of extrinsics and ego-motion, the working scope of visual-inertial systems has been expanded to drones and vessels in outdoor scenes. These approaches continuously perform calibration during the mission to guarantee the objective function to be always 'optimal'. However, this process typically requires environmental or motion constraints to be fully observable. Otherwise, the resulting extrinsics may become suboptimal or unreliable. Thus, we have to fix the extrinsics if they are accurate, which creates a demand for online calibration with a convergence identification. In order to cope with the centimeter-level calibration error and unexpected changes on extrinsics over time, it is also beneficial to model the extrinsic perturbation for multi-LiDAR systems.

4.1.2.2 Low Pose Drift

To provide accurate poses in real time, state-of-the-art (SOTA) LiDAR-based methods [110, 163, 205] commonly solve SLAM by two algorithms: *odometry* and *mapping*. These algorithms are designed to estimate poses in a coarse-to-fine fashion. Based on the original odometry algorithm, an approach that fully exploits multi-LiDAR measurements within a local window is required. The increasing constraints help to prevent degeneracy or failure of frame-to-frame registration. The subsequent mapping algorithm runs at a relatively low frequency and is given plenty of feature points and many iterations for better results. However, as we identify in Section 4.8, many SOTA approaches neglect the fact that uncertain points in the global map limit the accuracy. To minimize this adverse effect, we must develop a method to capture map points' uncertainties and reject outliers.

4.1.3 Contributions

To tackle these challenges, we propose M-LOAM, a robust system for multi-LiDAR extrinsic calibration, real-time odometry, and mapping. Without manual intervention, our system can start with several extrinsic-uncalibrated LiDARs, automatically calibrate their extrinsics, and provide accurate poses as well as a globally consistent map. Our previous work [197] proposed sliding window-based odometry to fuse LiDAR points with high-frequency IMU measurements. That framework inspires this thesis, where we try to solve the problem of multi-LiDAR fusion. In addition, we introduce a motion-based approach [79] to initialize extrinsics, and employ the tools proposed in [7] to represent uncertainties. Our design of M-LOAM presents the following *contributions*:

1. Automatic initialization that computes all critical states, including motion between consecutive frames as well as extrinsics for subsequent phases. It can start at arbitrary positions without any prior knowledge of the mechanical configuration or calibration objects (Section 4.6).

2. Online self-calibration with a general convergence criterion is executed simultaneously with the odometry. It has the capability to monitor the convergence and trigger termination in a fully unsupervised manner (Section 4.7.2).

3. Sliding window-based odometry that fully exploits information from multiple LiDARs. This implementation can be explained as small-scale frame-to-map registra-

tion, which further reduces the drift accumulated by the consecutive frame-to-frame odometry (Section 4.7.3).

4. Mapping with a two-stage approach that captures and propagates uncertainties from sensor noise, degenerate pose estimation, and extrinsic perturbation. This approach enables the mapping process with an awareness of uncertainty and helps us to maintain the consistency of a global map as well as boost the robustness of a system for long-duration navigation tasks (Section 4.8).

To the best of our knowledge, M-LOAM is the first complete solution to multi-LiDAR calibration and SLAM. The system is evaluated under extensive experiments on both handheld devices and autonomous vehicles, covering various scenarios from indoor offices to outdoor urban roads, and outperforms the SOTA LiDAR-based methods. Regarding the calibration on diverse platforms, our approach achieves an extrinsic accuracy of centimeters in translation and deci-degrees in rotation. For the SLAM in different scales, M-LOAM has been successfully applied to provide accurate poses and map results. To benefit the research community, we publicly release our code, implementation details, and the multi-LiDAR datasets.

4.2 Related Work

Scholarly works on SLAM and extrinsic calibration are extensive. In this section, we briefly review relevant results on LiDAR-based SLAM and online calibration methods for multi-sensor systems.

4.2.1 LiDAR-Based SLAM

Generally, many SOTA are developed from the iterative closest point (ICP) algorithm [124, 140, 161, 162], into which the methods including measurement preprocessing, degeneracy prediction, and sensor fusion are incorporated. These works have pushed the current LiDAR-based systems to become fast, robust, and feasible to large-scale environments.

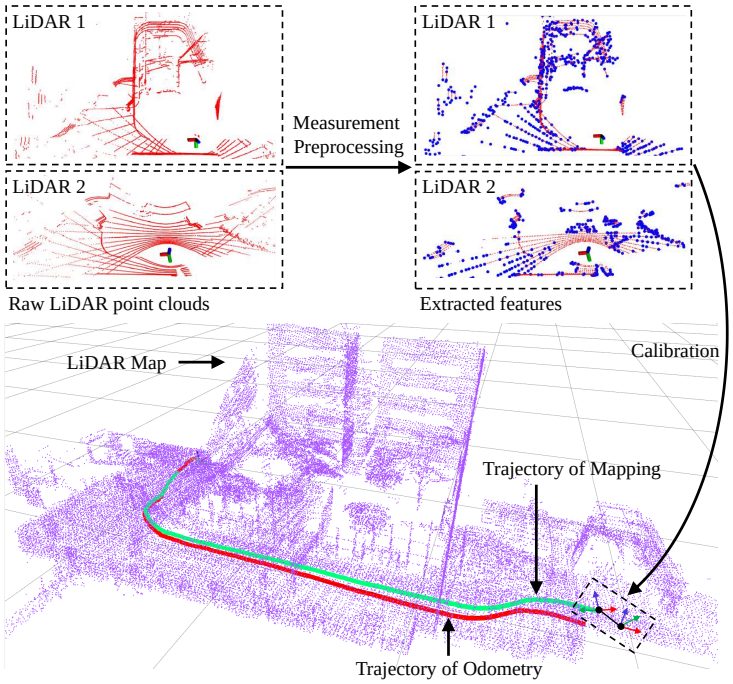


Figure 4.1: We visualize the immediate results of M-LOAM. The raw point clouds perceived by different LiDARs are denoised and extracted with edge (blue dots) and planar (red dots) features, which are shown at the top-right position. The proposed online calibration is performed to obtain accurate extrinsics. After that, the odometry and mapping algorithms use these features to estimate poses. The trajectory of the mapping (green) is more accurate than that of the odometry (red).

4.2.1.1 Measurement Preprocessing

As the front-end of a system, the measurement preprocessing encodes point clouds into a compact representation. We categorize related algorithms into either dense or sparse methods. As a typical dense method, SuMa [10] demonstrates the advantages of utilizing surfel-based maps for registration and loop detection. Its extended version [31] incorporates semantic constraints into the original cost function. However, the process of matching dense pixel-to-pixel correspondences in these methods is time-consuming and needs a GPU. These approaches are not applicable to our cases since we have to frequently perform the registration.

Sparse methods prefer to extract geometric features from raw measurements and are thus supposed to have real-time performance. Grant et al. [59] proposed a plane-based registration, while Velas et al. [182] represented point clouds as collar line segments. Compared to them, LOAM [205] has fewer assumptions about sensors and surroundings. It selects distinct points from both edge lines and local planar patches. Recently, several following methods have employed ground planes [134, 163], visual detection [28], proba-

bilistic grid maps [77], good features [84], or directly used dense scanners [15, 110, 138] to improve the performance of sparse approaches in noisy or structure-less environments. To limit the computation time when more LiDARs are involved, we extract the sparse edge and planar features like LOAM. But differently, we represent their residuals in a more concise and unified way.

4.2.1.2 Degeneracy in State Estimation

The geometric constraints of features are formulated as an ego-motion optimization problem. Different methods have been proposed to tackle the degeneracy issue. Zhang et al. [204] defined a *factor*, which is equal to the minimum eigenvalue of information matrices, to determine the system degeneracy. They also proposed a technique called *solution remapping* to update variables in well-conditioned directions. This technique was further applied to tasks including localization, registration, pose graph optimization, and inspection of sensor failure [69, 207, 214]. To quantify a nonlinear system's observability, Rong et al. [153] computed the condition number of the empirical observability Gramian matrix. Since our problem linearizes the objective function as done in Zhang et al. [204], we also introduce *solution remapping* to update states. Additionally, our online calibration method employs the *degeneracy factor* as a quantitative metric of the extrinsic calibration.

4.2.1.3 Multi-Sensor Fusion

Utilizing multiple sensors to improve the motion-tracking performance of single-LiDAR odometry is promising. Most existing works on LiDAR-based fusion combine visual or inertial measurements. The simplest way to deal with multi-modal measurements is loosely coupled fusion, where each sensor's pose is estimated independently. For example, LiDAR-IMU fusion is usually done by the extended Kalman filter (EKF) [122, 183, 211]. Tightly coupled algorithms that optimize sensors' poses by jointly exploiting all measurements have become increasingly prevalent in the community. They are usually done by either the EKF [65, 143, 220] or batch optimization [99, 100, 121, 197, 215]. Besides multi-modal sensor fusion, another approach is to explore the fusion of multiple LiDARs (uni-model), which is still an open problem.

Multiple LiDARs improve a system by maximizing the sensing coverage against extreme occlusion. Inspired by the success of tightly coupled algorithms, we achieve a

sliding window estimator to optimize states of multiple LiDARs. This fusion mode has been shown to have great advantages in multi-LiDAR systems.

4.2.2 Multi-Sensor Calibration

Precise extrinsic calibration is of paramount importance to any multi-sensor system. Traditional methods [32, 64, 81, 191] have to run an ad-hoc calibration procedure before a mission. This tedious process needs to be repeated whenever there is slight perturbation on the structure. A more flexible solution to estimate these parameters is combined SLAM-based techniques. Here, extrinsics are treated as one of the state variables and optimized along with the sensors' poses. This scheme is also applicable to some non-stationary parameters, such as robot kinematics, IMU biases, and time offsets.

Kummerle et al. [95] pioneered a hyper-graph optimization framework to calibrate an onboard laser scanner with wheel encoders. Their experiments reveal that online correction of parameters leads to consistent and accurate results. Teichman et al. [179], meanwhile, proposed an iterative SLAM-fitting pipeline to resolve the distortion of two RGB-D cameras. To recover spatial offsets of a multi-camera system, Heng et al. [67] formulated the problem as a bundle adjustment, while Ouyang et al. [133] employed the Ackermann steering model of vehicles to constrain extrinsics. As present in [98, 100, 147, 149], the online estimation of time offsets among sensors is crucial to IMU-centric systems. Qin et al. [147] utilized positions of visual features to formulate the temporal calibration problem, while Qiu et al. [149] proposed a more general solution to calibrate heterogeneous sensors by analyzing motion correlation.

This thesis implicitly synchronizes the time of multiple LiDARs with an hardware-based external clock and explicitly focuses on the extrinsic calibration. Our approach consists of an online procedure to achieve flexible multi-LiDAR extrinsic calibration. To monitor the convergence of estimated extrinsics, we propose a general criterion. In addition, we model the extrinsic perturbation to reduce its negative effect for long-term navigation tasks.

4.3 Problem Statement

We formulate M-LOAM in terms of the Maximum Likelihood Estimation (MLE). The MLE leads to a nonlinear optimization problem where the inverse of the Gaussian covariances weights the residual functions. Before delving into details of M-LOAM, we first introduce some basic concepts. Section 4.3.1 presents notations. Section 4.3.2 discusses the MLE, and Section 4.3.3 describes suitable models to represent uncertain measurements in $\mathbb{R}^3$ and transformations in $SE(3)$. Finally, Section 4.3.4 briefly presents the implementation of the MLE with approximate Gaussian noise in M-LOAM.

4.3.1 Notations and Definitions

The nomenclature is shown in Table 4.1. We consider a system that consists of one primary LiDAR and multiple auxiliary LiDARs. The primary LiDAR defines the base frame, and we use $()^{l1}/()^{b}$ to indicate it. The frames of the auxiliary LiDARs are denoted by $()^{l^{i,i>1}}$. We denote $\mathcal{F}$ as the set of available features extracted from the LiDARs' raw measurements. Each feature is represented as a point in 3D space: $\mathbf{p} = [x, y, z]^\top$. The state vector, composed of translational and rotational parts, is denoted by $\mathbf{x} = [\mathbf{t}, \mathbf{q}]$ where $\mathbf{t}$ is a 3×1 vector, and $\mathbf{q}$ is the Hamilton quaternion. But in the case that we need to rotate a vector, we use the 3×3 rotation matrix $\mathbf{R}$ in the *Lie group SO(3)*. We can convert $\mathbf{q}$ into $\mathbf{R}$ with the Rodrigues formula [169]. Section 4.8 associates uncertainty with poses on the vector space, where we use the 4×4 transformation matrix $\mathbf{T}$ in the *Lie group SE(3)* to represent a pose. A rotation matrix and translation vector can be associated with a transformation matrix according to

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{4.1}$$

4.3.2 Maximum Likelihood Estimation

We formulate the pose and extrinsic estimation of a multi-LiDAR system as an MLE problem [76]:

$$\hat{\mathbf{x}}_k = \arg\max_{\mathbf{x}_k} \; p(\mathcal{F}_k|\mathbf{x}_k) = \arg\min_{\mathbf{x}_k} \; f(\mathbf{x_k}, \mathcal{F}_k), \tag{4.2}$$

where $\mathcal{F}_k$ represents the available features at the k^{th} frame, $\mathbf{x}_k$ is the state to be optimized, and $f(\cdot)$ is the objective function. Assuming the measurement model to be subjected to

Gaussian noise [6], problem (4.2) is solved as a nonlinear least-squares (NLS) problem:

$$\hat{\mathbf{x}}_k = \arg\min_{\mathbf{x}_k} \sum_{i=1}^{m} \rho\Big(\big|\big|\mathbf{r}(\mathbf{x}_k, \mathbf{p}_{ki})\big|\big|^2_{\mathbf{\Sigma}_i}\Big), \tag{4.3}$$

where $\rho(\cdot)$ is the robust Huber loss to handle outliers [14], $\mathbf{r}(\cdot)$ is the residual function, and $\mathbf{\Sigma}_i$ is the covariance matrix. Iterative methods such as Gauss-Newton or Levenberg-Marquardt can offen be used to solve this NLS problem. These methods locally linearize the objective function by computing the Jacobian w.r.t. $\mathbf{x}_k$ as $\mathbf{J} = \partial f / \partial \mathbf{x}_k$. Given an initial guess, $\mathbf{x}_k$ is iteratively optimized by usage of $\mathbf{J}$ until convergence to find an optimum. At the final iteration, the least-squares covariance of the state is calculated as $\mathbf{\Xi} = \mathbf{\Lambda}^{-1}$ [24], where $\mathbf{\Lambda} = \mathbf{J}^\top \mathbf{J}$ is called the *information matrix*.

4.3.3 Uncertainty Representation

We employ the notations in [7] to represent data uncertainty. We first represent a noisy LiDAR point as

$$\mathbf{p} = \bar{\mathbf{p}} + \boldsymbol{\zeta}, \quad \boldsymbol{\zeta} \sim \mathcal{N}(\mathbf{0}, \mathbf{Z}), \tag{4.4}$$

where $\bar{\mathbf{p}}$ is a noise-free vector, $\boldsymbol{\zeta} \in \mathbb{R}^3$ is a small Gaussian perturbation variable with zero mean, and $\mathbf{Z}$ is a noise covariance of LiDAR measurements. To make (4.4) compatible with transformation matrices (e.g., $\mathbf{p}'_h = \mathbf{T}\mathbf{p}_h$), we also express it with 4×1 homogeneous coordinates:

$$\mathbf{p}_h = \begin{bmatrix} \bar{\mathbf{p}} \\ 1 \end{bmatrix} + \mathbf{D}\boldsymbol{\zeta} = \bar{\mathbf{p}}_h + \mathbf{D}\boldsymbol{\zeta}, \quad \boldsymbol{\zeta} \sim \mathcal{N}(\mathbf{0}, \mathbf{Z}), \tag{4.5}$$

where $\mathbf{D}$ is a matrix that converts a 3×1 vector into homogeneous coordinates. As investigated in [139], the LiDARs' depth measurement error (also called sensor noise) is primarily caused by the target distances. $\mathbf{Z}$ is simply set as a constant matrix.[1] We then define a random variable in $SE(3)$ with a small perturbation according to[2]

$$\mathbf{T} = \exp(\boldsymbol{\xi}^\wedge)\bar{\mathbf{T}}, \quad \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Xi}), \tag{4.6}$$

where $\bar{\mathbf{T}}$ is a noise-free transformation and $\boldsymbol{\xi} \in \mathbb{R}^6$ is the small perturbation variable with covariance $\mathbf{\Xi}$. This representation allows us to store the mean transformation as $\bar{\mathbf{T}}$ and

[1] $\mathbf{Z} = \mathrm{diag}(\sigma_x^2, \sigma_y^2, \sigma_z^2)$, where $\sigma_x, \sigma_y, \sigma_z$ are standard deviations of LiDARs' depth noise along different axes. We refer to manuals to obtain them for a specific LiDAR.

[2] The $(\cdot)^\wedge$ operator turns $\boldsymbol{\xi}$ into a member of the *Lie algebra* $\mathfrak{se}(3)$. The *exponential map* associates an element of $\mathfrak{se}(3)$ to a transformation matrix in $SE(3)$. Similarly, we can also use $(\cdot)^\wedge$ and $\exp(\cdot)$ to associate 3×1 vector $\boldsymbol{\phi}$ with a rotation matrix in $SO(3)$. [7] provides detailed expressions.

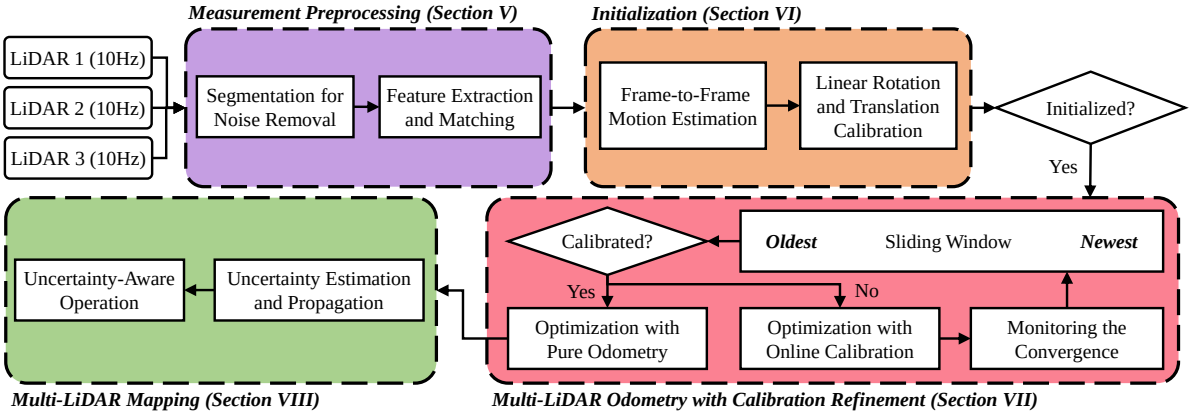


Figure 4.2: The block diagram illustrating the full pipeline of the proposed M-LOAM system. The system starts with measurement preprocessing (Section 4.5). The initialization module (Section 4.6) initializes values for the subsequent nonlinear optimization-based multi-LiDAR odometry with calibration refinement (Section 4.7). According to the convergence of calibration, the optimization is divided into two subtasks: online calibration and pure odometry. Finally, the uncertainty-aware multi-LiDAR mapping (Section 4.8) maintains a globally consistent map to decrease pose drift and noisy points.

use $\boldsymbol{\xi}$ for perturbation on the vector space. We consider that $\boldsymbol{\Xi}$ consists of two sources of errors:

- **Degenerate Pose Estimation** arises from cases such as lack of geometrical structures in poorly constrained environments [204]. It typically makes poses uncertain in their degenerate directions [17, 24]. Existing works resort to model-based and learning-based [96] methods to estimate pose covariances in the context of ICP.
- **Extrinsic Perturbation** always exists due to calibration errors [81], or vibration, impact, and temperature drift during long-term operation. Wide baseline sensors such as stereo cameras, especially, suffer even more extrinsic deviations than standard sensors. Such perturbation is detrimental to the measurement accuracy [78, 123] of multi-sensor systems but is hard to measure.

The computation of $\boldsymbol{\Xi}$ is detailed in Section 4.8.

4.3.4 MLE with Approximate Gaussian Noise in M-LOAM

We extend the MLE to design multiple M-LOAM estimators to solve the robot poses and extrinsics in a coarse-to-fine fashion. The most important consideration is the approximation of the Gaussian noise covariance $\boldsymbol{\Sigma}$ to realistic measurement models. Based

Table 4.1: Nomenclature

Notation	Explanation
	Coordinate System and Pose Representation
$()^w, ()^b, ()^{l^i}$	Frame of the world, base sensor, and i^{th} LiDAR
$()^{l^i_k}$	Frame of the i^{th} LiDAR while taking the k^{th} point cloud
I	Number of LiDARs in a multi-LiDAR system
$\mathbf{p}$, $\mathbf{p}_h$	3D Point in $\mathbb{R}^3$ and homogeneous coordinates
$\mathbf{x}$	State vector
$\mathbf{t}$	Translation vector in $\mathbb{R}^3$
$\mathbf{q}$	Hamilton quaternion
$\mathbf{R}$	Rotation matrix in the *Lie group* $SO(3)$
$\mathbf{T}$	Transformation matrix in the *Lie group* $SE(3)$
$\boldsymbol{\xi}, \boldsymbol{\zeta}, \boldsymbol{\theta}$	Gaussian noise variable
$\boldsymbol{\Sigma}$	Covariance matrix of residuals
$\mathbf{Z}$	Covariance matrix of LiDAR depth measurement noise
$\boldsymbol{\Xi}$	Covariance matrix of pose noise
	Odometry and Mapping
$\mathcal{F}$	Features extracted from raw point clouds
$\mathcal{E}, \mathcal{H}$	Edge and planar subset of extracted features
L, Π	Edge line and planar patch
$[\mathbf{w}, d]$	Coefficient vector of a planar patch
$\mathcal{M}$	Local map
$\mathcal{G}$	Global map
λ	Degeneracy factor

on the discussion in Section 4.3.3, we identify that three sources of error may make landmarks uncertain: sensor noise, degenerate pose estimation, and extrinsic perturbation. The frame-to-frame motion estimation (Section 4.6.1) is approximately subjected to the sensor noise. The tightly coupled odometry (Section 4.7.3) establishes a local map for the pose optimization, which should propagate pose uncertainties on each map point. Nevertheless, this operation is often time-consuming (around $10ms - 20ms$) if more LiDARs and sliding windows are involved. To guarantee the real-time odometry, we do not consider the pose uncertainty here. We simply set $\boldsymbol{\Sigma} = \mathbf{Z}$ as the covariance of residuals in these modules. In mapping, we are given sufficient time for an accurate pose and a global map. Therefore, we consider all uncertainty models. Section 4.8 explains how the pose uncertainty significantly affects the mapping precision and $\boldsymbol{\Sigma}$ is propagated.

4.4 System Overview

We make three assumptions to simplify the system design:

- LiDARs are synchronized, meaning that the latency of the LiDARs' measurements is almost zero.
- The platform undergoes sufficient rotational and translational motion in the period of calibration initialization.
- The local map of the primary LiDAR should share an overlapping FOV with auxiliary LiDARs for feature matching in refinement to shorten the calibration phase. This can be achieved by moving the robot.

Fig. 4.2 presents the structure of M-LOAM. The system starts with measurement preprocessing (Section 4.5), in which edge and planar features are extracted and tracked from the denoised point clouds. The initialization module (Section 4.6) provides all necessary values, including poses and extrinsics, for bootstrapping the subsequent nonlinear optimization-based M-LO. The M-LO fuses multi-LiDAR measurements to optimize the odometry and extrinsics in a sliding window. Finally, the probabilistic mapping module (Section 4.8) constructs a global map with sufficient features to eliminate the drift. The M-LO and mapping run concurrently in separated threads.

4.5 Measurement Preprocessing

We implement three sequential steps to process LiDARs' raw measurements. We first segment point clouds into many clusters to remove noisy objects, and then extract edge and planar features. To associate features between consecutive frames, we match a series of correspondences. In this section, each LiDAR is handled independently.

4.5.1 Segmentation for Noise Removal

With knowing the vertical scanning angles of a LiDAR, we can project the raw point cloud onto a range image without data loss. In the image, each valid point is represented by a pixel. The pixel value records the Euclidean distance from a point to the origin. We

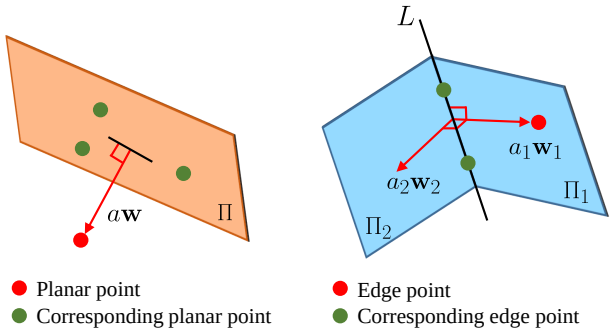


Figure 4.3: The planar and edge residuals. The red dot indicates the reference point and the green dots are its corresponding points.

apply the segmentation method proposed in [13] to group pixels into many clusters. We assume that two neighboring points in the horizontal or vertical direction belong to the same object if their connected line is roughly perpendicular ($> 60deg$) to the laser beam. We employ the breadth-first search algorithm to traverse all pixels, ensuring a constant time complexity. We discard small clusters since they may offer unreliable constraints in optimization.

4.5.2 Feature Extraction and Matching

We are interested in extracting the general edge and planar features. We follow [205] to select a set of feature points from measurements according to their curvatures. The set of extracted features $\mathcal{F}$ consists of two subsets: edge subset (high curve) $\mathcal{E}$ and planar subset (low curve) $\mathcal{H}$. Both $\mathcal{E}$ and $\mathcal{H}$ consist of a portion of features which are the most representative. We further collect edge points from $\mathcal{E}$ with the highest curvature and planar points from $\mathcal{H}$ with the lowest curvature. These points form two new sets: $\hat{\mathcal{E}}$ and $\hat{\mathcal{H}}$. The next step is to determine feature correspondences between two consecutive frames, $()^{l_{k-1}^i} \rightarrow ()^{l_k^i}$, to construct geometric constraints. For each point in $\hat{\mathcal{E}}^{l_k^i}$, two closest neighbors from $\mathcal{E}^{l_{k-1}^i}$ are selected as the edge correspondences. For a point in $\hat{\mathcal{H}}^{l_k^i}$, the three closest points to $\mathcal{H}^{l_{k-1}^i}$ that form a plane are selected as the planar correspondences.

4.6 Initialization

Optimizing the states of multiple LiDARs is highly nonlinear and needs initial guesses. This section introduces our motion and extrinsic initialization approach, which does not require any prior mechanical configuration of the sensor suite. It also does not involve

any manual effort, making it particularly useful for autonomous robots.

4.6.1 Scan-Based Motion Estimation

With found correspondences between two successive frames of each LiDAR, we estimate the frame-to-frame transformation by minimizing residual errors of all features. As illustrated in Fig. 4.3, the residuals are formulated by both edge and planar correspondences. Let $\mathbf{x}_k$ be the relative transformation between two scans of a LiDAR at the k^{th} frame. Regarding planar features, for a point $\mathbf{p} \in \hat{\mathcal{H}}^{l_k^i}$, if Π is the corresponding planar patch, the planar residual is computed as

$$\mathbf{r}_{\mathcal{H}}(\mathbf{x}_k, \mathbf{p}, \Pi) = a\mathbf{w}, \qquad a = \mathbf{w}^\top(\mathbf{R}_k\mathbf{p} + \mathbf{t}_k) + d, \tag{4.7}$$

where a is the point-to-plane Euclidean distance and $[\mathbf{w}, d]$ is the coefficient vector of Π. Then, for an edge point $\mathbf{p} \in \hat{\mathcal{E}}^{l_k^i}$, if L is the corresponding edge line, we define the edge residual as a combination of two planar residuals using (4.7) as

$$\mathbf{r}_{\mathcal{E}}(\mathbf{x}_k, \mathbf{p}, L) = \big[\mathbf{r}_{\mathcal{H}}(\mathbf{x}_k, \mathbf{p}, \Pi_1),\ \mathbf{r}_{\mathcal{H}}(\mathbf{x}_k, \mathbf{p}, \Pi_2)\big], \tag{4.8}$$

where $[\mathbf{w}_1, d_1]$ and $[\mathbf{w}_2, d_2]$ are the coefficients of Π_1 and Π_2, $\mathbf{w}_1$ coincides with the projection direction from L to $\mathbf{p}$, and Π_2 is perpendicular to Π_1 s.t. $\mathbf{w}_2 \perp \mathbf{w}_1$, and $\mathbf{w}_2 \perp L$. The above definitions are different from that of LOAM [205], and show two benefits. Firstly, the edge residuals offer additional constraints to the states. Furthermore, the residuals are represented as vectors, not scalars, allowing us to multiply a 3×3 covariance matrix. We minimize the sum of all residual errors to obtain the MLE as

$$\hat{\mathbf{x}}_k = \arg\min_{\mathbf{x}_k} \sum_{\mathbf{p} \in \hat{\mathcal{F}}^{l_k^i}} \rho\Big(\big\|\mathbf{r}(\mathbf{x}_k, \mathbf{p})\big\|^2_{\mathbf{\Sigma}_\mathbf{p}}\Big), \tag{4.9}$$

where the residual term is

$$\mathbf{r}(\mathbf{x}_k, \mathbf{p}) = \begin{cases} \mathbf{r}_{\mathcal{E}}(\mathbf{x}_k, \mathbf{p}, L) & \text{if } \mathbf{p} \in \hat{\mathcal{E}}^{l_k^i} \\ \mathbf{r}_{\mathcal{H}}(\mathbf{x}_k, \mathbf{p}, \Pi) & \text{if } \mathbf{p} \in \hat{\mathcal{H}}^{l_k^i} \end{cases}, \tag{4.10}$$

In practice, points are skewed after a movement on LiDARs with a rolling-shutter scan. After solving the incremental motion $\mathbf{x}_k$, we correct points' positions by transforming them into the last frame $()^{l_{k-1}^i}$. Let t_{k-1} and t_k be the start and end time of a LiDAR scan

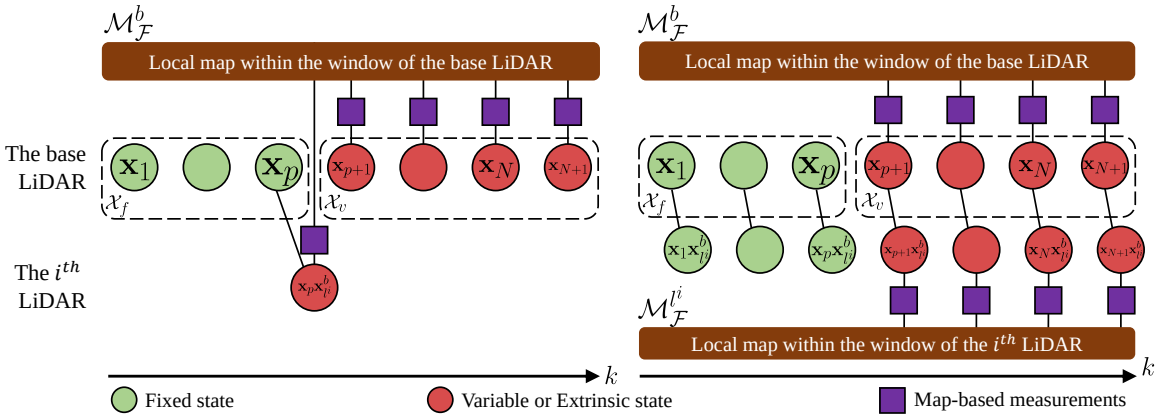


Figure 4.4: Illustration of a graphical model for a sliding window estimator ($p = 3, N = 6$) with online calibration (left) and pure odometry (right). $\mathcal{M}_{\mathcal{F}}^{b}$ ($\mathcal{M}_{\mathcal{F}}^{l^i}$) is the local map of the base (i^{th}) LiDAR. It consists of transformed and merged feature points captured by the base (i^{th}) LiDAR from the first N frames in the window. Note that the extrinsics are optimized in the online calibration, while they are fixed in the pure odometry.

respectively. For a point $\mathbf{p}$ captured at $t \in (t_{k-1}, t_k]$, it is transformed as[3]

$$\mathbf{p}^{l_{k-1}^i} = \mathbf{R}_k^{\tau}\mathbf{p} + \tau\mathbf{t}_k, \quad \tau = \frac{t - t_{k-1}}{t_k - t_{k-1}}, \tag{4.11}$$

where

$$\mathbf{R}^{\tau} = \exp(\boldsymbol{\phi}^{\wedge})^{\tau} = \exp(\tau\boldsymbol{\phi}^{\wedge}), \tag{4.12}$$

where the relative transformation is linearly interpolated [197].

4.6.2 Calibration of Multi-LiDAR System

The initial extrinsics are obtained by aligning motion sequences of two sensors. This is known as solving the hand-eye calibration problem $\mathbf{AX} = \mathbf{XB}$, where $\mathbf{A}$ and $\mathbf{B}$ are the historical transformations of two sensors and $\mathbf{X}$ is their extrinsics. As the robot moves, the following equations of the i^{th} LiDAR should hold for any k

$$\mathbf{R}_{l_k^i}^{l_{k-1}^i}\mathbf{R}_{l^i}^{b} = \mathbf{R}_{l^i}^{b}\mathbf{R}_{b_k}^{b_{k-1}}, \tag{4.13}$$

$$(\mathbf{R}_{l_k^i}^{l_{k-1}^i} - \mathbf{I}_3)\mathbf{t}_{l^i}^{b} = \mathbf{R}_{l^i}^{b}\mathbf{t}_{b_k}^{b_{k-1}} - \mathbf{t}_{l_k^i}^{l_{k-1}^i}, \tag{4.14}$$

where the original problem is decomposed into the rotational and translational components according to [196]. We implement this method to initialize the extrinsics online.

[3]The timestampe of each point can be obtained from LiDAR drivers.

4.6.2.1 Rotation Initialization

We rewrite (4.13) as a linear equation by employing the quaternion:

$$\mathbf{q}_{l_k^i}^{l_{k-1}^i} \otimes \mathbf{q}_{l^i}^{b} = \mathbf{q}_{l^i}^{b} \otimes \mathbf{q}_{b_k}^{b_{k-1}} \;\Rightarrow\; \left[\mathbf{Q}_1(\mathbf{q}_{l_{k-1}^i}^{l_k^i}) - \mathbf{Q}_2(\mathbf{q}_{b_k}^{b_{k-1}})\right] \mathbf{q}_{l^i}^{b} = \mathbf{Q}_k^{k-1}\mathbf{q}_{l^i}^{b}, \tag{4.15}$$

where $\otimes$ is the quaternion multiplication operator and $\mathbf{Q}_1(\cdot)$ and $\mathbf{Q}_2(\cdot)$ are the matrix representations for left and right quaternion multiplication [169]. Stacking (4.15) from multiple time intervals, we form an overdetermined linear system as

$$\begin{bmatrix} w_1^0 \cdot \mathbf{Q}_1^0 \\ \vdots \\ w_K^{K-1} \cdot \mathbf{Q}_K^{K-1} \end{bmatrix}_{4K\times 4} \mathbf{q}_{l^i}^{b} = \mathbf{Q}_K \mathbf{q}_{l^i}^{b} = \mathbf{0}_{4K\times 4}, \tag{4.16}$$

where K is the number of constraints, and w_k^{k-1} are robust weights defined as the angle in the angle-axis representation of the residual quaternion:

$$w_k^{k-1} = \rho'(\phi), \quad \phi = 2\arctan\left(||\mathbf{q}_{xyz}||/|q_w|\right), \quad \mathbf{q} = (\check{\mathbf{q}}_{l^i}^{b})^* \otimes (\mathbf{q}_{l_k^i}^{l_{k-1}^i})^* \otimes \check{\mathbf{q}}_{l^i}^{b} \otimes \mathbf{q}_{b_k}^{b_{k-1}}, \tag{4.17}$$

where $\rho'(\cdot)$ is the derivative of the Huber loss, $\check{\mathbf{q}}_{l^i}^{b}$ is the currently estimated extrinsic rotation, and $\mathbf{q}^*$ is the inverse of $\mathbf{q}$. Subject to $\|\mathbf{q}_{l^i}^{b}\| = 1$, we find the closed-form solution of (4.16) using SVD. For full observability of the 3-DoF rotation, sufficient motion excitation are required. Under sufficient constraints, the null space of $\mathbf{Q}_K$ should be rank one. This means that we only have one zero singular value. Otherwise, the null space of $\mathbf{Q}_K$ may be larger than one due to degenerate motions on one or more axes. Therefore, we need to ensure that the second small singular value $\sigma_{\min 2}$ is large enough by checking whether this condition is achieved. We set a threshold $\sigma_{\mathbf{R}}$, and terminate the rotation calibration if $\sigma_{\min 2} > \sigma_{\mathbf{R}}$. The increasing data grows the row of $\mathbf{Q}_K$ rapidly. To bound the computational time, we use a priority queue [35] with the length $K = 300$ to incrementally store historical constraints. Constraints with small rotation are removed.

4.6.2.2 Translation Initialization

Once the rotational calibration is finished, we construct a linear system from (4.14) by incorporating all the available data as

$$\begin{bmatrix} \mathbf{R}_{l_1^i}^{l_0^i} - \mathbf{I}_3 \\ \vdots \\ \mathbf{R}_{l_K^i}^{l_{K-1}^i} - \mathbf{I}_3 \end{bmatrix}_{3K\times 3} \mathbf{t}_{l^i}^{b} = \begin{bmatrix} \hat{\mathbf{R}}_{l^i}^{b}\mathbf{t}_{b_1}^{b_0} - \mathbf{t}_{l_1^i}^{l_0^i} \\ \vdots \\ \hat{\mathbf{R}}_{l^i}^{b}\mathbf{t}_{b_K}^{b_{K-1}} - \mathbf{t}_{l_K^i}^{l_{K-1}^i} \end{bmatrix}_{3K\times 1}, \tag{4.18}$$

where $\mathbf{t}_{l^i}^b$ is obtained by applying the least-squares approach. However, the translation at the $z-$ axis is unobservable if the robot motion is planar. In this case, we set $t_z = 0$ and rewrite (4.18) by removing the $z-$ component of $\mathbf{t}_{l^i}^b$. Unlike [79], our method cannot initialize t_z by leveraging ground planes and must calculate it in the refinement phase (Section 4.7.2).

4.7 Tightly Coupled Multi-LiDAR Odometry With Calibration Refinement

Taking the initial estimates as input, we propose a tightly coupled M-LO to optimize all states within a sliding window. This procedure is inspired by the recent success of bundle adjustment, graph-based formation, and marginalization in multi-sensor systems [102, 145, 197].

4.7.1 Formulation

The full state vector in the sliding window is defined as

$$\begin{aligned}
\mathcal{X} &= [\mathcal{X}_f, \quad \mathcal{X}_v, \quad \mathcal{X}_e] \\
&= [\mathbf{x}_1, \cdots, \mathbf{x}_p, \ \mathbf{x}_{p+1}, \cdots, \mathbf{x}_{N+1}, \ \mathbf{x}_{l^2}^b, \cdots, \mathbf{x}_{l^I}^b], \\
\mathbf{x}_k &= [\mathbf{t}_{b_k}^w, \mathbf{q}_{b_k}^w], \quad k \in [1, N+1], \\
\mathbf{x}_{l^i}^b &= [\mathbf{t}_{l^i}^b, \mathbf{q}_{l^i}^b], \quad i \in [1, I],
\end{aligned} \tag{4.19}$$

where $\mathbf{x}_k$ is the state of the primary sensor in the world frame at different timestamps, $\mathbf{x}_{l^i}^b$ represents the extrinsics from the primary LiDAR to other auxiliary LiDARs, and $N+1$ is the number of states in the window. To establish data association between these states, we build a local map. We use p to index the pivot state of the window and set $\mathbf{x}_p$ as the origin of the local map. With the relative transformations from the pivot frame to other frames, the map is constructed by concatenating with features at the first N frames i.e. $\mathcal{F}^{l_k^i}, k \in [1, N]$. The local feature map of the i^{th} LiDAR, which consists of the local edge map and local planar map, is denoted by $\mathcal{M}^{l^i}$. We split $\mathcal{X}$ into three groups: $\mathcal{X}_f$, $\mathcal{X}_v$, and $\mathcal{X}_e$. $\mathcal{X}_f = [\mathbf{x}_1, \cdots, \mathbf{x}_p]$ are considered as the fixed, accurate states. $\mathcal{X}_v = [\mathbf{x}_{p+1}, \cdots, \mathbf{x}_{N+1}]$ are considered as the variables which are updated recursively during optimization. $\mathcal{X}_e = [\mathbf{x}_{l^2}^b, \cdots, \mathbf{x}_{l^I}^b]$ are the extrinsics, whose setting, in contrast, is a

little complicated. It depends on the convergence of the online calibration. We minimize the sum of all residual errors within the sliding window to obtain a MAP estimation as

$$\hat{\mathcal{X}} = \arg\min_{\mathcal{X}} \left\{ \left\| \mathbf{r}_{pri}(\mathcal{X}) \right\|^2 + f_{\mathcal{M}}(\mathcal{X}) \right\}, \tag{4.20}$$

where $\mathbf{r}_{pri}(\mathcal{X})$ is the prior term from the state marginalization defined in Section 4.7.5 and $f_{\mathcal{M}}(\mathcal{X})$ is the sum of residual errors from the local map. Different from the recursive estimation in Section 4.6.1, the presented sliding window estimator jointly optimizes all states in the window. This approach outputs more accurate results since the local map provides dense and reliable correspondences. If sensors are precisely calibrated, the constraints from other LiDARs are also used. According to the convergence of calibration, we divide the problem into two subtasks: *online calibration* (variable $\mathcal{X}_e$) and *pure odometry* (fixed $\mathcal{X}_e$). At each task, the definition of $f_{\mathcal{M}}(\mathcal{X})$ is different, and we will present the details in Section 4.7.2 and 4.7.3. An illustration of the batch estimator is shown in Fig. 4.4.

4.7.2 Optimization With Online Calibration

We exploit the map-based measurements to refine the coarse initialization results. Here, we treat the calibration as a registration problem. $f_{\mathcal{M}}(\mathcal{X})$ is divided into two functions w.r.t. $\mathcal{X}_v$ and $\mathcal{X}_e$. For states in $\mathcal{X}_v$, the constraints are constructed from correspondences between features of the primary sensor at the latest frames i.e. $\mathcal{F}^{b_k}, k \in [p{+}1, N{+}1]$ and those of the primary local map i.e. $\mathcal{M}^b$. For states in $\mathcal{X}_e$, the constraints are built up from correspondences between features of auxiliary LiDARs at the p^{th} frame i.e. $\mathcal{F}^{l_p^i}$ and the map $\mathcal{M}^b$.

The correspondences between $\mathcal{F}^{b_k}$ and $\mathcal{M}^b$ are found using the method in [205]. KD-Tree is used for fast indexing in a map. For each edge point, we find a set of its nearest points in the local edge map within a specific region. This set is denoted by $\mathcal{S}$, and its covariance is then computed. The eigenvector associated with the largest value implies the direction of the corresponding edge line. By calculating the mean of $\mathcal{S}$, the position of this line is also determined. For each planar point, the coefficients of the corresponding plane in the local planar map are obtained by solving a linear system such as $\mathbf{ws}{+}d = 0, \forall \mathbf{s} \in \mathcal{S}$. Similarly, we find correspondences between $\mathcal{F}^{l_p^i}$ and $\mathcal{M}^b$. Finally, we define the objective

as the sum of all measurement residuals for the online calibration as

$$
\begin{aligned}
f_{\mathcal{M}}(\mathcal{X}) &= f_{\mathcal{M}}(\mathcal{X}_v) + f_{\mathcal{M}}(\mathcal{X}_e) \\
&= \sum_{k=p+1}^{N+1} \sum_{\mathbf{p} \in \mathcal{F}^{b_k}} \rho\Big(\big\| \mathbf{r}(\mathbf{x}_p^{-1}\mathbf{x}_k, \mathbf{p}) \big\|^2_{\mathbf{\Sigma_p}} \Big) + \sum_{i=2}^{I} \sum_{\mathbf{p} \in \mathcal{F}^{l_p^i}} \rho\Big(\big\| \mathbf{r}(\mathbf{x}_{l^i}^b, \mathbf{p}) \big\|^2_{\mathbf{\Sigma_p}} \Big),
\end{aligned} \tag{4.21}
$$

where $\mathbf{x}_p^{-1}\mathbf{x}_k$ defines the relative transformation from the pivot frame to the k^{th} frame.

4.7.3 Optimization With Pure Odometry

Once we finish the online calibration by fulfilling the convergence criterion (Section 4.7.4), the optimization with pure odometry given accurate extrinsics is then performed. In this case, we do not optimize the extrinsics, and utilize all available map-based measurement to improve the single-LiDAR odometry. We incorporate constraints between features of all LiDARs and local maps into the cost function as

$$
\begin{aligned}
f_{\mathcal{M}}(\mathcal{X}) &= f_{\mathcal{M}}(\mathcal{X}_v) \\
&= \sum_{k=p+1}^{N+1} \sum_{\mathbf{p} \in \mathcal{F}^{b_k}} \rho\Big(\big\| \mathbf{r}(\mathbf{x}_p^{-1}\mathbf{x}_k, \mathbf{p}) \big\|^2_{\mathbf{\Sigma_p}} \Big) + \sum_{i=2}^{I} \sum_{k=p+1}^{N+1} \sum_{\mathbf{p} \in \mathcal{F}^{l_k^i}} \rho\Big(\big\| \mathbf{r}(\mathbf{x}_p^{-1}\mathbf{x}_k\mathbf{x}_{l^i}^b, \mathbf{p}) \big\|^2_{\mathbf{\Sigma_p}} \Big),
\end{aligned} \tag{4.22}
$$

where $\mathbf{x}_p^{-1}\mathbf{x}_k\mathbf{x}_{l^i}^b$ is the transformation from the primary LiDAR at the pivot frame to auxiliary LiDARs at the k^{th} frame.

4.7.4 Monitoring the Convergence of Calibration

While working on the online calibration in an unsupervised way, it is of interest to decide whether calibration is converging. After the convergence, we fix the extrinsics. This is beneficial to our system since both the odometry and mapping are given more geometric constraints from auxiliary LiDARs for more accurate poses. As derived in [204], the *degeneracy factor* λ, which is the smallest eigenvalue of the *information matrix*, reveals the condition of a least-square optimization problem. Motivated by this work, we use λ to indicate whether our problem contains sufficient constraints or not for accurate extrinsics. The detailed pipeline to update extrinsics and monitor the convergence is summarized in Algorithm 1. The algorithm takes the function $f_{\mathcal{M}}(\cdot)$ defined in (4.21) as well as the current extrinsics as input, and produces the optimized extrinsics. On line 4, we compute

Algorithm 1: Monitoring Calibration Convergence

Input: objective $f_{\mathcal{M}}(\cdot)$, current extrinsics $\check{\mathbf{x}} \triangleq \mathbf{x}_{li}^{b}$
Output: optimal extrinsics $\hat{\mathbf{x}}$, covariance matrix $\boldsymbol{\Xi}_{calib}$

```
1  Denote L the set of all eligible estimates;
2  if calibration is ongoing then
3      Linearize f_M at x̌ to obtain Λ = (∂f_M/∂x)^⊤ ∂f_M/∂x;
4      Compute the smallest eigenvalue λ of Λ;
5      if λ > λ_calib then
6          Set x̌ as the current extrinsics of the system;
7          L ← L ∪ x̌;
8          if |L| > L_calib then
9              x̂ ← E[x] as the mean;
10             Ξ_calib ← Cov[x] as the covariance;
11             Stop the online calibration;
12 return x̂, Ξ_calib
```

λ from the information matrix of the cost function. On lines 5–7, the extrinsics are updated if λ is larger than a threshold. On line 8, we use the number of candidate extrinsics to check the convergence. On lines 9–10, the convergence criterion is met, and the termination is triggered. We then compute the sampling mean of $\mathcal{L}$ as the resulting extrinsics and the sampling covariance as the calibration covariance.

4.7.5 Marginalization

We apply the marginalization technique to remove the oldest states in the sliding window. The marginalization is a process to incorporates historical information as a prior into the objective, which reduces the information loss caused by discarding states. It is an essential step to maintain the consistency of odometry and calibration results. In our system, $\mathbf{x}_p$ is the only state to be marginalized. By applying the Schur complement, we obtain $\boldsymbol{\Lambda}_{rr}^{*}$ and $\mathbf{g}_{r}^{*}$ with the form of $\boldsymbol{\Lambda}\delta\mathcal{X} = -\mathbf{g}$. The prior residuals are written as $\|\mathbf{r}_{pri}\|^2 = \mathbf{g}_{r}^{*\top}\boldsymbol{\Lambda}_{rr}^{*-1}\mathbf{g}_{r}^{*}$.

4.8 Uncertainty-Aware Multi-LiDAR Mapping

We first review the pipeline of the mapping module of typical LiDAR SLAM systems [110,163,205]. Taking the prior odometry as input, the algorithm constructs a global map and refines the poses with enough constraints. This is done by minimizing the sum of all

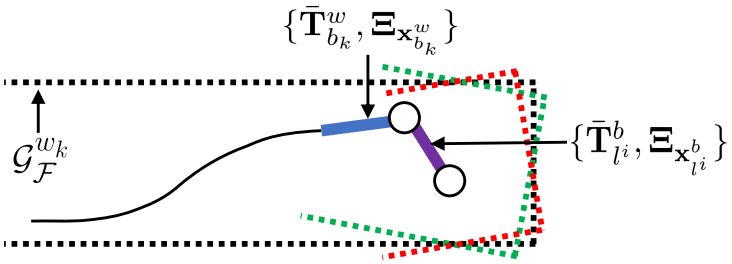


Figure 4.5: Illustration of the mapping process and occurrence of noisy map points. The black curve represents historical poses. The blue curve indicates the current pose of the primary LiDAR. The purple curve shows the extrinsics from the primary LiDAR to the auxiliary LiDAR. With the pose and extrinsics, input features (red and green dots) are transformed and added into the global map (black dots). The noisy poses make new map points uncertain.

map-based residual errors as

$$\hat{\mathbf{x}}^{w}_{b_k} = \underset{\mathbf{x}^{w}_{b_k}}{\arg\min} \sum_{i=1}^{I} \sum_{\mathbf{p}\in\mathcal{F}^{l^i_k}} \rho\Big(\big\|\mathbf{r}(\mathbf{x}^{w}_{b_k}\mathbf{x}^{b}_{l^i}, \mathbf{p})\big\|^2_{\boldsymbol{\Sigma}_{\mathbf{p}}}\Big), \tag{4.23}$$

where $\mathcal{F}^{l^i_k}$ are the input features while perceiving the k^{th} point cloud, $\mathcal{G}^{w_k}$ is the global map, and $\mathbf{x}^{w}_{b_k}\mathbf{x}^{b}_{l^i}$ denotes the state of the i^{th} LiDAR. We use the method in Section 4.7.2 to find correspondences between $\mathcal{F}^{l^i_k}$ and $\mathcal{G}^{w_k}$. After solving (4.23), the resulting pose is used to transform current features into the world frame, and add them into the map. To reduce the computational and memory complexity, the map is also downsized using a voxel grid filter [157]. However, the precision of optimization depends on the map quality. Fig. 4.5 visualizes the occurrence of noisy map points (landmarks) transformed by the uncertain pose. We believe that three sources of uncertainties make map points noisy: sensor noise, degenerate pose estimation, and extrinsic perturbation.

In the next section, we propagate the uncertainties of LiDAR points and poses (represented as transformation matrices) on map points. As a result, each map point is modeled as an i.i.d Gaussian variable. We then propose an uncertainty-aware approach to improve the robustness and accuracy of the multi-LiDAR mapping algorithm.

4.8.1 Uncertainty Propagation

Continuing on the preliminaries in Section 4.3.3, we now introduce how $\boldsymbol{\Xi}$ is computed. The mapping poses are optimized by solving the NLS problem (4.23). We directly calculate the inverse of the *information matrix*, i.e. $\boldsymbol{\Xi}_{\mathbf{x}^{w}_{b_k}} = \boldsymbol{\Lambda}^{-1}$, as the covariance. The

setting of the extrinsic covariances depends on a specific situation. We generally define the extrinsic covariance as

$$\boldsymbol{\Xi}_{\mathbf{x}_{l^i}^b} = \alpha \cdot \boldsymbol{\Xi}_{calib}, \qquad \boldsymbol{\xi}_{l^i}^b \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Xi}_{\mathbf{x}_{l^i}^b}), \tag{4.24}$$

where $\boldsymbol{\xi}_{l^i}^b$ is the perturbation variable of extrinsics, $\boldsymbol{\Xi}_{calib}$ is the calibration covariance calculated according to Algorithm 1, and α is a scaling parameter allowing us to increase the magnitude of the covariance. If a multi-LiDAR system has been recently calibrated, we set $\alpha = 1$, whereas if the system has been used for a long time and not re-calibrated, there should be small extrinsic deviations on LiDARs, and α is set to be larger. Given the mean pose of the base sensor, the mean extrinsics, and their covariances, we then compute the mean poses of other LiDARs and the covariances i.e. $\{\mathbf{T}_{l_k^i}^w, \boldsymbol{\Xi}_{l_k^i}^w\}$. This is a problem about compounding two noisy poses, and we follow the fourth-order approximation in [7] to calculate them. We need to pass the Gaussian uncertainty of a point through a noisy transformation to produce a mean and covariance for a new landmark in the map $\{\mathbf{y}, \boldsymbol{\Sigma}\}$. By transforming a point into the world frame, we have

$$\mathbf{y} \triangleq \mathbf{T}_{l_k^i}^w \mathbf{p}_h = \exp(\boldsymbol{\xi}_{l_k^i}^{w^\wedge}) \bar{\mathbf{T}}_{l_k^i}^w (\bar{\mathbf{p}}_h + \mathbf{D}\boldsymbol{\zeta}) \approx \left(\mathbf{I} + \boldsymbol{\xi}_{l_k^i}^{w^\wedge}\right) \bar{\mathbf{T}}_{l_k^i}^w (\bar{\mathbf{p}}_h + \mathbf{D}\boldsymbol{\zeta}), \tag{4.25}$$

where we keep the first-order approximation of the exponential map. If we multiply out the equation and retain only the first-order terms, we have

$$\mathbf{y} \approx \mathbf{h} + \mathbf{H}\boldsymbol{\theta}, \tag{4.26}$$

where

$$\begin{aligned} &\mathbf{h} = \bar{\mathbf{T}}_{l_k^i}^w \bar{\mathbf{p}}_h, \qquad \mathbf{H} = \left[(\bar{\mathbf{T}}_{l_k^i}^w \bar{\mathbf{p}}_h)^\odot \quad \bar{\mathbf{T}}_{l_k^i}^w \mathbf{D} \right], \\ &\boldsymbol{\theta} = [\boldsymbol{\xi}_{l_k^i}^{w\top}, \boldsymbol{\zeta}^\top]^\top, \quad \boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Theta}), \quad \boldsymbol{\Theta} = \mathrm{diag}(\boldsymbol{\Xi}_{l_k^i}^w, \mathbf{Z}), \end{aligned} \tag{4.27}$$

where $\odot$ converts a 4×1 column into a 4×6 matrix as

$$\begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}^\odot = \begin{bmatrix} \eta \mathbf{I} & -\boldsymbol{\varepsilon}^\wedge \\ \mathbf{0}^\top & \mathbf{0}^\top \end{bmatrix}, \quad \boldsymbol{\varepsilon} \in \mathbb{R}^3, \quad \eta = 1. \tag{4.28}$$

All perturbation variables are embodied in $\boldsymbol{\theta}$ in $\mathbb{R}^9$. The covariance $\boldsymbol{\Theta} = \mathrm{diag}(\boldsymbol{\Xi}_{l_k^i}^w, \mathbf{Z})$ denotes the combined uncertainties of sensor readings, estimated poses, and extrinsics. Obviously, $\mathbf{y}$ is also Gaussian with the mean $\bar{\mathbf{y}}$ and propagated covariance $\boldsymbol{\Sigma}$ as

$$\bar{\mathbf{y}} = \mathbf{h}, \quad \boldsymbol{\Sigma} = \mathbf{H}\boldsymbol{\Theta}\mathbf{H}^\top, \tag{4.29}$$

where we follow [91] to use the trace, i.e., $\mathrm{tr}(\boldsymbol{\Sigma})$, to quantify the magnitude of a covariance.

4.8.2 Uncertainty-Aware Operation

The original mapping algorithm is integrated with three additional steps to boost its performance and robustness. In the last section, we show that the covariances of all map points are propagated by considering three sources of error. First, problem (4.23) is integrated with the propagated covariance in (4.29). This operation lets the cost function take the pose uncertainty and extrinsic perturbation into account. As a result, a point that stays near the origin should have a high weight. Moreover, the primary LiDAR tends to have more confidence than auxiliary LiDARs given the extrinsic covariances.

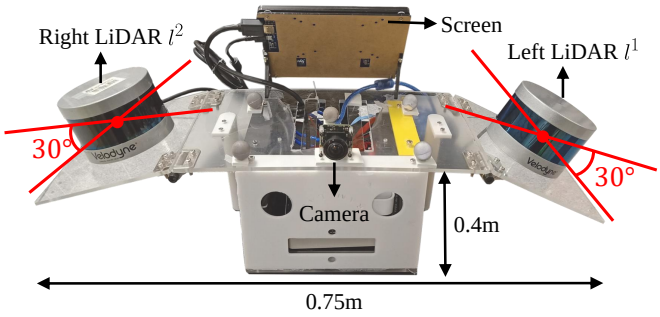
After the optimization, $\boldsymbol{\Xi}_{\mathbf{x}_{b_k}^w}$ is updated as the inverse of the new *information matrix*. Second, the uncertainty of each point after transformation is re-propagated. We filter out outliers if $\text{tr}(\boldsymbol{\Sigma})$ is larger than a threshold. Finally, we modify the original voxel grid filter to downsize the global map in a probabilistic way. The modified filter samples points for each cube according to their covariances. Let $\{\mathbf{y}_i, \boldsymbol{\Sigma}_i\}$ be the i^{th} point in a cube, and m be the number of points in the cube. The sampled mean and covariance of a cube are

$$\bar{\mathbf{y}} = \sum_{i=1}^{M} w_i \mathbf{y}_i, \quad \boldsymbol{\Sigma} = \sum_{i=1}^{M} w_i^2 \boldsymbol{\Sigma}_i, \tag{4.30}$$

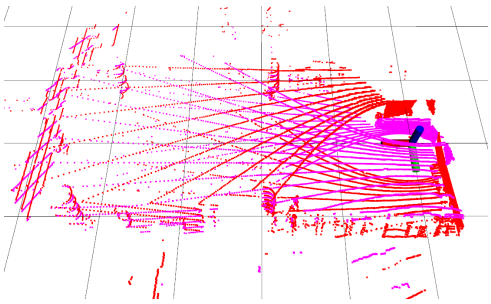
where w is the threshold, and $w_i = \frac{w - \text{tr}(\boldsymbol{\Sigma}_i)}{\sum_{i=1}^{m}[w - \text{tr}(\boldsymbol{\Sigma}_i)]}$ is a normalized weight.

4.9 Experiment

We perform simulated and real-world experiments on three platforms to test the performance of M-LOAM. First, we calibrate the multi-LiDAR systems on all presented platforms. The proposed algorithm is compared with SOTA methods, and two evaluation metrics are introduced. Second, we demonstrate the SLAM performance of M-LOAM in various scenarios covering indoor environments and outdoor urban roads. Moreover, to evaluate the sensibility of M-LOAM against extrinsic error, we test it on a handheld device and vehicle under different levels of extrinsic perturbation. Finally, we provide a study to comprehensively evaluate M-LOAM's performance and computation time with various LiDAR combinations.

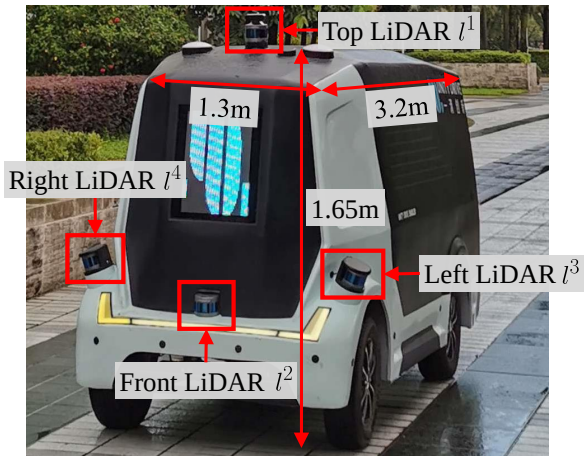


(a) The Real handheld device

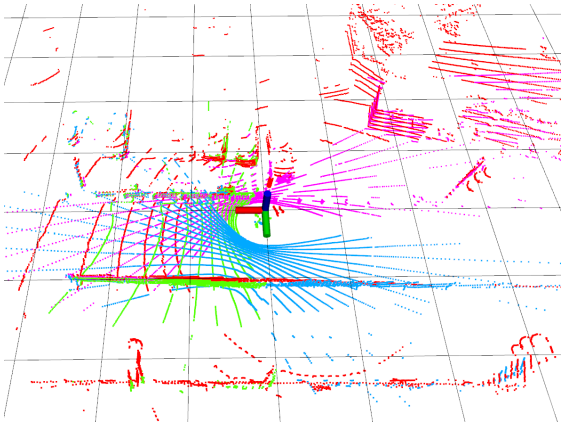


(b) Calibrated point cloud

Figure 4.6: (a) The real handheld device for indoor tests. Two VLP-16s are mounted at the left and right sides respectively. The attached camera is used to record test scenes. (b) The calibrated point cloud consists of points from the left (red) and right (pink) LiDARs.



(a) The Real vehicle



(b) Calibrated point cloud

Figure 4.7: (a) The real vehicle for large-scale, outdoor tests. Four RS-16s are mounted at the top, front, left, and right position respectively. (b) The calibrated point cloud consists of points from the top (red), front (green), left (blue), and right (pink) LiDARs.

4.9.1 Implementation Details

We use the PCL library [157] to process point clouds and the Ceres Solver [2] to solve nonlinear problems. In experiments which are not specified, our algorithm is executed on a desktop with an i7 CPU@4.20 GHz and 32 GB RAM. Three platforms with different multi-LiDAR systems are tested: a simulated robot, a handheld device, and a vehicle. The LiDARs on real platforms are synchronized with the external GPS clock triggered at a ns-level accuracy.

- **The Simulated Robot (SR)** is built on the Gazebo [92]. Two 16-beam LiDARs are mounted on a mobile robot for testing. We build a closed simulated rectangular room. We use the approach from [127] to set the LiDAR configuration for maximiz-

Table 4.2: Parameters for calibration and SLAM.

$\sigma_{\mathbf{R}}$	λ_{calib}	$\mathcal{L}_{calib}$	N	p	w	α
0.25	70	25	4	2	100	≥ 1

Table 4.3: Calibration results. ↓ indicates that the lower the value, the better the score.

Case	Method	Rotation $[deg]$			Translation $[m]$			$EGT_{\mathbf{R}}$ $[deg, \downarrow]$	$EGT_{\mathbf{t}}$ $[m, \downarrow]$	$\overline{MME}$ [↓]	
		x	y	z	x	y	z			$r = 0.3m$	$r = 0.4m$
SR1 (Left-Right)	Auto-Calib	6.134	1.669	0.767	0.001	−0.635	−0.083	33.911	0.209	−2.016	−2.463
	Proposed (Ini.)	44.154	7.062	1.024	−0.027	−0.719	0.000	8.229	0.328	−2.240	−2.685
	Proposed (Ini.+Ref.)	40.870	0.397	0.237	−0.012	−0.475	−0.206	**0.997**	**0.018**	**−2.690**	**−3.073**
	PS-Calib	40.021	−0.005	−0.010	0.001	−0.476	−0.218	0.037	0.003	−2.730	−3.115
	W/O Calib	0.000	0.000	0.000	0.000	0.000	0.000	40.000	0.525	−2.358	−2.704
	GT	40.000	0.000	0.000	0.000	−0.477	−0.220	–	–	−2.733	−3.111
SR2 (Left-Right)	Auto-Calib	4.680	−1.563	0.647	0.032	−0.751	−0.022	35.337	0.339	−2.336	−2.447
	Proposed (Ini.)	40.854	3.517	0.285	−0.019	−0.667	0.000	3.632	0.291	−2.607	−2.804
	Proposed (Ini.+Ref.)	38.442	0.111	−0.037	0.000	−0.504	−0.205	**1.549**	**0.030**	**−3.016**	**−3.192**
	PS-Calib	40.021	−0.005	−0.010	0.001	−0.476	−0.218	0.0365	0.003	−3.113	−3.306
	W/O Calib	0.000	0.000	0.000	0.000	0.000	0.000	40.000	0.525	−2.875	−2.878
	GT	40.000	0.000	0.000	0.000	−0.477	−0.220	–	–	−3.117	−3.313
RHD (Left-Right)	Auto-Calib	7.183	−3.735	33.329	0.653	−2.006	−0.400	44.312	1.612	−3.612	−2.711
	Proposed (Ini.)	36.300	0.069	−3.999	0.113	−0.472	−0.103	6.443	0.112	−3.664	−2.839
	Proposed (Ini.+Ref.)	37.545	−0.376	0.773	0.066	−0.494	−0.113	**2.491**	**0.064**	**−3.681**	**−2.862**
	CAD Model	40.000	0.000	0.000	0.000	−0.456	−0.122	2.077	0.092	−3.662	−2.833
	W/O Calib	0.000	0.000	0.000	0.000	0.000	0.000	40.000	0.560	−3.696	−2.868
	PS-Calib	39.629	−1.664	1.193	0.033	−0.540	−0.142	–	–	−3.696	−2.868
RV (Top-Front)	Auto-Calib	−19.634	21.610	−3.481	−0.130	−0.282	−0.850	22.852	0.791	−2.705	−2.282
	Proposed (Ini.)	1.320	7.264	3.011	−0.324	0.227	0.000	3.217	1.433	−2.721	−2.332
	Proposed (Ini.+Ref.)	−2.057	6.495	2.133	0.528	−0.036	−1.102	**0.274**	**0.081**	**−2.885**	**−2.370**
	CAD Model	0.000	10.000	0.000	0.795	0.000	−1.364	4.505	0.351	−2.771	−2.312
	W/O Calib	0.000	0.000	0.000	0.000	0.000	0.000	7.227	1.252	−2.785	−2.306
	PS-Calib	−1.817	6.629	2.134	0.536	0.039	−1.131	–	–	−2.902	−2.416

ing the sensing coverage. We moved the robot in the room at an average speed of $0.5m/s$. The ground-truth extrinsics and poses are provided.

- **The Real Handheld Device (RHD)** is made for handheld tests and shown in Fig. 4.6. Its configuration is similar to that of the SR. Besides VLP-16s[4], we also install a mini computer (Intel NUC) for data collection and a camera (mvBlueFOX-MLC200w) for recording test scenes. We used this device to collect data on the campus with an average speed of $2m/s$.

- **The Real Vehicle (RV)** is a vehicle for autonomous logistic transportation [116].

[4] `https://velodynelidar.com/products/puck`

We conduct experiments on this platform to demonstrate that our system also performs well in large-scale, challenging outdoor environments. As shown in Fig. 4.7, four RS-LiDAR-16s[5] are rigidly mounted at the top, front, left, and right positions respectively. We drove the vehicle through urban roads at an average speed of $3m/s$. Ground-truth poses are obtained from a coupled LiDAR-GPS-encoder localization system that was proposed in [215, 219].

Table 4.2 shows the parameters which are empirically set. $\sigma_{\mathbf{R}}$, λ_{calib}, and $\mathcal{L}_{calib}$ are the convergence thresholds in calibration. Setting the last two parameters requires a preliminary training process, which is detailed in the supplementary material [82]. p and N are the size of the local map and the sliding window in the odometry respectively. w is the threshold of filtering uncertain points in mapping, and α is the scale of the extrinsic covariance. We set $\alpha = 10$ for the case of injecting large perturbation in Section 4.9.4. Otherwise, $\alpha = 1$.

4.9.2 Performance of Calibration

4.9.2.1 Evaluation Metrics

We introduce two metrics to assess the LiDAR calibration results from different aspects:

- **Error Compared With Ground truth (EGT)** computes the distance between the ground truth and the estimated values in terms of rotation and translation as

$$EGT_{\mathbf{R}} = \left\| \ln(\mathbf{R}_{gt}\mathbf{R}_{est}^{-1})^{\vee} \right\|, \quad EGT_{\mathbf{t}} = \left\| \mathbf{t}_{gt} - \mathbf{t}_{est} \right\|, \tag{4.31}$$

- **Mean Map Entropy (MME)** is proposed to measure the compactness of a point cloud [38]. It has been explored as a standard metric to assess the quality of registration algorithms [151]. Given a calibrated point cloud, the normalized mean map entropy is

$$MME = \frac{1}{m}\sum_{i=1}^{m} \ln\left[\det(2\pi e \cdot \mathbf{C}_{\mathbf{p}_i}) \right], \tag{4.32}$$

  where m is the size of the point cloud and $\mathbf{C}_{\mathbf{p}_i}$ is the sampling covariance within a local radius r around $\mathbf{p}_i$. For each calibration case, we select 10 consecutive frames

[5] `https://www.robosense.ai/rslidar/rs-lidar-16`

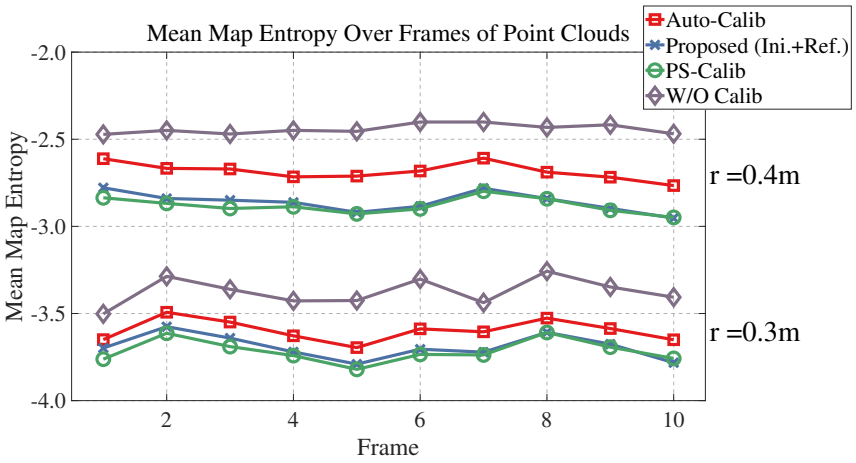


Figure 4.8: The MME values over 10 consecutive frames of point clouds which are calibrated by different approaches on the RHD platform. The lower the value, the better the score for a method.

of point clouds that contain many planes and compute their average MME values for evaluation.

Since the perfect ground truth is unknown in real-world applications, we use the results of **PS-Calib** [81] as the "relative ground truth" to compute the *EGT*. PS-Calib is a well-understood, target-based calibration approach, which should have similar or superior accuracy to our method [74]. Another metric is the MME, which computes the score in an unsupervised way. It can be interpreted as an information-theoretic measure of the compactness of a point cloud.

4.9.2.2 Calibration Results

The multi-LiDAR systems of all the presented platforms are calibrated by our methods. To initialize the extrinsics, we manually move these platforms with sufficient rotations and translations. We compare our method with **Auto-Calib** [79] that is an offline multi-LiDAR calibration approach. Although Auto-Calib follows a similar initialization-refinement procedure to obtain the extrinsics, it is different from our algorithm in several aspects. For example, Auto-Calib only uses planar features in refinement. And it assumes that LiDARs' views should have large overlapping regions. The results using the initial (**Initialization**), uncalibrated (**W/O Calib**), **CAD model**, and ground-truth (**GT**) extrinsics are also provided for reference. Table 4.3 reports the calibration results, where two simulated cases (same extrinsics, different motions) and two real-world cases

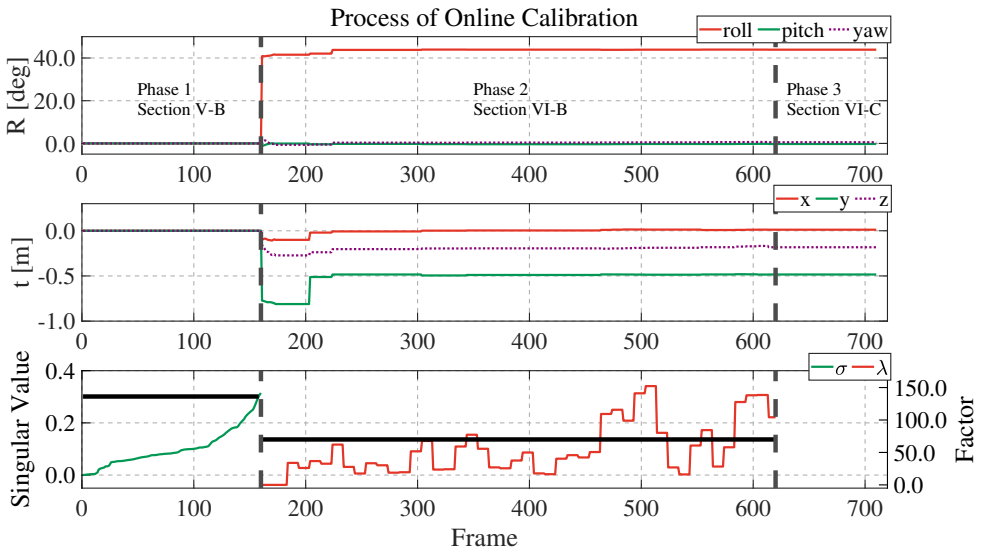


Figure 4.9: Detailed illustration of the whole calibration process, including the initialization and optimization with online calibration on the RHD. Different phases are separated by bold dashed lines. In Phase 1, the initial rotation and translation are estimated with the singular value-based exit criteria (Section 4.6.2). In Phase 2, the nonlinear optimization-based calibration refinement process is performed. The convergence is determined by the *degeneracy factor* (Section 4.7.4). Phase 3 only optimizes the LiDAR odometry with fixed extrinsics. The black lines in the bottom plot indicate the setting thresholds $\sigma_{\mathbf{R}}$ and λ_{calib}, which are defined in Section 4.7.4.

are tested. Due to space limitation, we only demonstrate the calibration between the top LiDAR and front LiDAR on the vehicle.

Our hand-eye-based method successfully initializes the rotation offset ($< 9deg$) for all cases, but fails to recover the translation offset ($> 0.3m$) on the SR and RV. Both have to perform planar movement with a long distance for initialization, making the recovery of the $x-, y-$ translation poor due to the drift of motion estimation. The planar movement also causes the $z-$ translation to be unobservable. But we can move the RHD in 6DoF and rapidly gather rich constraints. Its initialization results are thus good. Regarding the online refinement, our algorithm outperforms Auto-Calib and demonstrates comparable performance with PS-Calib in terms of the EGT ($< 3deg$ and $< 0.07m$) and MME metrics. Based on these results, we conclude that the initialization phase can provide coarse rotational estimates, and the refinement for precise extrinsics is required.

We explicitly show the test on the RHD in detail. In Fig. 4.8, we plot all MME values over different frames of calibrated point clouds, where the results are consistent with Table 4.3. Whether r is set to $0.3m$ or $0.4m$, our method always has a better score than Auto-Calib. Fig. 4.9 illustrates all phases of the calibration process on the RHD, with the

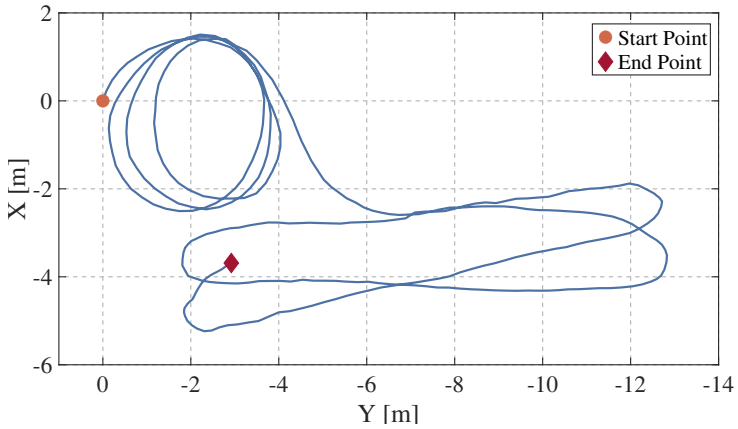


Figure 4.10: Calibration trajectory of the sensor suite estimated by M-LO on the RHD. The dot and diamond indicate the start and end point respectively.

trajectory of the sensor suite shown in Fig. 4.10. The calibration starts with recovering the rotational offsets (Phase 1, Section 4.6.2) without prior knowledge about the mechanical configuration. Phase 1 exits when the second small singular values of $\mathbf{Q}_K$ are larger than a threshold. The translational components are then computed. Phase 2 (Section 4.7.2) performs a nonlinear optimization to jointly refine the rotation and translation. This process may last for a prolonged period if there are no sufficient environmental constraints. However, our sliding window-based marginalization scheme ensures a bounded-complexity program to consistently update the extrinsics. The convergence condition is monitored with the *degeneracy factor* (Section 4.7.4). After convergence, we turn off the calibration and and enter phase 3 for pure odometry (Section 4.7.3) and mapping (Section 4.8), which are evaluated in Section 4.9.3.

We also evaluate the sensitivity of our refinement method to different levels of initial guesses: the CAD model as well as rough rotational and translational initialization. Quantitative results can be found in the supplementary material [82].

4.9.3 Performance of SLAM

We evaluate M-LOAM on both simulated and real-world sequences which are collected by the SR, RHD, and RV platforms. The multi-LiDAR systems are calibrated with our online approach (Section 4.7.2). The detailed extrinsics can be found on the first, third, and fourth row in Table 4.3. We compare M-LOAM with two SOTA, open-source LiDAR-based algorithms: **A-LOAM**[6] (the advanced implementation of LOAM [205])

[6] `https://github.com/HKUST-Aerial-Robotics/A-LOAM`

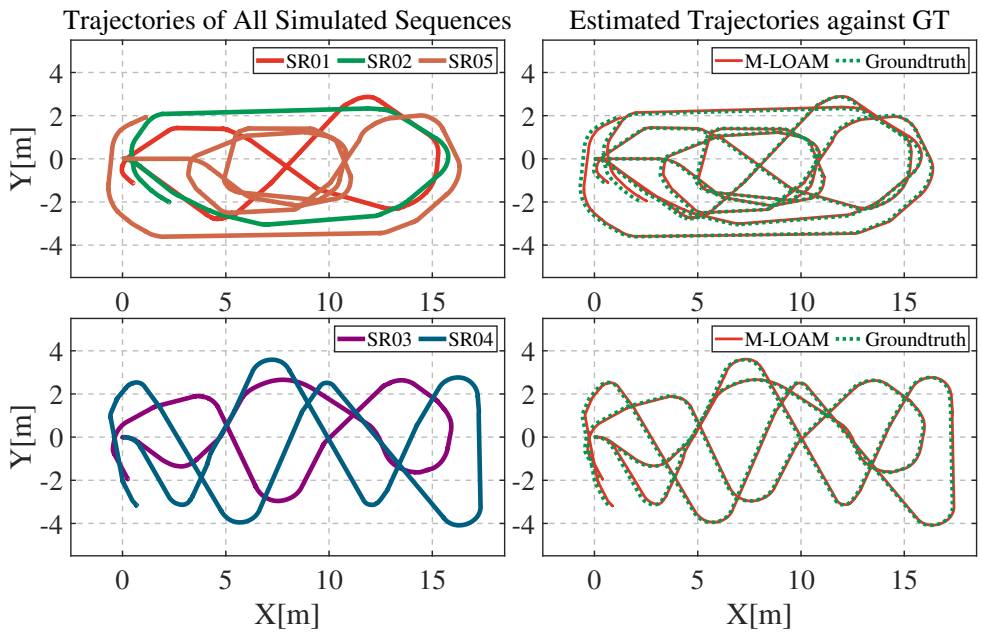


Figure 4.11: (Left) Trajectories of the *SR01-SR05* sequences with different lengths. (Right) M-LOAM's trajectories compared against the ground truth.

and **LEGO-LOAM** [163]. Both of them directly take the calibrated and merged point clouds as input, while our approach employs the optimization-based method to fuse point clouds. There are many differences in detail among these methods, as presented in the technical sections. Overall, our system is complete with online calibration, uncertainty estimation, and probabilistic mapping. LEGO-LOAM is a ground-optimized system and requires LiDARs to be horizontally installed. It easily fails on the SR and RV. We thus provide its results only on the RV sequences for a fair comparison. The results estimated by parts of M-LOAM are also provided. These are denoted by **M-LO** and **M-LOAM-wo-ua**, indicating our proposed odometry (Section 4.7.3) and the complete M-LOAM without the awareness of uncertainty (Section 4.8.2), respectively. To fulfill the real-time requirement, we run the odometry at 10Hz and the mapping at 5Hz.

4.9.3.1 Simulated Experiment

We move the SR to follow 5 paths with the same start point to verify our method. Each sequence is performed with 10-trial SLAM tests, and at each trial, zero-mean Gaussian noises with an sd of $0.05m$ are added the point clouds. The ground-truth and the estimated trajectories of M-LOAM are plotted in Fig. 4.11. The absolute trajectory error (ATE) on all sequences is shown in Table 4.4, as evaluated in terms of root-mean-square error (RMSE) [210]. All sequences are split into either an *easy* or *hard* level according to their

Table 4.4: ATE [210] on simulated sequences

Metric	Sequence	Length	M-LO	M-LOAM-wo-ua	M-LOAM	A-LO	A-LOAM
$\mathbf{RMSE_t}[m]$	*SR01easy*	40.6m	0.482	**0.041**	**0.041**	2.504	0.060
	SR02easy	39.1m	0.884	**0.034**	**0.034**	3.721	0.060
	SR03hard	49.2m	0.838	**0.032**	**0.032**	4.738	0.059
	SR04hard	74.2m	0.757	**0.032**	**0.032**	2.083	0.388
	SR05hard	81.2m	0.598	**0.033**	**0.033**	4.841	0.208
$\mathbf{RMSE_R}[deg]$	*SR01easy*	40.6m	3.368	0.824	**0.676**	26.484	0.751
	SR02easy	39.1m	7.971	1.070	0.882	37.903	**0.576**
	SR03hard	49.2m	6.431	0.994	0.865	38.923	**0.750**
	SR04hard	74.2m	5.728	0.919	0.772	21.027	**0.711**
	SR05hard	81.2m	6.509	0.754	**0.554**	87.999	2.250

length. First, M-LO outperforms A-LO around $4-10$ orders of magnitudes, which shows that the batch estimator can refine the frame-to-frame odometry. Second, we observe that the mapping module greatly refines the odometry module. Third, M-LOAM outperforms other methods in most cases. This is due to two main reasons. 1) The small calibration error may potentially affect the map quality and degrade M-LOAM-wo-ua and A-LOAM performance. 2) The estimates from A-LO do not provide a good pose prior to A-LOAM. Since the robot has to turn around in the room for exploration, A-LOAM's mapping error accumulates rapidly and further makes the optimized poses worse. This explains why A-LOAM has large error in *SR04hard* and *SR05hard*. One may argue that A-LOAM has less rotational error than other methods on *SR02easy*-*SR04hard*. We explain that A-LOAM uses ground points to constrain the roll and pitch angles, while M-LOAM tends to filter them out.

We show the results of *SR05hard* in detail. The estimated trajectories are shown in Fig. 4.12. A-LOAM has a few defects in the marked box region and at the tail of their trajectories, while M-LOAM-wo-ua's trajectory nearly overlaps with that of M-LOAM. The relative pose errors (RPE) evaluated by [210] are shown in Fig. 4.13. In this plot, M-LOAM has lower rotation and translation errors than others over a long distance.

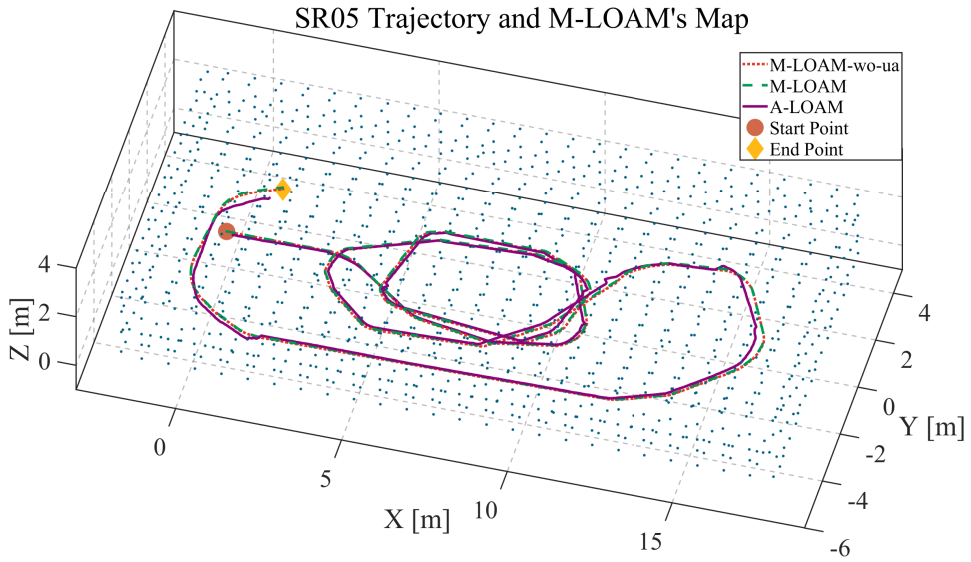


Figure 4.12: Trajectories on *SR05* of M-LOAM-wo-ua, M-LOAM, and A-LOAM and the map constructed by M-LOAM. A-LOAM has a few defects, while M-LOAM-wo-ua's trajectory nearly overlaps with that of M-LOAM.

Table 4.5: Mean Relative Pose Drift

Sequence	Length	M-LO	M-LOAM-wo-ua	M-LOAM	A-LO	A-LOAM
RHD02	197m	3.82%	0.29%	**0.07**%	14.18%	1.13%
RHD03	164m	0.88%	**0.029**%	0.044%	5.31%	0.32%
RHD04	695m	7.30%	0.007%	**0.003**%	34.02%	6.03%

4.9.3.2 Indoor Experiment

We used the handheld device to collect four sequences called *RHD01corridor*, *RHD02lab*, *RHD03garden*, and *RHD04building* to test our approach.

The first experiment is done in a long and narrow corridor. As emphasized in [24], this is a typical poorly-constrained environment. Here we show that the uncertainty-aware operation is beneficial to our system. In Fig. 4.14, we illustrate the sample poses of M-LOAM and the generated map on *RHD01corridor*. These ellipses represent the size of the pose covariances. A large radius indicates that the pose in the $x-, z-$ directions of each mapping step is uncertain. This is mainly caused by the fact that only a small set of points scan the walls and ceiling, which cannot provide enough constraints. Map points become uncertain due to noisy transformations. The uncertainty-aware operation of M-LOAM is able to capture and discard uncertain points. This leads to a map with a reasonaly good signal-to-noise ratio, which generally improves the precision of optimization. This is the

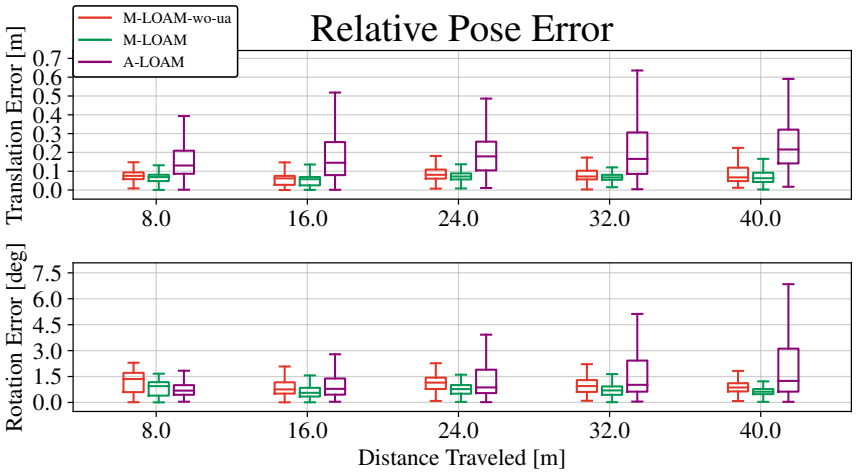


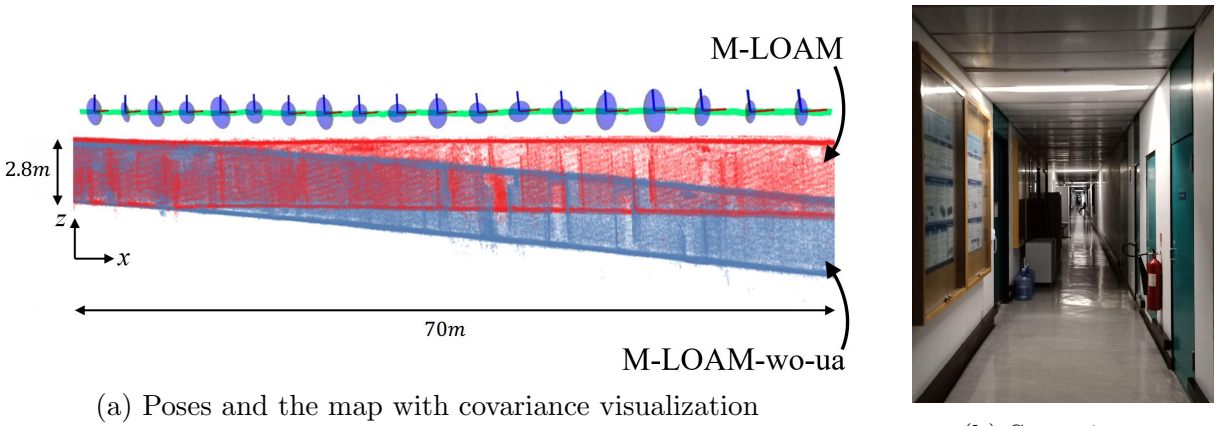
Figure 4.13: The mean RPE on *SR05* with 10 trials. For the distance $40m$, the median values of the relative translation and rotation error of M-LOAM-wo-ua, M-LOAM, and A-LOAM are (0.87deg, 0.07m), (**0.62deg**, **0.06m**), and (1.26deg, 0.22m) respectively.

reason why M-LOAM outperforms M-LOAM-wo-ua.

We conduct more experiments to demonstrate the performance of M-LOAM on other RHD sequences comprehensively. For evaluation, these datasets contain at least a closed loop. Our results of *RHD02lab* are shown in Fig. 4.15. This sequence contains two loops in a lab region. Fig. 4.15a shows M-LOAM's map, and Fig. 4.15b shows the trajectories estimated by different methods. Both M-LOAM-wo-ua and A-LOAM accumulates significant drift at the $x-, y-, z-$ directions after two loops, while M-LOAM's results are almost drift free. Fig. 4.15c shows the estimated poses and map points in the first loop. The covariances of the poses and points evaluated by M-LOAM are visible as ellipses and colored dots in the figure. Besides the corridor in scene 1, we also mark the well-conditioned environment in scene 2 for comparison. Fig. 4.15d shows the scene pictures. The results fit our previous explanation that the points in scene 1 are uncertain because of noisy poses. In contrast, scene 2 has more constraints for estimating poses, making the map points certain.

Fig. 4.16 shows the results of *RHD03garden*. Since the installation of LiDARs on the RHD has a large roll angle, the areas over a 20m height are scanned. Another experiment is carried out in a longer sequence. This dataset lasts for 12 minutes, and the total length is about $700m$. The estimated trajectories and M-LOAM's map are aligned with Google Map in Fig. 4.17. Both M-LOAM-wo-ua and M-LOAM provide more accurate and consistent results than A-LOAM.

Finally, we evaluate the pose drift of methods with 10 repeated trials on *RHD02–*



(a) Poses and the map with covariance visualization

(b) Scene image

Figure 4.14: (a) Side view of sample poses with covariances estimated by M-LOAM and generated map on *RHD01corridor*. The below blue map is created by M-LOAM-wo-ua. The upper red map is created with M-LOAM. The covariances of pose calculated by M-LOAM are visualized as blue ellipses. A large radius represents a high uncertainty of a pose. The pose estimates in the $x-, z-$ direction are degenerate and uncertain, making the map points on the ceiling and ground noisy. M-LOAM is able to maintain the map quality by smoothing the noisy points. (b) The scene image.

RHD04. We employ the point-to-plane ICP [140] to measure the distance between the start and end point. This ground truth distance is used to compare with that of the estimates, and the mean relative drift is listed in Table 4.5. Both M-LOAM-wo-ua and M-LOAM achieve a similar accuracy on *RHD03* and *RHD04* since the surroundings of these sequences are almost well-conditioned. We conclude that the uncertainty-aware operation is not really necessary in well-conditioned environments and well-calibrated sensors, but maximally reduces the negative effect of uncertainties.

4.9.3.3 Outdoor Experiment

The large-scale, outdoor sequence was recorded with the RV platform (Fig. 4.7). This sequence covers an area around $1100m$ in length and $450m$ in width and has $110m$ in height changes. The total path length is about $3.23km$. The data lasts for 38 minutes, and contains the 10-Hz point clouds from fdour LiDARs and 25-Hz ground-truth poses. This experiment is very significant to test the stability and durability of M-LOAM.

M-LOAM's trajectory against the ground truth and the built map is aligned with Google Map in Fig. 4.18. We present the RPE of M-LOAM, M-LOAM-wo-ua, A-LOAM, and LEGO-LOAM in Fig. 4.19. A-LOAM has the highest errors among them. Both M-LOAM-wo-ua and M-LOAM have competitive results with LEGO-LOAM. In addition,

Table 4.6: ATE given different extrinsics from bad to good: Inject perturbation, Initialization, and CAD model.

Case	Extrinsic Source	Rotation $[deg]$			Translation $[m]$			ATE: $\mathbf{RMSE_t}[m]$ ($\mathbf{RMSE_R}[deg]$)			
		x	y	z	x	y	z	M-LOAM-wo-ua	M-LOAM	A-LOAM	LEGO-LOAM
RHD (Left-Right)	Inject Perturbation	49.629	8.236	11.193	0.133	−0.440	−0.042	7.74(29.63)	**0.88**(**8.10**)	4.16(17.79)	–
	Initialization	36.300	0.069	−3.999	0.113	−0.472	−0.103	0.90(**6.46**)	**0.79**(6.92)	1.11(6.44)	–
	CAD Model	40.000	0.000	0.000	0.000	−0.456	−0.122	0.53(3.61)	**0.20**(**2.43**)	0.53(3.94)	–
RV (Top-Front)	Inject Perturbation	8.183	16.629	12.134	0.636	0.139	−1.031	0.75(4.05)	**0.56**(**3.33**)	17.85(15.46)	1.06(6.01)
	Initialization	1.320	7.264	3.011	−0.324	0.227	0.000	0.67(3.45)	**0.60**(**3.23**)	11.72(8.37)	0.90(4.06)
	CAD Model	0.000	10.000	0.000	0.795	0.000	−1.364	0.48(**2.29**)	**0.43**(2.56)	12.95(5.40)	0.73(2.44)

the outlier terms of M-LOAM are fewer than other methods. We thus extend our previous findings that the uncertainty-aware mapping has the capability to enhance the robustness of the system.

4.9.4 Sensitivity to Noisy Extrinsics

In this section, we evaluate the sensitivity of M-LOAM to different levels of extrinsic perturbation. On the RHD and RV platforms, we test our method by setting the extrinsics with different levels of accuracy: CAD model, initialization, and perturbation injection. The experiment settings are listed in Table 4.6. The injected perturbation is the simulated shock on the ground truth extrinsics with $[10, 10, 10]deg$ in roll, pitch, and yaw and $[0.1, 0.1, 0.1]m$ along the $x-$, $y-$, and $z-$ axes. We use *RHD02lab* and a partial sequence on RV to compare M-LOAM with the baseline methods. It should be noted that extrinsic calibration is turned off, and we only use the top and front LiDAR on the RV in experiments. The estimated trajectories under the largest perturbation are shown in Fig. 4.20 and Fig. 4.21 for different platforms. These methods are marked with '(inj)'. We calculate the ATE in Table 4.6. Here, M-LOAM's trajectory on *RHD02lab* in Section 4.9.3.2 is used to compute the error. We observe that all methods' performance is degraded along with the increasing extrinsic perturbation. But both M-LOAM-wo-ua and M-LOAM have smaller error. In particular, under the largest perturbation, M-LOAM is much more robust since it can consistently track sensors' poses.

4.9.5 Single LiDAR v.s. Multiple LiDARs

In this section, we explore the specific improvements in utilizing more LiDARs in M-LOAM. The RV platform has four LiDARs. We use One-, Two-, Three-, Four-LiDAR to denote the setups of l^1, $l^{1,2}$, $l^{1,2,3}$, and $l^{1,2,3,4}$ respectively (Fig. 4.7a). We also use

Table 4.7: Average feature number and running time on a desktop of M-LOAM on the RV sequence with different LiDAR setups.

Setup	One-LiDAR	Two-LiDAR	Three-LiDAR	Four-LiDAR
Edge features	1061	1366	1604	1851
Planar features	4721	5881	7378	8631
Measurement $[ms]$	4.6 ± 0.4	4.8 ± 0.5	5.5 ± 0.7	6.0 ± 0.7
Odometry $[ms]$	27 ± 5	57 ± 7	69 ± 7	73 ± 9
Mapping $[ms]$	78 ± 11	91 ± 13	111 ± 16	126 ± 15

*Specifications: Intel i7 CPU@4.20 GHz and 32 GB RAM

x-Odom and x-Map to denote results provided by the odometry and mapping using different setups, respectively. The tests are carried out on the complete RV sequence. We first report statistics of the program in Table 4.7, including the average number of edge and planar features as well as average running time of measurement processing, optimization with pure odometry, and uncertainty-aware mapping. We see that the odometry time increases with more LiDARs because the system needs to handle more geometric constraints. This phenomenon does not appear in measurement preprocessing since we parallelize this module. The mapping time does not grow linearly because we use the voxel grid filter to bound the map's complexity. We also provide the running time on an Intel NUC (i7 CPU@3.1GHz) in the supplementary material [82], where the results are consistent with Table 4.7.

To demonstrate that more features boost the system performance, we conduct experiments in two cases: driving at a normal speed and high speed. We use the original RV sequence in the first case. To simulate that the vehicle is moving faster in the second case, we extract one frame from every three frames to construct a new dataset. We evaluate the odometry and mapping of these setups in Fig. 4.22. The errors of the odometry decrease if more LiDARs are used. In the second case, the median values of Four-LiDAR-Odom are smaller than those of One-LiDAR-Odom around $8deg$ relative rotation error and 3.5% translation error. But this improvement in mapping is small because the map already provides sufficient constraints. When the vehicle is moving at a higher speed, the global map becomes sparser. Consequently, the improvement of mapping on multiple LiDARs is noticeable. The boxplot of the Four-LiDAR setup has a smaller variance than others.

4.10 Discussion

4.10.1 Main Advantages

We highlight that M-LOAM is a robust, reliable, and complete system to provide accurate calibration, odometry, and mapping results for different multi-LiDAR systems. We can extend M-LOAM to many types of LiDAR combinations, as shown in experiments. A typical application of M-LOAM is autonomous driving, where the multi-LiDAR system is gradually becoming a standard setup on vehicles. As verified in the experiments, the usage of multiple LiDARs boosts the SLAM performance in both robustness and accuracy. For other perception problems such as 3D object detection [78] and tracking, the multi-LiDAR systems are also beneficial.

As compared with the SOTA, M-LOAM introduces the sliding window-based tightly-coupled odometry to fuse multiple LiDARs and the uncertainty-aware mapping to maintain the globally consistent map with good noise-singal ratio. Furthermore, rather than operating calibrated and merged point clouds directly, it processes the multi-LiDAR measurements in a separate way. This design is advantageous in several aspects: *1)* programs (e.g., segmentation and feature extraction) can be easily parallelized, *2)* the LiDAR's scan models can be used to generate a range image without data loss, and *3)* the extrinsic perturbation on the system can be formulated. We consider that the above improvements enable M-LOAM to outperform A-LOAM on most sequences. Compared with LEGO-LOAM, which uses ground features, M-LOAM is more applicable to diverse applications. Nevertheless, integrating M-LOAM with the ground-optimization pipeline of LEGO-LOAM for mobile robots is encouraging.

The Gaussian distribution is our core hypothesis in modeling data uncertainty. Based on it, we use a tractable method to estimate covariances of poses (derived from information matrices) and extrinsics (given a sampling covariance after calibration). Even though these covariances are approximate, as shown in experiments, e.g., Fig. 4.20, the proposed uncertainty-aware operation significantly improves the robustness of M-LOAM against degeneracy and extreme extrinsic perturbation.

4.10.2 Limitations

We recognize that the proposed calibration and SLAM methods have limitations. First, the calibration process requires some pre-set thresholds which are obtained from experiments. Its accuracy is not perfect for applications such as HD map construction. In practice, the errors should be smaller than $0.01m$ and $1deg$. Otherwise, the calibration errors are proportionally propagated to the map and deteriorate the map quality. This effect cannot be entirely eliminated even though the extrinsic perturbation is modeled by our method.

Second, our system utilizes several point cloud registrations in different phases to estimate states. As a typical non-convex problem, registration requires a good initial transformation. But LiDARs only produce a low-frequency data stream, making this problem sometimes challenging. For example, when a robot moves and turns at a high frequency, our method barely tracks its poses. Also, we use the linear model to interpolate sensors' poses, which cannot represent smooth or fast motion well. In these cases, it would be better to use high-order curves such as B-spline for interpolation [136].

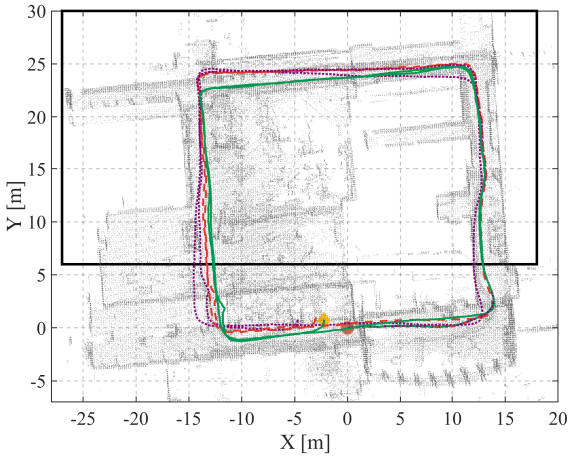
Finally, we extract the simple edge and planar points from environments. However, these features present drawbacks in real tests. For instance, they only provide constraints in their perpendicular directions. In a long tunnel, where all planes are mostly parallel, M-LOAM may fail. Another example is that such features do not have enough recognition power to enable robust matching across frames with large viewpoint changes. As compared with surfel-based or visual features, they are less useful for tasks such as place recognition and relocalization.

4.11 Conclusion and Future Work

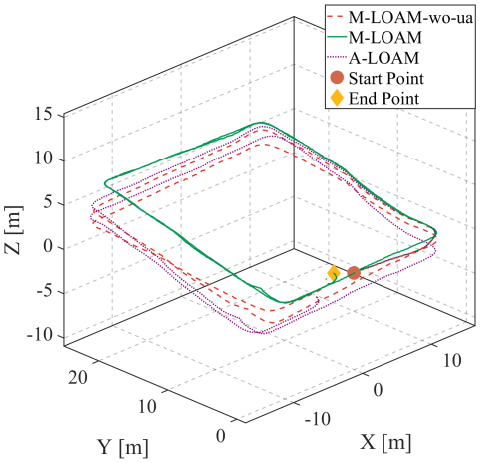
In this thesis, we propose a complete and robust solution for multi-LiDAR extrinsic calibration and SLAM. This approach contains several desirable features, including fast segmentation for noise removal, motion and extrinsic initialization, online extrinsic calibration with convergence identification, a tightly coupled M-LO, and uncertainty-aware multi-LiDAR mapping. We conduct extensive experiments covering scenarios from indoor offices to outdoor urban roads for evaluation. Our approach calibrates kinds of multi-LiDAR systems for different platforms. It yields accuracies centimeters in trans-

lation and deci-degrees in rotation and is comparable to a SOTA target-based method. For SLAM, the proposed system typically reaches a localization accuracy below 40cm in medium-scale ($> 150m$) scenarios and of a few meters in the large-scale urban roads ($> 3.2km$). For the benefit of the community, we make our implementation open-source.

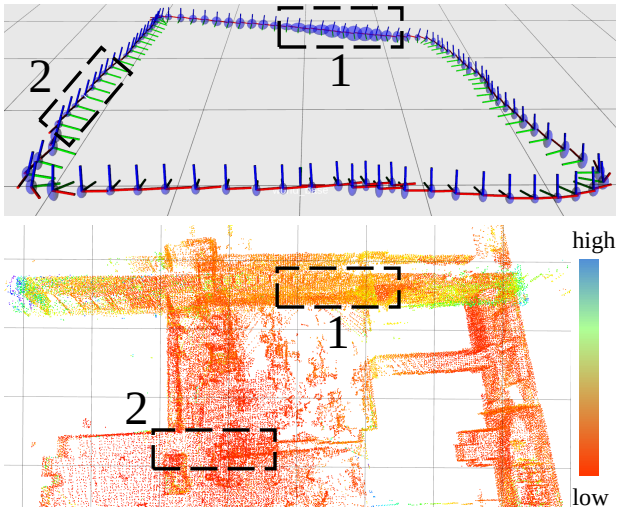
There are several directions for future research. Adding a loop-closure module into our systems is desirable, which helps to correct the accumulated drift and keep the global map [30]. Another research direction concerns object-centric SLAM. Two challenges are recently growing in the community. On the one hand, the widely used low-level geometric features are not representative and sensitive to viewpoint change. On the other hand, data sparsity and occlusion in LiDAR-based object detectors are the dominant bottlenecks. A possible solution to them is to develop a SLAM approach which can use object-level features to optimize both ego-motion and motion of dynamic objects. Trials on cameras or visual-inertial systems have been proposed in [144,194,208], while works on LiDARs are rare. Finally, extending our approach with sensors in various modalities, e.g., IMUs [197], radars [9] and event-cameras [51], is promising. For instance, we can propagate the IMU noise model to predict pose uncertainties, or the proposed convergence criteria can be used for the extrinsic calibration of multi-modal sensors.



(a) Map and trajectories



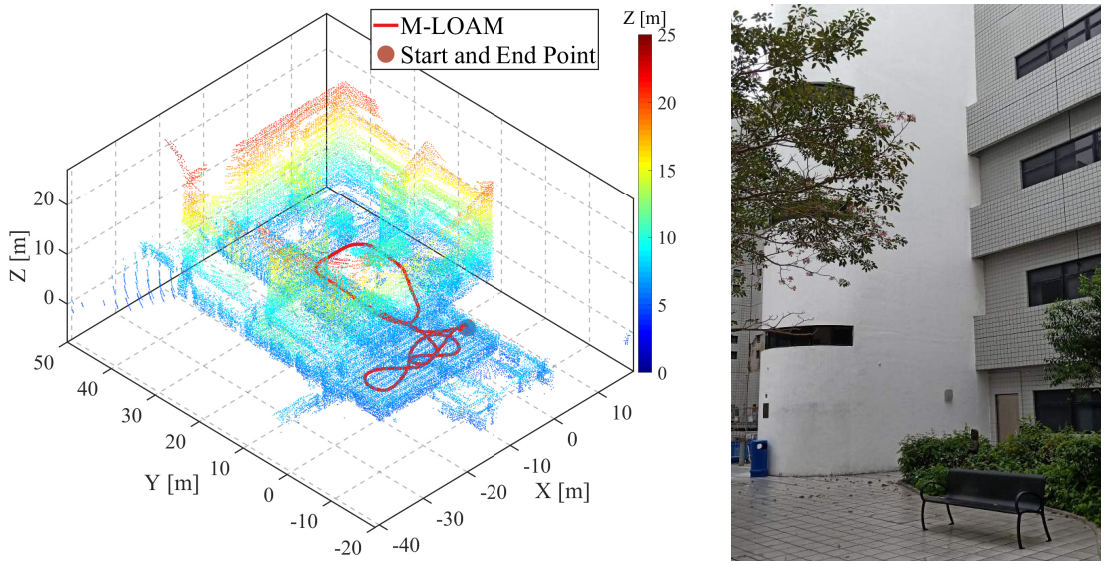
(b) Trajectories



(c) Pose and map points with uncertainties

(d) Scene images

Figure 4.15: Results on *RHD02lab*. (a) Map generated in a laboratory and estimated trajectories (from right to left). The black box is the region shown in the bottom figures. Two loops are in this sequence. (b) Trajectories in another viewpoint. (c) Visualization of poses and map points uncertainty. The grid size is 5m. Covariances of poses are represented as blue ellipses. The larger the radius, the higher the uncertainty. The uncertainty of a point is measured by the trace of its covariance. The larger the trace, the higher the uncertainty. The marked regions indicate the degenerate (scene 1) and well-conditioned (scene 2) pose estimation respectively. With compounded uncertainty propagation, the map points in scene 1 become uncertain. (d) Scene images.



(a) Map and M-LOAM's trajectory

(b) Scene image

Figure 4.16: Results of *RHD03garden*. (a) Map generated in a garden, and the trajectory estimated by M-LOAM. The colors of the points vary from blue to red, indicating the altitude changes ($0m$ to $23m$) (b) Scene images.

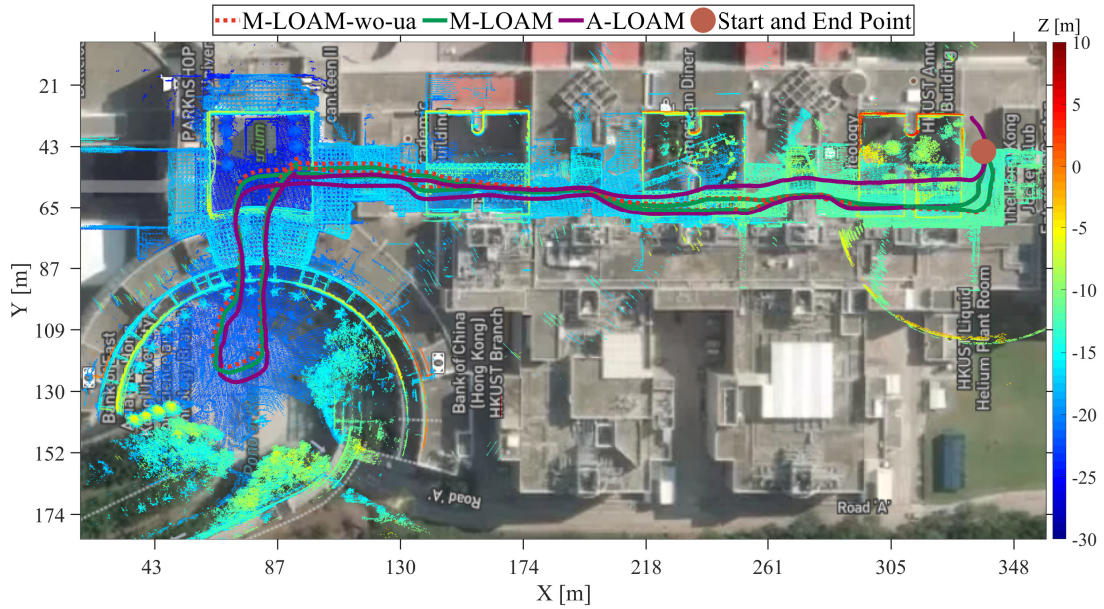


Figure 4.17: Mapping results of *RHD04building* that goes through the HKUST academic buildings and the trajectories estimated by different methods (total length is $700m$). The map is aligned with Google Maps. The colors of the points vary from blue to red, indicating the altitude changes ($0m$ to $40m$)

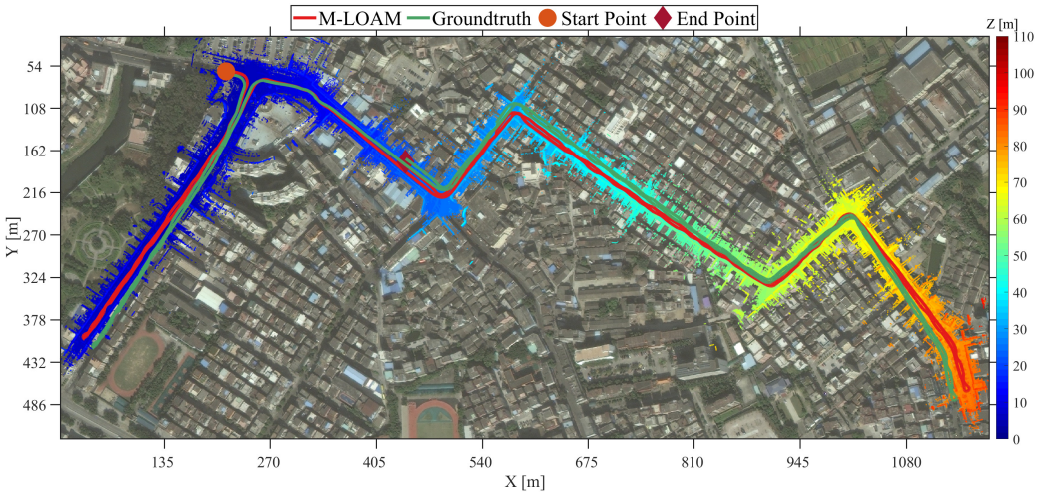


Figure 4.18: Mapping results of urban road and estimated trajectory against the ground truth on the RV sequence (total length is $3.23km$). The colors of the points vary from blue to red, indicating the altitude changes ($-5m$ to $105m$).

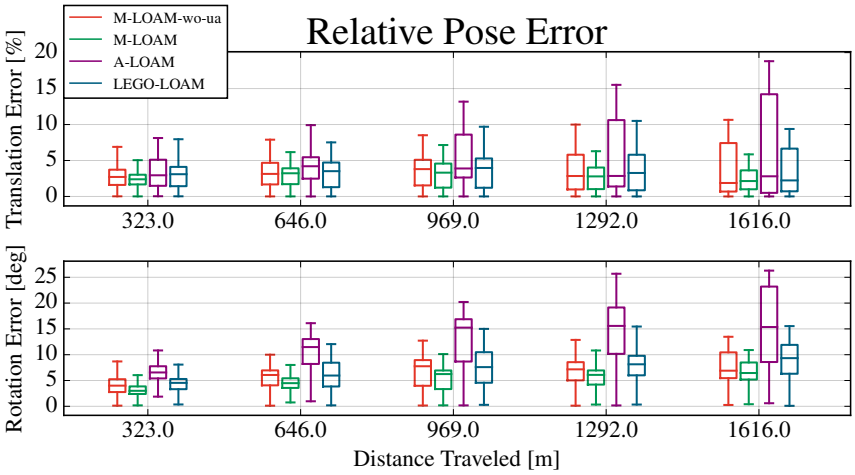
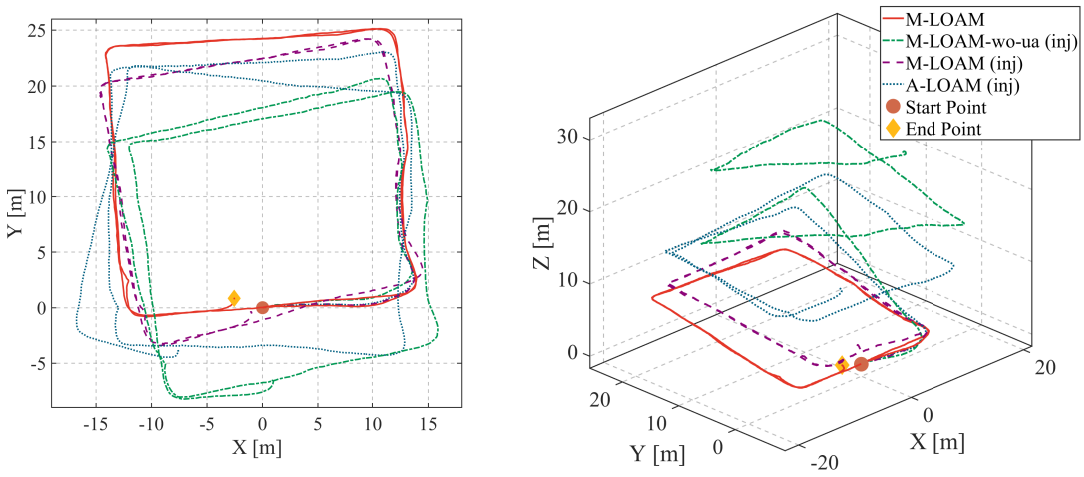


Figure 4.19: RPE on the RV sequence. For the $1616m$ distance, the median values of the relative translation (in percentage) and rotation error of M-LOAM-wo-ua, M-LOAM, A-LOAM, and LEGO-LOAM are (**6.90deg**, **1.87%**), (**6.45deg**, **2.14%**), (15.36deg, 2.80%), (9.33deg, 2.23%) respectively.



(a) Trajectories in a down view

(b) Trajectories in another view

Figure 4.20: Trajectories on *RHD02lab* with being injected by a large extrinsic perturbation. The detailed settings are shown in Table 4.6.

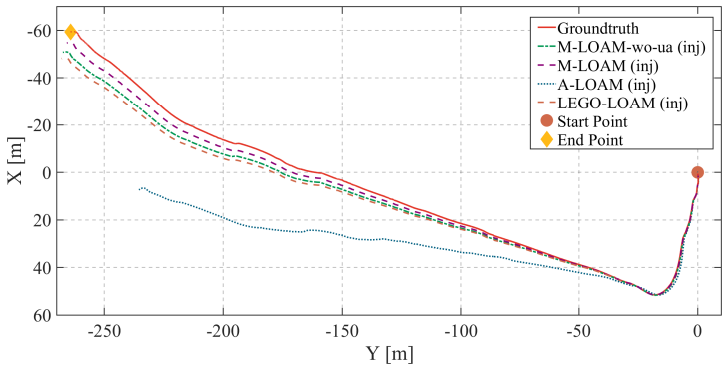
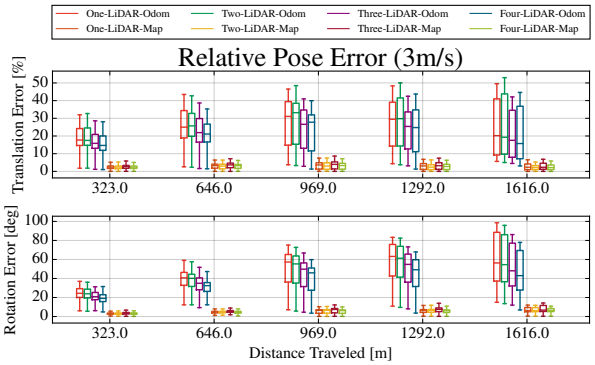
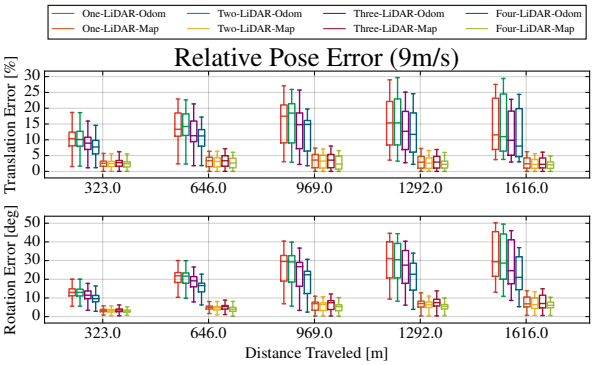


Figure 4.21: Trajectories on 341m-length sequence (a part of the RV sequence) injected with a large extrinsic perturbation. The detailed settings are shown in Table 4.6.



(a) RPE in the case of $3m/s$. From one to four LiDARs, the median values of the rotation and translation error (in percentage) in odometry are: (56.53deg, 20.19%), (54.53deg, 19.29%), (48.08deg, 17.57%), (**42.96deg**, **15.73%**) respectively, while those in mapping are: (6.27deg, 2.35%), (6.26deg, 2.16%), (6.77deg, 2.32%), (**6.45deg**, **2.14%**) respectively



(b) RPE in the case of $9m/s$. From one to four LiDARs, the median values of the rotation and translation error (in percentage) in odometry are: (29.38deg, 11.56%), (28.67deg, 11.00%), (24.60deg, 9.83%), (**21.02deg**, **8.01%**) respectively, while those in the mapping are: (6.86deg, 2.42%), (6.43deg, 2.19%), (7.04deg, 2.24%), (**6.06deg**, **2.11%**) respectively

Figure 4.22: RPE of M-LOAM on the RV sequence with different numbers of LiDARs in two cases. Better visualization in the colored version.

4.12 Appendix

4.12.1 Jacobians of Residuals

The state vector is defined as $\mathbf{x} = [\mathbf{t}, \mathbf{q}]$. We convert $\mathbf{q}$ into a rotation matrix $\mathbf{R}$ by the Rodrigues formula [169]:

$$\mathbf{R} = (q_w^2 - \mathbf{q}_{xyz}^\top \mathbf{q}_{xyz})\mathbf{I} + 2\mathbf{q}_{xyz}\mathbf{q}_{xyz}^\top + 2q_w\mathbf{q}_{xyz}^\wedge, \tag{4.33}$$

4.12.1.1 Jacobians of $\mathbf{r}_\mathcal{H}$

The residuals in (4.9) are rewritten as

$$\begin{aligned}\mathbf{r}_\mathcal{H}(\mathbf{x}, \mathbf{p}) &= \big[\mathbf{w}^\top(\mathbf{R}\mathbf{p} + \mathbf{t}) + d\big]\mathbf{w} \\ &= \mathrm{diag}(\mathbf{w}) \begin{bmatrix}\mathbf{w}^\top \\ \mathbf{w}^\top \\ \mathbf{w}^\top\end{bmatrix} (\mathbf{R}\mathbf{p} + \mathbf{t}) + d\mathbf{w} \\ &= \mathbf{W}(\mathbf{R}\mathbf{p} + \mathbf{t}) + d\mathbf{w}.\end{aligned} \tag{4.34}$$

Using the left perturbation: $\mathbf{R}\exp(\delta\boldsymbol{\phi}^\wedge) \approx \mathbf{R}(\mathbf{I} + \delta\boldsymbol{\phi}^\wedge)$, the Jacobians of the rotation and translation are calculated as

$$\begin{aligned}\frac{\partial \mathbf{r}_\mathcal{H}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{x}} &= [\frac{\partial \mathbf{r}_\mathcal{H}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{t}}, \ \frac{\partial \mathbf{r}_\mathcal{H}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{q}}], \\ &= [\mathbf{W}, \ -\mathbf{W}\mathbf{R}\mathbf{p}^\wedge, \ \mathbf{0}_{3\times 1}].\end{aligned} \tag{4.35}$$

where the quaternion is updated according to $\delta\mathbf{q} \approx [\frac{1}{2}\delta\boldsymbol{\phi}, 1]^\top$.

4.12.1.2 Jacobians of residuals in $f_\mathcal{M}$ for Online Calibration

The objective function in (4.21) has two terms: $f_\mathcal{M}(\mathcal{X}_v)$ and $f_\mathcal{M}(\mathcal{X}_e)$. For the first term, the Jacobians are given by

$$\frac{\partial \mathbf{r}(\mathbf{x}_p^{-1}\mathbf{x}_k, \mathbf{p})}{\partial \mathbf{x}_k} = [\mathbf{W}\mathbf{R}_p^\top, -\mathbf{W}\mathbf{R}_p^\top \mathbf{R}_k \mathbf{p}^\wedge, \mathbf{0}_{3\times 1}], \tag{4.36}$$

where $k \in [p+1, N+1]$. Since the second term has the same form as (4.9), the Jacobians are given by (4.35) as

$$\frac{\partial \mathbf{r}(\mathbf{x}_{l^i}^b, \mathbf{p})}{\partial \mathbf{x}_{l^i}^b} = [\mathbf{W}, \ -\mathbf{W}\mathbf{R}_{l^i}^b \mathbf{p}^\wedge, \ \mathbf{0}_{3\times 1}], \tag{4.37}$$

where $i \in [2, I]$.

4.12.1.3 Jacobians of residuals in $f_{\mathcal{M}}$ for Pure Odometry

The Jacobians of the residuals in (4.22) are computed as

$$\frac{\partial \mathbf{r}(\mathbf{x}_p^{-1}\mathbf{x}_k\mathbf{x}_{li}^b, \mathbf{p})}{\partial \mathbf{x}_k} = \\ [\mathbf{W}\mathbf{R}_p^\top,\ -\mathbf{W}\mathbf{R}_p^\top\mathbf{R}_k(\mathbf{R}_{li}^b\mathbf{p} + \mathbf{t}_{li}^b)^\wedge,\ \mathbf{0}_{3\times 1}], \tag{4.38}$$

where $i \in [1, I]$ and $k \in [p+1, N+1]$.

4.12.2 Marginalization

According to the sliding window formulation, several states will be marginalized out after optimization. For the whole state vector $\mathcal{X}$, let us denote $\mathcal{X}_m$ as the set of states to be marginalized, $\mathcal{X}_r$ as the set of remaining states which are independent to $\mathcal{X}_m$, and $\mathcal{X}_n$ as the irrelevant states. Due to conditional independence, the following marginalization step only involves the states containing $\mathcal{X}_m$ and $\mathcal{X}_r$.

By linearizing the cost function (4.20) at a fixed point, we formulate the linear problem: $\mathbf{\Lambda}\delta\mathcal{X} = -\mathbf{g}$, where $\mathbf{\Lambda} = \sum \mathbf{J}^\top\mathbf{\Sigma}^{-1}\mathbf{J}$ is the *information matrix*, $\mathbf{g} = \mathbf{J}^\top\mathbf{\Sigma}^{-1}\mathbf{r}$, $\mathbf{r}$ is the residual, $\mathbf{J}$ is the Jacobian matrix of the residual w.r.t current states, $\mathbf{\Sigma}$ is the covariance matrix, and $\delta\mathcal{X}$ is the state vector, which minimizes the above linear problem. After estimating $\delta\mathcal{X}$, the Gauss-Newton algorithm will update the current state vector recursively such as $\mathcal{X} \boxplus \delta\mathcal{X} \rightarrow \mathcal{X}$, where $\boxplus$ denotes the operator of the addition of two state vectors. By representing the problem using block matrices, we have

$$\begin{bmatrix} \mathbf{\Lambda}_{mm} & \mathbf{\Lambda}_{mr} \\ \mathbf{\Lambda}_{rm} & \mathbf{\Lambda}_{rr} \end{bmatrix} \begin{bmatrix} \delta\mathcal{X}_m \\ \delta\mathcal{X}_r \end{bmatrix} = - \begin{bmatrix} \mathbf{g}_m \\ \mathbf{g}_r \end{bmatrix}. \tag{4.39}$$

where we apply the Schur complement to yield:

$$\begin{bmatrix} \mathbf{\Lambda}_{mm} & \mathbf{\Lambda}_{mr} \\ \mathbf{0} & \mathbf{\Lambda}_{rr}^* \end{bmatrix} \begin{bmatrix} \delta\mathcal{X}_m \\ \delta\mathcal{X}_r \end{bmatrix} = - \begin{bmatrix} \mathbf{g}_m \\ \mathbf{g}_r^* \end{bmatrix}, \tag{4.40}$$

where

$$\begin{aligned} \mathbf{\Lambda}_{rr}^* &= \mathbf{\Lambda}_{rr} - \mathbf{\Lambda}_{rm}\mathbf{\Lambda}_{mm}^{-1}\mathbf{\Lambda}_{mr}, \\ \mathbf{g}_r^* &= \mathbf{g}_r - \mathbf{\Lambda}_{rm}\mathbf{\Lambda}_{mm}^{-1}\mathbf{g}_m. \end{aligned} \tag{4.41}$$

We define $\mathbf{g}_{r,0} = \mathbf{g}_r^*$ at the first iteration which are fixed during the marginalization. At the next iteration, the remaining states are updated with the form of $\mathcal{X}_r \boxplus \delta\mathcal{X}_r \rightarrow \mathcal{X}_r$.

We thus have the new $\mathbf{g}_r^*$ from (4.41) as

$$\mathbf{g}_r^* = \mathbf{g}_{r,0} - \mathbf{\Lambda}_{rr}^* \delta \mathcal{X}_r, \tag{4.42}$$

We add the resulting $\mathbf{\Lambda}_{rr}^*$ and $\mathbf{g}_r^*$ to construct the cost function, where the constraints of the marginalized states are incorporated. This helps to maintain the consistency of the states in future optimization. The error term from the marginalization is written as $\|\mathbf{r}_{pri}\|^2 = \mathbf{g}_r^{*\top} \mathbf{\Lambda}_{rr}^{*-1} \mathbf{g}_r^*$. Since the Ceres solver [2] uses Jacobians to update variables, when implementing the marginalization, we need to use Jacobians to represent *information matrices*. We factorize $\mathbf{\Lambda}_{rr}^*$ with eigenvalues and eigenvectors:

$$\mathbf{\Lambda}_{rr}^* = \mathbf{P} \mathbf{\Psi} \mathbf{P}^\top. \tag{4.43}$$

Let $\mathbf{J}^* = \sqrt{\mathbf{\Psi}} \mathbf{P}^\top$, $\mathbf{r}^* = \sqrt{\mathbf{\Psi}^{-1}} \mathbf{P}^\top \mathbf{g}_r^*$, we have

$$\mathbf{J}^{*\top} \mathbf{J}^* = \mathbf{\Lambda}_{rr}^*, \quad \mathbf{J}^{*\top} \mathbf{r}^* = \mathbf{g}_r^*. \tag{4.44}$$

Hence, the prior residuals are equal to $\left\| \mathbf{r}^* + \mathbf{J}^* \Delta \mathcal{X}_r \right\|^2$.

4.13 Supplementary Materials

A video showing methodology and experimental results of our system to is available at: `https://www.youtube.com/watch?v=VqaIb3GaCmE`. Here, we additionally present more quantitative and quanlitative results for calibration and SLAM of M-LOAM.

4.13.1 Performance of Calibration

4.13.1.1 Parameter Setting via. a Training Process

Setting the convergence thresholds: λ_{calib} and $\mathcal{L}_{calib}$ for the online calibration requires a preliminary training process. Since the simulated data provides the perfect extrinsic ground truth, we additionally collected data in simulated experiments as the training sequence. We ran the online calibration program on this sequence with 5 trials. At each trial, we set $\lambda_{calib} \in \{50, 70, 90, 110, 130\}$ respectively and analyze the calibration error. We find that setting $\lambda_{calib} = 70$ is proper, while setting it too small or large may result in inaccurate results. Fig. 4.23 plots the values of λ and the calibration error when setting $\lambda_{calib} = 70$. We observe that the calibration error decreases along with calibration frames.

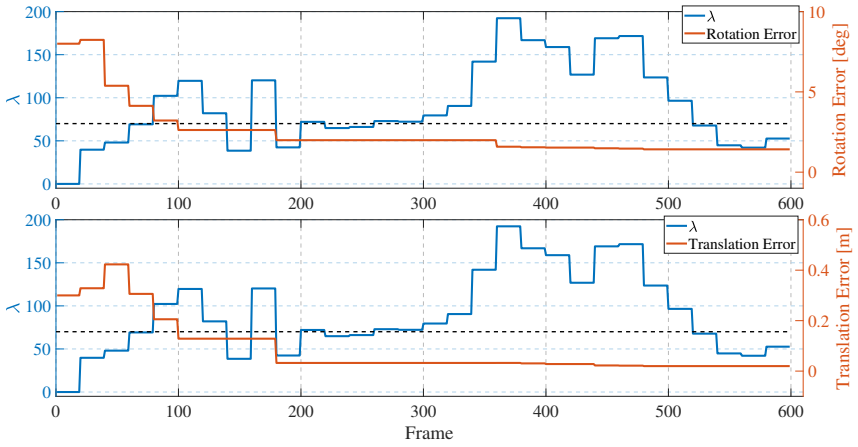


Figure 4.23: The value of λ and the calibration error over frames.

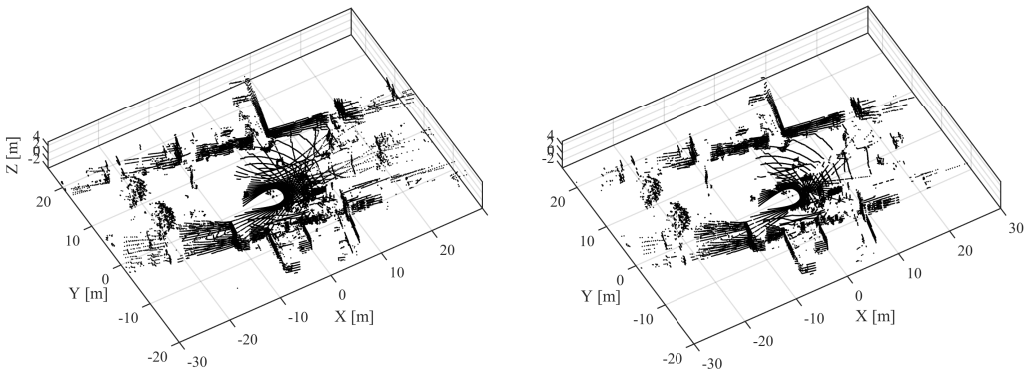
Table 4.8: Online calibration results given different-level initialization. ↓: the lower the better score.

Case	Method	Rotation $[deg]$			Translation $[m]$			$EGT_{\mathbf{R}}$ $[deg, \downarrow]$	$EGT_{\mathbf{t}}$ $[m, \downarrow]$
		x	y	z	x	y	z		
W/O Initialization	Initialization	0.000	0.000	0.000	0.000	0.000	0.000	39.697	0.560
	Refinement	2.448	0.781	2.241	0.102	−0.479	−0.005	**37.265**	**0.165**
Rough Rotation 1	Initialization	25.000	0.000	0.000	0.000	0.000	0.000	14.788	0.560
	Refinement	27.083	−0.048	1.345	0.0668	−0.476	−0.123	**12.647**	**0.075**
Rough Rotation 2	Initialization	35.000	0.000	0.000	0.000	0.000	0.000	5.076	0.560
	Refinement	36.615	−0.180	0.907	0.067	−0.499	−0.077	**3.376**	**0.083**
Rough Translation	Initialization	0.000	0.000	0.000	0.000	−0.500	−0.100	39.697	0.525
	Refinement	2.594	0.862	2.577	0.110	−0.556	0.0425	**37.133**	**0.200**
CAD Initialization	Initialization	40.000	0.000	0.000	0.000	−0.456	−0.122	2.077	0.092
	Refinement	38.759	−0.266	0.697	0.0678	−0.487	−0.094	**1.726**	**0.079**
PS-Calib (GT)		39.629	−1.664	1.193	0.033	−0.540	−0.142	–	–

Also, from this training process, we consider that setting $\mathcal{L}_{calib}$ (the maximum size of $\mathcal{L}$) ≥ 15 is proper. For the handheld device and vehicle platform, we skipped this tedious process and directly used the above settings. But in specific cases that if we want to shorten the calibration process, we can set the calibration threshold $\mathcal{L}_{calib}$ to be smaller.

4.13.1.2 Sensitivity to Bad Initialization

To calibrate a multi-LiDAR system without any prior knowledge about the mechanical configuration, both the initialization and refinement phases are required. The role of the initialization is to provide good initial rotation and translation. Otherwise, the ICP-based refinement phase is easily stuck into a local minimum. One may argue that if the tedious initialization is always needed. Our response is that if users can manually provide a good initial guess (i.e., $< 5deg$ and $< 0.5m$ w.r.t. the ground truth) from a CAD model or



(a) Raw point cloud. (b) Segmented point cloud.

Figure 4.24: An example of the raw point cloud (a) and the segmented point cloud (b) on the RV sequence.

manual measurement, the initialization can be skipped.

To comprehensively evaluate the sensitivity of our method to bad initial guesses, we performed calibration experiments additionally. Given different-level initialization: W/O Initialization, two rough rotational initialization, one rough translational initialization, and CAD initialization, we run our calibration program on the handheld calibration sequence. Among them, the CAD model provides the most accurate initialization. But due to the defect of mechanical configurations, the precision of the CAD model is not always good. We observe that our method is sensitive to the "bad" rotational initialization, but can recover the translational offset on a small error w.r.t. the ground truth. We explain that our method is easily stuck into the local minimum if bad initial guesses are given.

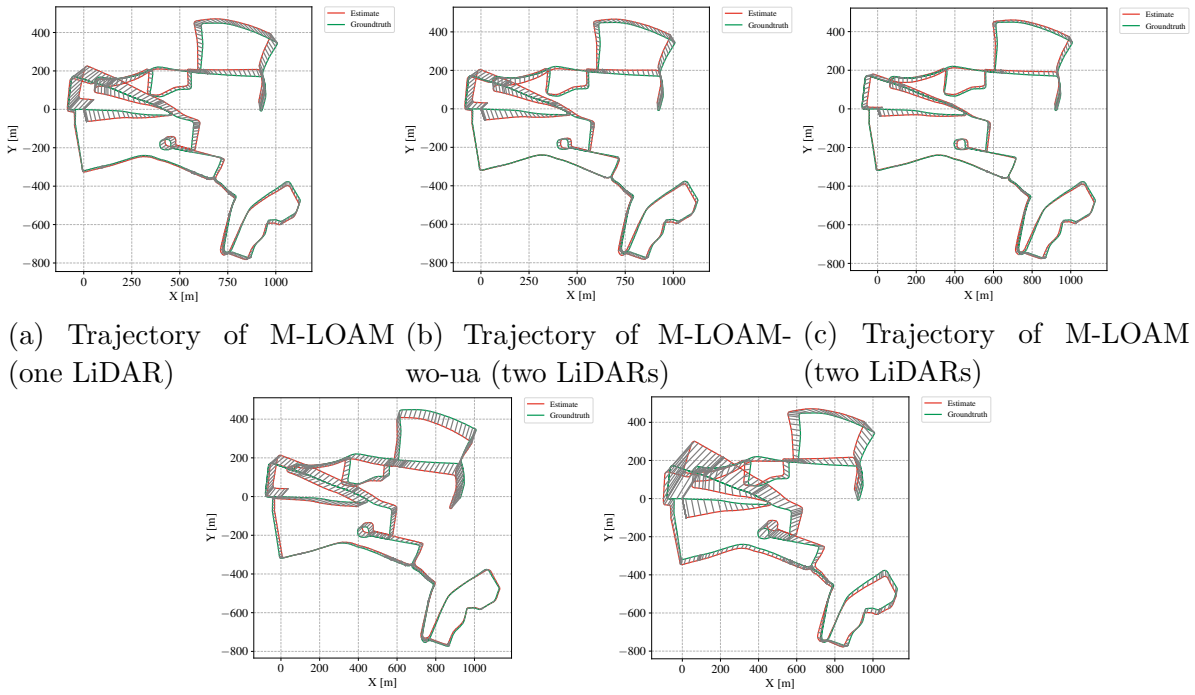
4.13.2 Performance of SLAM

4.13.2.1 Segmentation

We additionally show the segmentation results in outdoor environments, which can help readers to understand our method. An example of the segmented point cloud compared with the raw point cloud on the RV sequence is visualized in the above figure. It should be noted that this point cloud is merged with data from four LiDARs (the top, front, left, and right LiDAR). We observe that some small clusters are removed, but the major objects (e.g., walls) are kept in the segmented point cloud.

Table 4.9: Mean and Variance of the Relative Pose Drift

Sequence	M-LO	M-LOAM-wo-ua	M-LOAM	A-LO	A-LOAM
RHD02	$3.82\% \pm 0.0696\%$	$0.29\% \pm 0.0705\%$	$\mathbf{0.07}\% \pm 0.0119\%$	$14.18\% \pm 0.0117\%$	$1.13\% \pm 0.0239\%$
RHD03	$0.88\% \pm 0.0140\%$	$\mathbf{0.029}\% \pm 0.0104\%$	$0.044\% \pm 0.0128\%$	$5.31\% \pm 0.0003\%$	$0.32\% \pm 0.0100\%$
RHD04	$7.30\% \pm 0.3238\%$	$0.007\% \pm 0.0043\%$	$\mathbf{0.003}\% \pm 0.0043\%$	$34.02\% \pm 0.0096\%$	$6.03\% \pm 0.8939\%$



(a) Trajectory of M-LOAM (one LiDAR) (b) Trajectory of M-LOAM-wo-ua (two LiDARs) (c) Trajectory of M-LOAM (two LiDARs)

(d) Trajectory of A-LOAM (two LiDARs) (e) Trajectory of LEGO-LOAM (two LiDARs)

Figure 4.25: Qualitative results of different methods on the Oxford RoboCar sequence.

4.13.2.2 Indoor Experiment

We quantitatively evaluate the proposed method on three handheld sequences: *RHD02lab*, *RHD03garden*, and *RHD04building* with 10 repeated trials. To better understand our approach's consistency with baseline methods, we provide the mean and variance of the relative pose drift in Table 4.9.

4.13.2.3 Outdoor Experiment

We additionally conduct an outdoor experiment on the Oxford RobotCar platform [9]. This platform is installed with two HDL-32. In this dataset, this vehicle drove through urban roads at an average speed of $10m/s$ to collect data. We select a sequence[7] lasting 34

[7] `https://oxford-robotics-institute.github.io/radar-robotcar-dataset/datasets`

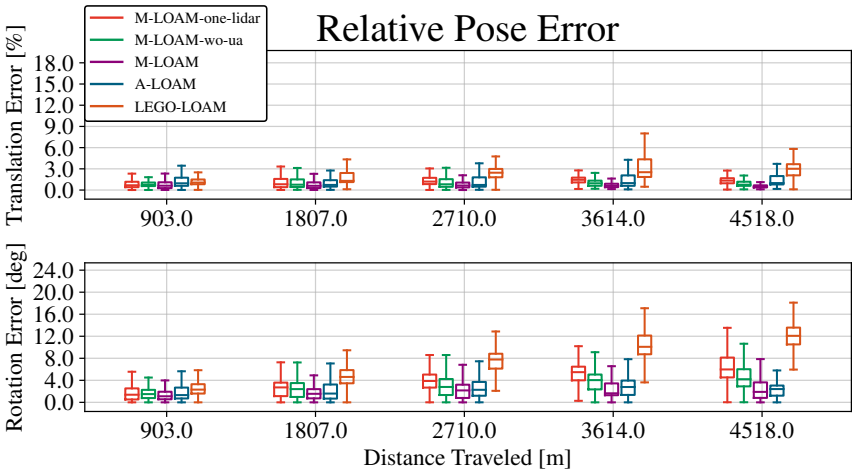


Figure 4.26: RPE on the Oxford sequence. For the distance $4518m$, the median values of the relative translation (in percentage) and rotation error of M-LOAM-one-lidar, M-LOAM-wo-ua, M-LOAM, A-LOAM. LEGO-LOAM are (5.98deg, 1.33%), (4.20deg, 0.76%), (**1.90deg**, **0.48%**), (2.41deg, 0.99%), (12.08deg, 3.01%) respectively.

Table 4.10: Average Running time on an Intel NUC of M-LOAM on the RV sequence with different LiDAR combinations.

Combination	One-LiDAR	Two-LiDAR	Three-LiDAR	Four-LiDAR
Measurement $[ms]$	7 ± 2.0	23 ± 4.1	10 ± 0.8	12 ± 2.1
Odometry $[ms]$	92 ± 16	113 ± 9	113 ± 9	125 ± 12
Mapping $[ms]$	176 ± 36	178 ± 30.0	207 ± 37	229 ± 31

*Specifications: i7 CPU@3.1 GHz and 8 GB memory

minutes which was recorded on 2019/01/18 to test the SLAM performance of M-LOAM with baseline methods. This sequence covers an area that is around $1400m$ in length and $1300m$ in width. The total path length is about $9.04km$. Fig. 4.25 demonstrates the trajectories estimated by M-LOAM and baseline methods: M-LOAM-one-lidar (using only the left LiDAR), M-LOAM-wo-ua, A-LOAM, and LEGO-LOAM. Fig. 4.26 shows the relative pose error w.r.t the ground truth. Both M-LOAM-wo-ua and M-LOAM consistently have better accuracy than M-LOAM-one-lidar. This also shows that using more LiDARs can enhance the SLAM performance. Furthermore, M-LOAM outperforms other methods, which demonstrates the good performance of M-LOAM with various LiDAR combinations.

4.13.2.4 Single LiDAR v.s. Multiple LiDARs

One may be curious about M-LOAM's computation time on an embedded computer. In Table 4.10, we provide the computation time on an Intel NUC. It should be noted

that we ran the rosbag at a low frequency on the NUC. This can ensure the no data is dropped. We hope that this table is helpful to readers. In our extending work [84], we have explored greedy-based feature selection method to accelerate M-LOAM. Readers can check this thesis for details.

CHAPTER 5

GREEDY-BASED FEATURE SELECTION FOR EFFICIENT LIDAR SLAM

Modern LiDAR-SLAM (L-SLAM) systems have shown excellent results in large-scale, real-world scenarios. However, they commonly have a high latency due to the expensive data association and nonlinear optimization. This thesis demonstrates that actively selecting a subset of features significantly improves both the accuracy and efficiency of an L-SLAM system. We formulate the feature selection as a combinatorial optimization problem under a cardinality constraint to preserve the information matrix's spectral attributes. The stochastic-greedy algorithm is applied to approximate the optimal results in real-time. To avoid ill-conditioned estimation, we also propose a general strategy to evaluate the environment's degeneracy and modify the feature number online. The proposed feature selector is integrated into a multi-LiDAR SLAM system. We validate this enhanced system with extensive experiments covering various scenarios on two sensor setups and computation platforms. We show that our approach exhibits low localization error and speedup compared to the state-of-the-art L-SLAM systems. To benefit the community, we have released the source code: `https://ram-lab.com/file/site/m-loam`.

5.1 Introduction

5.1.1 Motivation

State estimation is a classic and fundamental problem in robotics [19]. Over the past decades, LiDARs have attracted much attention from the simultaneous localization and mapping (SLAM) community due to their accuracy and reliability in range measurements. Recent work [10, 111, 119, 163, 205] has pushed LiDAR-SLAM (L-SLAM) systems that are accurate and robust. However, L-SLAM systems commonly present a high latency on a variety of on-board processors with limited computation resources. This issue is critical if the scale of SLAM becomes large, or modules such as high-level decision making are integrated. Thus, towards real-time SLAM for diverse applications, such systems must

exhibit low latency (time delay between input and output) along with the preservation of their accuracy and robustness.

L-SLAM comprises two major computational tasks in L-SLAM: data association and optimization. Data association indicates feature matching between the current frame to the reference frame, while optimization solves the pose parameters by maximizing a likelihood function given a set of constraints. Compared to visual features such as SIFT [120] and ORB [156], matching 3D features is known to be less accurate [193], thus producing much higher outlier rates. To enforce accuracy, most L-SLAM systems exploit thousands of features to solve a large nonlinear least-squares (NLS) problem. However, this scheme presents significant drawbacks. The data association has to perform numerous nearest-neighbor queries to match correspondences, which is commonly time-consuming. Given plentiful measurements, the computational complexity of the optimization based on gradient descent also grows quadratically.

A prevalent solution to bound the complexity is to perform data sampling. For instance, many LiDAR-based object detectors [25, 142, 166] leverage the farthest point sampling (FPS) [40] to process input points. However, classic sampling methods do not consider downstream tasks specifically. Taking FPS as an example, it selects a subset of points with the objective to achieve a maximal coverage of the input set [97]. But for SLAM, it would be better if the process of point selection conforms to the optimization objective. Ideally, exploiting the set of selected features in optimization should lead to low latency and performance improvements.

5.1.2 Contributions

This thesis proposes a general and straightforward feature selection algorithm for L-SLAM systems. We have a crucial observation behind our approach: not all geometric constraints contribute equally to the localization accuracy. Intuitively, well-conditioned constraints should distribute different directions, constraining the pose from different angles [204]. For instance, orthogonal constraints commonly outweigh their parallel counterparts. The selected features, which are the most valuable/informative to the pose estimation, are defined as **good features** [212], and both the data association and state optimization utilize them only.

This thesis extends our previous work on multi-LiDAR SLAM [83]. Multiple LiDARs

enable a robot to maximize its perceptual awareness of environments and obtain sufficient measurements, but inevitably increase the processing time. In this thesis, we investigate the latency issue. From the traditional perspective, there is a trade-off between the latency and accuracy [12]. But in Section 5.6, we demonstrate that the proposed feature selection method boosts the accuracy ($>$ 22% error reduction) and efficiency ($>$ 30% time reduction) of an L-SLAM system. Furthermore, by evaluating the environment's degeneracy and adaptively setting the number of good features, our method also works well in non-ideal cases. Overall, our work presents the following contributions:

1. We transform the good feature section in LiDAR-based pose estimation into a problem that preserves the spectral property of information matrices.

2. We propose and integrate the feature selection method into a multi-LiDAR SLAM system. We also propose to evaluate the environment's degeneracy and adaptively change the number of good features online.

3. We evaluate our approach under extensive experiments with two sensor setups, and computation platforms in terms of accuracy, robustness, and latency.

5.2 Related Work

Scholarly works on SLAM are extensive. Since we focus on the optimal feature selection to improve L-SLAM's efficiency, we review related works on two research topics: feature extraction and selection.

5.2.1 Feature Extraction

Feature extraction is a process to build an informative, compact, and interpretable representation of raw measurements [61]. It has played a crucial role in the front end of many L-SLAM systems to facilitate subsequent tasks. SuMa and its variants [10, 30, 31] convert point clouds into range images and generated surfel-based maps. In contrast, LOAM [205] was proposed to extract features on both edge lines and planar surfaces. LEGO-LOAM [163] leverages ground features to constrain poses in the vertical direction. The following approaches apply visual detection [28] or directly used dense scanners [15, 111] to enhance the feature extraction in noisy or structureless environments. To decrease

the number of features, Zhao et al. [218] presented a probabilistic framework to extract important region of interest by calculating features' densities and distributions. This solution enables the removal of dynamic objects and redundant points in busy urban environments.

All of these methods extract features depending on local geometric structures, but they have not considered the explicit relationship between the pose estimation to select the most useful features. Our system is built on LOAM's framework in feature selection. As a complement to the above appearance-based approaches, our solution identifies a set of good features by utilizing motion information.

5.2.2 Feature Selection

Motion-based feature selection methods have been widely applied in visual SLAM [21, 34, 46, 212], and many of them are based on the covariance or information matrices that capture uncertainties of poses [86]. These methods formulate the feature selection as a submatrix selection problem and aim to find a subset of features with the objective to preserve the information matrix's spectral attributes. Recent works by Carlone et al. [21] and Zhao et al. [212] have investigated greedy-based algorithms [126] to solve this NP-hard feature selection problem at a polynomial-time complexity.

A common limitation of Carlone's and Zhao's works is that they assure sufficient features to be available. Under this assumption, the pose optimization with a set of good features remains well-conditioned. However, robots sometimes need to work in degraded environments such as textureless regions for cameras and narrow corridors for LiDARs. Therefore, only utilizing good features with a fixed size becomes degenerate. Based on Zhao's feature selection approach, our method additionally evaluates environments' degeneracy online. This enables us to adaptively change the number of good features to avoid the risk of ill conditions.

5.3 Nonlinear Least-Squares Pose Estimation

We formulate the pose estimation of a L-SLAM system as an maximum likelihood estimation (MLE) [6]:

$$\hat{\mathbf{x}}_K = \arg\max_{\mathbf{x}_K} p(\mathcal{F}_K|\mathbf{x}_K) = \arg\min_{\mathbf{x}_K} f(\mathbf{x}_K, \mathcal{F}_K), \tag{5.1}$$

where $\mathcal{F}_K$ represents the available features at the K^{th} frame, $\mathbf{x}_K$ is the robot's pose to be optimized, and $f(\cdot)$ is the objective function. Assuming the measurement model to be Gaussian, problem (5.1) is solved as an NLS problem:

$$\hat{\mathbf{x}}_K = \underset{\mathbf{x}_K}{\arg\min} \sum_{i=1}^{N} \rho\big(\big|\big|\mathbf{r}(\mathbf{x}_K, \mathbf{p}_{Ki})\big|\big|^2_{\mathbf{W}_i}\big), \tag{5.2}$$

where $\rho(\cdot)$ is the robust loss [14] and $\mathbf{W}_i$ is the covariance matrix. Problem (5.2) is equivalently rewritten as [6]

$$\hat{\mathbf{x}}_K = \underset{\mathbf{x}_K}{\arg\min} \sum_{i=1}^{N} \big|\big|\mathbf{r}(\mathbf{x}_K, \mathbf{p}_{Ki})\big|\big|^2_{\mathbf{\Sigma}_i}, \tag{5.3}$$

where $\mathbf{\Sigma}_i^{-1} = \rho'(||\mathbf{r}(\mathbf{x}_{\text{op},K}, \mathbf{p}_{Ki})||^2_{\mathbf{W}_i})\mathbf{W}_i^{-1}$ is the alternative covariance matrix and $\rho'(\cdot)$ is the derivative of $\rho(\cdot)$. (5.2) is simplified as an iterative reweighed least-squares problem. Iterative methods such as Gauss-Newton or Levenberg-Marquardt can offen be used to solve this problem. These methods locally linearize the objective function by computing the Jacobian w.r.t. $\mathbf{x}_K$ as $\mathbf{J} = \partial f / \partial \mathbf{x}_K$. Given an initial guess, $\mathbf{x}_K$ is iteratively optimized by usage of $\mathbf{J}$ until convergence to find an optimum.

At the final iteration, the least-squares covariance of the state is calculated as $\mathbf{\Xi} = \mathbf{\Lambda}^{-1}$ [24], where $\mathbf{\Lambda} = \mathbf{J}^\top \mathbf{J}$ is called the *information matrix*. This equation reveals the relationship between the pose covariance and information matrix. Generally, exploiting plentiful measurements in optimization should minimize poses' uncertainties. This explains why SLAM systems commonly use all available features.

This thesis focuses on low-latency applications in which the speed is highly prioritized. It requires us to utilize only a subset of features to accelerate the algorithm. As suggested in [212], we can check whether a feature is selected or not by comparing the gains in spectrum of $\mathbf{\Lambda}$. The word: "spectrum" denotes the set of eigenvalues of a matrix [57].

5.4 Methodology

This section first formulates the good feature selection problem and introduces a metric to guide the selection. We apply the stochastic-greedy method, which combines the random sampling procedure, to solve this problem in real time. We then extend this algorithm to achieve efficient feature selection. Finally, we propose to evaluate the environment's degeneracy online to avoid ill-conditioning estimation.

Table 5.1: Possible definitions of $f(\mathbf{\Lambda})$.

$\mathrm{tr}(\mathbf{\Lambda})$	trace [172]
$\lambda_{\min}(\mathbf{\Lambda})$	minimum eigenvalue [21]
$\log\det(\mathbf{\Lambda})$	log determinant [212]

5.4.1 Problem Formulation

We denote $N = |\mathcal{F}_K|$ as the number of all available features, M as the maximum number of selected features, and $\mathcal{S}_K$ as the good feature set. We denote $f(\cdot)$ as the metric to quantify the spectral attribute of a matrix. We formute the feature selection problem under a cardinality constraint as

$$\underset{\mathcal{S}_K \subset \mathcal{F}_K}{\arg\max}\ f\big[\mathbf{\Lambda}(\mathcal{S}_K)\big] \quad \text{subject to } |\mathcal{S}_K| \leq M, \tag{5.4}$$

where $\mathbf{\Lambda}(\mathcal{S}_K)$ is the information matrix on the good feature set. Table 5.1 gives possible definitions of $f(\mathbf{\Lambda})$.

Since problem (5.4) is NP-hard, we cannot find efficient algorithms to obtain the optimal subset for real-time applications. Fortunately, all these metrics are submodular and monotone increasing [126], allowing the solution to be approximate via greedy methods with a performance guarantee. Zhao et al. [212] experimented with these metrics in bundle adjustment, where the log determinant was demonstrated to have the lowest pose error and computation time. We thus choose the $\log\det(\mathbf{\Lambda})$ option as our metric.

5.4.2 Stochastic Greedy Algorithm

The class of greedy methods has been studied to solve problem (5.4). Here, we introduce the stochastic-greedy algorithm [126], which applies randomized acceleration to avoid brute-force search. The idea is simple: at each round, the current best feature is picked up by examining all elements from a random subset. This is different from the simple greedy approach, which has to search the whole set. We define the size of the random subset as $\frac{N}{M}\log(\frac{1}{\epsilon})$, where ϵ is the *decay factor*. The time complexity is $O(N\log(\frac{1}{\epsilon}))$, which is independent of M. The stochastic-greedy algorithm has been proved to have near-optimal performance in [126]:

Theorem 1: Let $f(\cdot)$ be the non-negative, monotone, and submodular function. Setting the size of the random subset as $\frac{N}{M}\log(\frac{1}{\epsilon})$. Denote $\mathcal{S}_K^*$ as the optimal set and

Algorithm 2: Stochastic Greedy-Based Good Feature Selection for NLS Pose Estimation

Input: $f(\cdot) \triangleq \log\det(\cdot)$, M, ϵ, $\mathcal{M}_K$, $\mathcal{F}_K$, $\mathbf{x}_K$;
Output: good feature set $\mathcal{S}_K^{\#}$;
1 Initialize the set $\mathcal{S}_K^{\#} = \emptyset$;
2 **while** $|\mathcal{S}_K^{\#}| < M$ and $t_{comp} < t_{max}$ **do**
3 $\quad\mathcal{R} \leftarrow$ the random subset is obtained by sampling $\frac{N}{M}\log(\frac{1}{\epsilon})$ random elements from $\mathcal{F}_K \backslash \mathcal{S}_K^{\#}$;
4 $\quad$**foreach** $\mathbf{p}_i \in \mathcal{R}$ **do**
5 $\quad\quad$Search the correspondence from $\mathcal{M}_K$;
6 $\quad\quad$**if** the correspondence is found **then**
7 $\quad\quad\quad$Compute the residual $\mathbf{r}_i(\check{\mathbf{x}}_K)$;
8 $\quad\quad\quad$Compute $\mathbf{\Lambda}_i = \mathbf{J}_i^\top \mathbf{\Sigma}_{\boldsymbol{i}}^{-1} \mathbf{J}_i$ w.r.t. $\mathbf{p}_i$ where $\mathbf{J}_i$ is the Jacobian of $\mathbf{r}_i(\cdot)$;
9 $\quad\quad$**else**
10 $\quad\quad\quad\mathcal{R} \leftarrow \mathcal{R} \backslash \{\mathbf{p}_i\}$; $\quad\mathcal{F}_K \leftarrow \mathcal{F}_K \backslash \{\mathbf{p}_i\}$;
11 $\quad i^* \leftarrow \arg\max_{\mathbf{p}_i \in \mathcal{R}} \log\det\left[\mathbf{\Lambda}(\mathcal{S}_K^{\#}) + \mathbf{\Lambda}_i\right]$;
12 $\quad\mathbf{\Lambda}(\mathcal{S}_K^{\#}) \leftarrow \mathbf{\Lambda}(\mathcal{S}_K^{\#}) + \mathbf{\Lambda}_{i^*}$;
13 $\quad\mathcal{S}_K^{\#} \leftarrow \mathcal{S}_K^{\#} \cup \{\mathbf{p}_{i^*}\}$; $\quad\mathcal{F}_K \leftarrow \mathcal{F}_K \backslash \{\mathbf{p}_{i^*}\}$;

$\mathcal{S}_K^{\#}$ the result found by the stochastic-greedy algorithm. $\mathcal{S}_K^{\#}$ enjoys the approximation guarantees in expectation:

$$E\Big(f\big[\mathbf{\Lambda}(\mathcal{S}_K^{\#})\big]\Big) \geq \underbrace{(1 - 1/e - \epsilon)}_{\text{expected ratio}} f\big[\mathbf{\Lambda}(\mathcal{S}_K^{*})\big]. \tag{5.5}$$

5.4.3 Good Feature Selection for Pose Estimation

Based on the theoretical results, we employ the stochastic-greedy to achieve the feature selection for pose estimation. The detailed pipeline is summarized in Algorithm 2.

1. Overview: The algorithm starts with the $\log\det(\cdot)$ metric, the number of good features M, the *decay factor* ϵ, the map in the reference frame $\mathcal{M}_K$, the set of all available features in the current frame $\mathcal{F}_K$, and the initial pose $\mathbf{x}_K$. It produces the good feature set $\mathcal{S}_K^{\#}$.

2. Line 2: The loop is exited if one of the following conditions is satisfied: M good features are found or the computation time exceeds t_{max}. The second condition limits the cost of finding good features. Since $\log\det(\cdot)$ is submodular with diminishing returns, early termination does not induce much information loss.

3. Lines 4–10: The correspondence of each feature in the random subset $\mathcal{R}$ is found from $\mathcal{M}_K$. The residual is then computed. If a feature has already been visited at previous iterations, we skip these steps.

4. Lines 11–13: The feature which leads to the maximum enhancement of the objective is selected. After that, the information matrix $\mathbf{\Lambda}(\mathcal{S}_K^{\#})$, $\mathcal{S}_K^{\#}$, and $\mathcal{F}_K$ are updated.

Furthermore, the process of feature selection implicitly performs outlier rejection: outliers are penalized by the robust loss in (5.3) with relatively small weights. They contribute less to $\mathbf{\Lambda}$ than standard features and will be selected with a low probability. Therefore, selecting good features might reduce the biases between estimates and the ground truth.

5.4.4 Setting the Number of Good Features

Setting a proper size for the good feature set is essential to the system. Previous methods manually set M as a constant value ($M = 100$ [212]) or a fixed ratio of all features ($M = 0.5N$ [21]). These schemes are feasible if sufficient features are always available. But if a robot has to work in non-ideal scenarios such as textureless walls or narrow corridors, utilizing a small set of features is not reliable. On the other hand, if we change the hard-coded number M in a specific situation, it will inevitably increase the cost of deploying and maintaining a SLAM system on real platforms.

It would be better if M were adaptively changed by evaluating the degeneracy online. Inspired by [204], the magnitude of the degeneracy can be quantified by a *factor* λ. Differently, we define the factor using the log determinant metric as $\lambda = \log\det\mathbf{\Lambda}(\mathcal{F}_K)$. Computing the information matrix on the full feature set is time-consuming. Since the robot performs a continuous movement in an environment, it is enough to compute λ at every time interval ($1s \sim 2s$).

Fig. 5.1 plots the values of λ on different sequences. *RHD01lab* contains several degenerate scenarios, while other sequences do not (see Section 5.6.3). Therefore, we empirically set $\lambda_{th} = 42$. If $\lambda \geq \lambda_{th}$, we select 20% of features from the full set as the good features (i.e., $M = 0.2N$). Otherwise, we use 80% of features from the set.

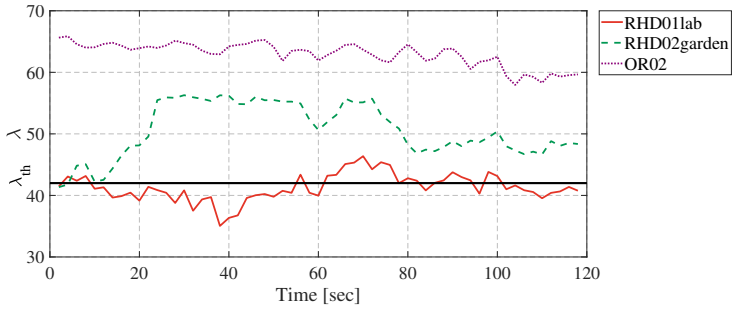
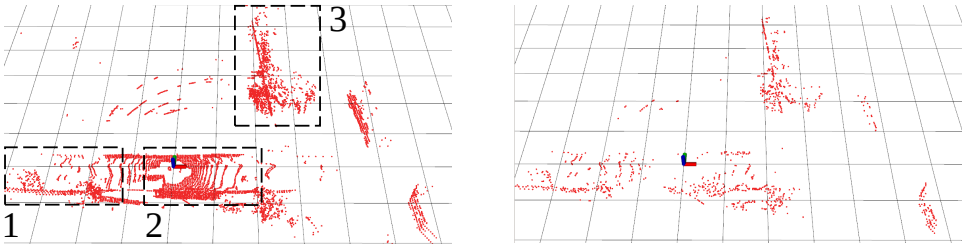


Figure 5.1: The value of the *degeneracy factor* λ on different sequences.



(a) Full feature set (4693 points) (b) Good features (929 points).

Figure 5.2: A qualitative example of good features selected by our greedy-based feature selection algorithm. This method pick up points on objects which provide strong geometric constraints, as indicated in the region 1 and 3. Points on the ground, which occupy around 50% of the full feature set, are indicated by the region 2. Since they only constrain poses on the $z-$ axis, our method only selected them with a small number. This is the main difference from the fully random sampling method (see Section 5.6.2).

5.5 GF-Enhanced Multi-LiDAR SLAM System

The proposed feature selection method has been verified in an L-SLAM system called M-LOAM [83]. To distinguish it from the original system, the enhanced system is denoted by **M-LOAM-gf**. M-LOAM-gf solves SLAM with multiple LiDARs by two algorithms: *odometry* and *mapping*. Generally, these algorithms are designed to estimate poses in a coarse-to-fine fashion. Since they similarly formulate the NLS problem for pose estimation, the feature selection can be applied to both. Fig. 5.3 illustrates the overall structure of M-LOAM-gf. Note that loop closure is not included.

We give real definitions to the feature set $\mathcal{F}_K$. We extract features located on local edges and planar surfaces from the LiDARs' raw measurements. According to the points' curvatures, we select a set of edge and planar features to form $\mathcal{F}_K$. The next step is to match features between the reference frame and the robot's current frame. In both *odometry* and *mapping*, we use the feature map in the reference frame to associate data with $\mathcal{F}_K$. The only difference is the scale. The local map in *odometry* is built within a

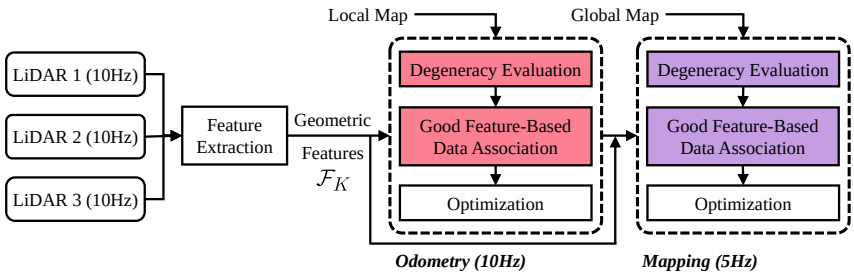


Figure 5.3: Block diagram of the pipeline of M-LOAM-gf.

Table 5.2: Mean and std of $\log\det\mathbf{\Lambda}(\mathcal{S}_K^{\#})$ on RHD and OR sequences.

Sequence	Greedy	Rnd	Full
RHD01lab	$\mathbf{35.1 \pm 2.3}$	30.3 ± 2.7	39.3 ± 2.5
RHD02garden	$\mathbf{41.4 \pm 4.0}$	39.7 ± 5.0	48.2 ± 5.1
OR01	$52.6 \pm \mathbf{1.0}$	$\mathbf{52.7} \pm 1.6$	61.8 ± 1.7
OR02	$50.8 \pm \mathbf{0.9}$	$\mathbf{51.8} \pm 2.2$	60.1 ± 2.1
OR03	$\mathbf{51.7 \pm 1.4}$	49.9 ± 2.6	58.3 ± 2.4
OR04	$\mathbf{51.6 \pm 1.2}$	50.3 ± 2.2	58.7 ± 2.1
OR05	$\mathbf{52.5 \pm 1.7}$	51.9 ± 2.7	59.3 ± 2.4
Average	**47.96**	46.66	55.10

small time interval ($< 0.5s$), while the global map in *mapping* is constructed using all features in keyframes. For convenience, we use $\mathcal{M}_K$ to denote both the local and global maps.

With the found correspondences, we can optimize the relative transformation by minimizing the sum of all errors. The good feature selection algorithm enables M-LOAM-gf to select only a set of features in optimization while preserving the spectrum of the information matrix. An example of good features are shown in Fig. 5.2. After obtaining the good feature set $\mathcal{S}_K^{\#}$ in Section 5.4.3, the objective function is written as

$$\hat{\mathbf{x}}_K = \arg\min_{\mathbf{x}_K} \sum_{\mathbf{p}\in\mathcal{S}_K^{\#}} \left|\left|\mathbf{r}(\mathbf{x}_K, \mathbf{p})\right|\right|^2_{\mathbf{\Sigma_p}}, \tag{5.6}$$

for which the expression of the residuals and Jacobians is detailed in our supplementary material [82].

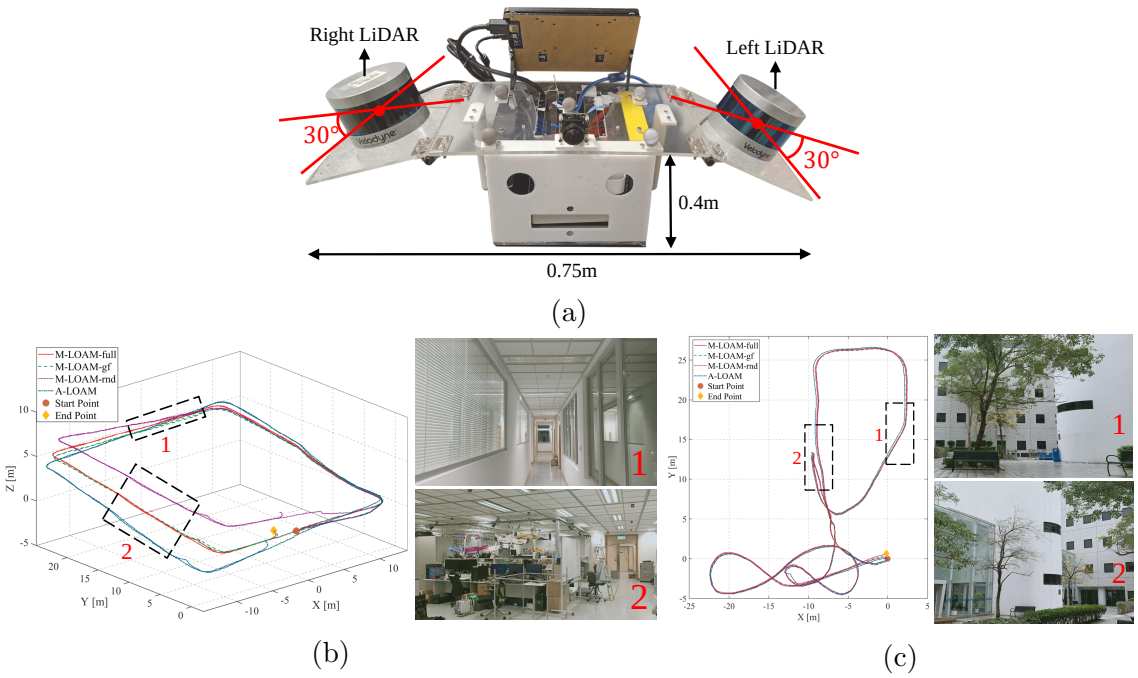


Figure 5.4: Estimated trajectories of different methods and the scene image on (b) *RHD01lab* and (c) *RHD02garden* using (a) the real handheld device (RHD).

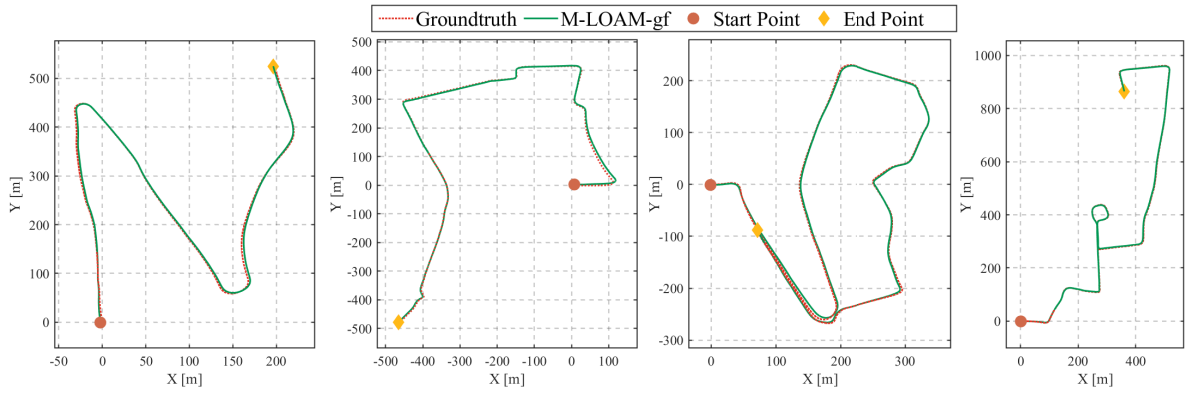
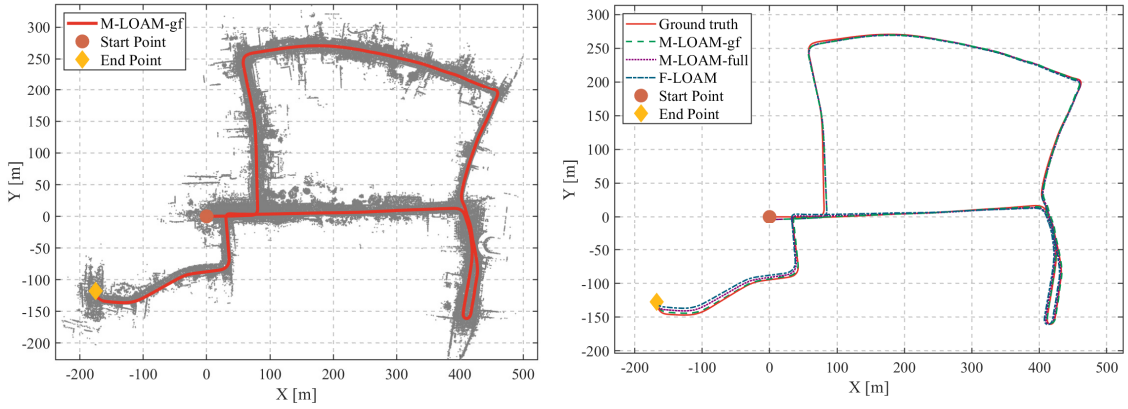


Figure 5.5: M-LOAM-gf's trajectories on *OR01*, *OR03*, *OR04*, and *OR05* from the Oxford Robocar dataset [9] are aligned with the ground truth .

5.6 Experiment

We evaluate the performance of M-LOAM-gf on real-world experiments. First, we validate the stochastic-greedy-based feature selection. Second, we demonstrate the localization accuracy of M-LOAM-gf in various scenarios covering indoor environments and outdoor urban roads with two multi-LiDAR setups. Three SOTA L-SLAM systems are compared. We also study M-LOAM-gf's latency on an on-board processor with limited computation resources.



(a) (b)

Figure 5.6: Results on the *OR02*. (a) Map and M-LOAM-gf's path. (b) Estimated trajectories of different methods are aligned with the ground truth.

5.6.1 Implementation Details

We use the PCL library [157] to process point clouds and the Ceres Solver [1] to solve the NLS problems. Our method is tested on sequences collected with two platforms:

- **Real Handheld Device (RHD)** is made for indoor tests and shown in Fig. 5.4a. It is installed with two VLP-16[1]. We held this device to collect two sequences (*RHD01lab* and *RHD02garden*) with an average speed of $2m/s$.
- **Oxford Robocar (OR)** [8] is a vehicle equipped with two HDL-32E[2]. Datasets were recorded by driving the car on urban roads at an average speed of $10m/s$. 32 repeated traversals of a 9 km route were collected. Ground-truth poses in $SE(2)$ are available. We select one sequence lasting 34 minutes and split it into 5 sequences named *OR01–OR05* for evaluation.

5.6.2 Validation on Good Feature Selection

This section validates that the greedy algorithm selects a set of valuable features with a large $\log\det \mathbf{\Lambda}(\mathcal{S}_K^{\#})$. Our stochastic-greedy method (label: **greedy**) is compared with the fully randomized selection method (label: **rnd**). Table 5.2 reports the means and standard deviations (std) of $\log\det \mathbf{\Lambda}(\mathcal{S}_K^{\#})$. The values with the full feature set (label: **full**) are provided for reference. On *OR01–OR02*, the greedy method gains a smaller

[1] `https://velodynelidar.com/products/puck`

[2] `https://velodynelidar.com/products/hdl-32e`

Table 5.3: Translational RMSE $[m]$ [210] on OR sequences (the two best results are marked as bold text). The average RMSE of different methods are **1.971**, 2.578, 2.688, 6.679, **2.522**, and 5.523.

Sequence	*OR01*	*OR02*	*OR03*	*OR04*	*OR05*
Dimension	$525m \times 252m$	$629m \times 431m$	$573m \times 896m$	$337m \times 498m$	$517m \times 968m$
M-LOAM-gf	**1.986** $\pm$ 0.013	**2.473** $\pm$ 0.151	**1.955** $\pm$ 0.105	**1.720** $\pm$ 0.047	**1.719** $\pm$ 0.027
M-LOAM-rnd	2.324 $\pm$ 0.028	**3.107** $\pm$ 0.104	2.829 $\pm$ 0.158	2.242 $\pm$ 0.084	**2.387** $\pm$ 0.432
M-LOAM-full	2.449 $\pm$ 0.014	3.120 $\pm$ 0.076	3.063 $\pm$ 0.062	2.239 $\pm$ 0.024	2.567 $\pm$ 0.487
A-LOAM	4.504 $\pm$ 0.055	7.535 $\pm$ 0.410	12.995 $\pm$ 0.006	**1.913** $\pm$ 0.057	6.446 $\pm$ 0.015
F-LOAM	**2.174** $\pm$ 0.007	3.709 $\pm$ 0.044	**2.019** $\pm$ 0.024	2.238 $\pm$ 0.062	2.469 $\pm$ 0.061
LEGO-LOAM	4.334 $\pm$ 0.000	4.369 $\pm$ 0.083	5.440 $\pm$ 0.102	6.224 $\pm$ 0.021	7.248 $\pm$ 0.260

objective than the rnd method. This is reasonable since the greedy algorithm cannot always achieve the best performance according to (5.5). By considering the std and the larger means on most sequences, we conclude that the greedy method outperforms the rnd method.

5.6.3 Performance of SLAM

We compare the accuracy, robustness, and latency of M-LOAM-gf with several baseline methods:

- **M-LOAM-rnd** is the variant of M-LOAM with the rnd feature selection module in mapping.
- **M-LOAM-full** is the original M-LOAM that uses the full feature set in mapping.
- **A-LOAM**[3], **F-LOAM**[4], and **LEGO-LOAM** [163] are three SOTA, open-source L-SLAM systems. All of them are the improved versions of LOAM [205].

The *odometry* and *mapping* of all methods run at 10 Hz and 5 Hz. For a fair comparison, the loop closure modules in some baselines are deactivated. The resolutions of the voxel filter [157] on the edge and planar features are $0.2m$ and $0.4m$.

[3] `https://github.com/HKUST-Aerial-Robotics/A-LOAM`

[4] `https://github.com/wh200720041/floam`

5.6.3.1 Qualitative Comparison

We first test our method on RHD sequences. *RHD01lab* is recorded by moving around an office space, in which several scenes provide only poor geometric constraints. Fig. 5.4b indicates two examples. Scene 1 is a long and narrow corridor, which is a typical degenerate environment [197]. Scene 2 is an indoor office, providing well-conditioned constraints. M-LOAM-gf successfully tracks robot poses due to its capability in evaluating the environment's degeneracy. M-LOAM-rnd has a sudden drift in scene 1 since using only 20% of features cannot constrain the poses. A-LOAM also fails because it cannot model the uncertainty in mapping, which is detailed in [83]. *RHD02garden* is collected in a garden. Estimated trajectories and scene images are shown in Fig. 5.4c. Since the environment is well-conditioned, all trajectories are comparative.

5.6.3.2 Localization Accuracy

We then perform a large-scale outdoor test on OR sequences under 10 repetitions. Environments on *OR* sequences commonly provide sufficient features. We visualize M-LOAM-gf's trajectory against the ground truth and the built map on *OR02* in Fig. 5.6, and plot trajectories the other sequences in Fig. 5.5. Each method is evaluated the absolute trajectory error (ATE) and the relative pose error (RPE) [210]. Due to limited space, we report the translation ATE in Table 5.3, and show complete results in our supplementary material [82]. M-LOAM-gf does not just preserve the accuracy of M-LOAM-full, but it also reduces the ATE on all sequences. The average translation ATE of M-LOAM-gf is 22% lower than that of F-LOAM (the second-best method). The feature selection implicitly rejects outliers (see Section 5.4.3), which is essential to such accuracy gains. Thus, M-LOAM-rnd also improves M-LOAM-full. But its drift is larger than that of M-LOAM-gf. The performance of the fully randomized operation in M-LOAM-rnd is not guaranteed, which occasionally lead to inconsistent results.

5.6.3.3 Latency

Experiments in the above sections are conducted on a desktop with an i7 CPU@4.2 GHz and 32 GB memory. The average latency of *mapping* of M-LOAM-gf over 1725 and 10356 frames on the RHD and OR sequences is $62.69ms$ and $106.82ms$ respectively. To demonstrate that our feature selection method boosts an L-SLAM system on processors

Table 5.4: Average latency [ms] of mapping on an Intel NUC.

Sequence	Method	Mapping		
		Data association	Optimization	Total
RHD	M-LOAM-gf	17.90	3.09	1**15.48**
	M-LOAM-rnd	6.33	3.65	**92.57**
	M-LOAM-full	16.17	7.01	128.32
	A-LOAM	–	–	131.08
OR	M-LOAM-gf	46.18	4.64	**149.25**
	M-LOAM-rnd	21.70	6.86	**108.10**
	M-LOAM-full	54.35	22.85	230.35
	A-LOAM	–	–	313.69
	F-LOAM	–	–	271.30
	LEGO-LOAM	–	–	158.23

*Latency: Time delay between the input and output of a function.

with limited resources, M-LOAM-gf is also tested on an Intel NUC[5] with an i7 CPU@3.1 GHz and 8 GB memory. The average latency is reported in Table 5.4. We run the rosbag at a low frequency to ensure no data loss.

First of all, we observe that M-LOAM-rnd has lower latency than M-LOAM-gf in the GF-based data association. This is because the rnd is an $O(M)$ algorithm, while the stochastic-greedy algorithm is $O(N \log(1/\epsilon))$. Second, compared with M-LOAM-full, M-LOAM-gf may need more time for feature matching but significantly save time in nonlinear optimization. Finally, M-LOAM-gf, M-LOAM-rnd, and LEGO-LOAM are three real-time systems ($< 200ms$ at each mapping frame) for the Intel NUC. Both M-LOAM-gf and M-LOAM-rnd outperform LEGO-LOAM in terms of accuracy. LEGO-LOAM implicitly performs feature selection since it filters out points if the distances to their correspondences are larger than a threshold. But this naive and hard-coded solution leads to large accuracy loss.

5.7 Conclusion

In this thesis, we propose a greedy-based feature selection method for NLS pose estimation using LiDARs. The feature selector retains the most valuable LiDAR features with the objective of preserving the information matrix's spectrum. The stochastic-greedy algorithm is applied for the real-time selection. Moreover, we also investigate the degen-

[5] `zh.wikipedia.org/wiki/Next_Unit_of_Computing`

eracy issue of utilizing good features for pose estimation in structureless environments. We propose a strategy to adaptively change the number of good features to avoid ill-conditioned estimation. The feature selection is integrated into a multi-LiDAR SLAM system, followed by evaluation on sequences with two sensor setups and computation platforms. The enhanced system is shown to have great efficiency and higher localization accuracy than SOTA methods. The idea of feature selection is general and can be applied to many NLS problems.

Future work will concern two possible directions. The first direction is to utilize data-driven methods [186] to online tune parameters which were mannually set. Another direction is to apply the proposed feature selection to other tasks, such as bundle adjustment [213] and cross-model localization [73].

5.8 Supplementary Materials

5.8.1 Residuals and Jacobians of Edge and Planar Features

With the found correspondences between the reference frame and the robot's frame, we estimate the relative transformation by minimizing residual errors of all features. As illustrated in Fig. 5.7, the residuals stem from both edge and planar correspondences. Let the state vector $\mathbf{x}_K = [\mathbf{t}_K, \mathbf{q}_K]$ be the relative transformation at timestampe K, where $\mathbf{t}_K \in \mathbb{R}^3$ is the translation vector and $\mathbf{q}_K \in \mathbb{R}^4$ is the Hamilton quaternion. For an edge feature $\mathbf{p}$, if L is the corresponding edge line, the edge residual is computed as

$$\mathbf{r}_e(\mathbf{x}_K, \mathbf{p}, L) = \frac{||(\mathbf{p}' - \mathbf{p}_1) \times (\mathbf{p}' - \mathbf{p}_2)||}{||\mathbf{p}_1 - \mathbf{p}_2||}, \quad \mathbf{p}' = \mathbf{R}_K \mathbf{p} + \mathbf{t}_K, \tag{5.7}$$

where

$$\mathbf{R} = (q_w^2 - \mathbf{q}_{xyz}^\top \mathbf{q}_{xyz})\mathbf{I} + 2\mathbf{q}_{xyz}\mathbf{q}_{xyz}^\top + 2q_w\mathbf{q}_{xyz}^\wedge, \tag{5.8}$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are two side points of L, and the operator $(\cdot)^\wedge$ converts a 3×1 vector into a skew-symmetric matrix. Regarding a planar feature $\mathbf{p}$, if Π is the corresponding planar patch, the planar residual is computed as

$$\mathbf{r}_p(\mathbf{x}_K, \mathbf{p}, \Pi) = \mathbf{w}^\top \mathbf{p}' + d, \tag{5.9}$$

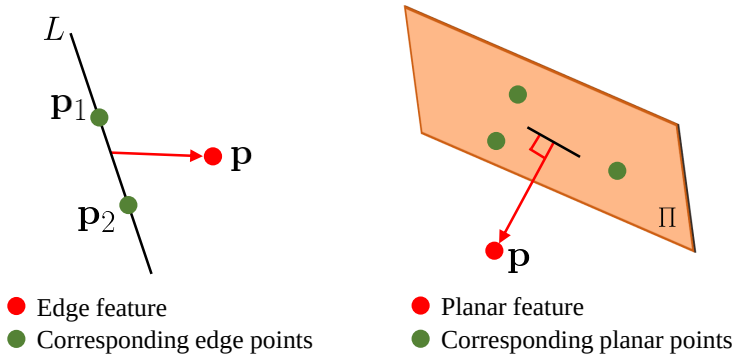


Figure 5.7: The edge and planar residuals. The red dot indicates the reference point and the green dots are its corresponding points.

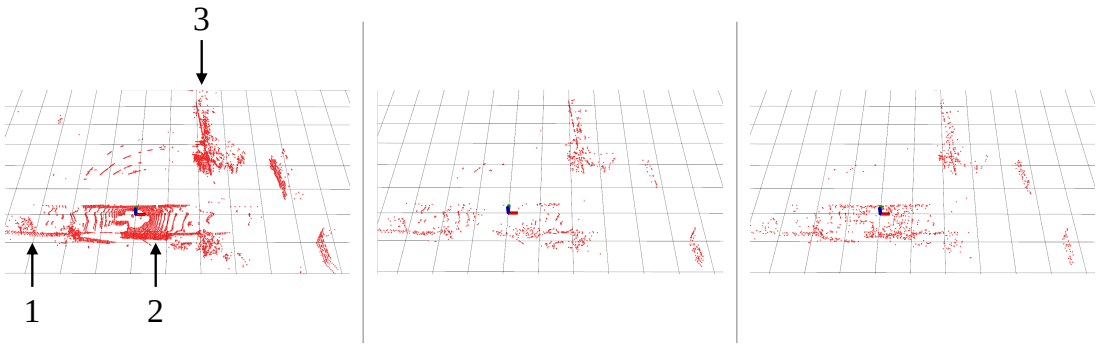


Figure 5.8: Qualitative results the stochastic-greedy and fully randomized feature selection methods. (Left) M-LOAM-full. (Middle) M-LOAM-gf. (Right) M-LOAM-rnd. We indicates three regions to show the differences between M-LOAM-gf and M-LOAM-rnd. In M-LOAM-gf, points are selected if they contribute to the spectrum of the information matrix more than others. But in M-LOAM-rnd, each point has the equal probability to be selected, so more features on the ground are picked up.

where $[\mathbf{w}, d]$ are the coefficients of Π. We minimize the sum of all residual errors to obtain the MLE with the good feature set as

$$\hat{\mathbf{x}}_K = \arg\min_{\mathbf{x}_K} \sum_{\mathbf{p} \in \mathcal{S}_K^{\#}} \left\| \mathbf{r}(\mathbf{x}_K, \mathbf{p}) \right\|^2_{\mathbf{\Sigma_p}}, \tag{5.10}$$

where the residual term is

$$\mathbf{r}(\mathbf{x}_K, \mathbf{p}) = \begin{cases} \mathbf{r}_e(\mathbf{x}_K, \mathbf{p}, L) & \text{if } \mathbf{p} \text{ is an edge feature} \\ \mathbf{r}_p(\mathbf{x}_K, \mathbf{p}, \Pi) & \text{if } \mathbf{p} \text{ is a planar feature} \end{cases}, \tag{5.11}$$

5.8.1.1 Jacobians of Edge Residuals

We first have two identities: $\mathbf{r} \times \mathbf{s} = \mathbf{r}^{\wedge}\mathbf{s}$ [7] and $\frac{\partial}{\partial \mathbf{x}}||\mathbf{x}|| = \frac{\mathbf{x}}{||\mathbf{x}||}$. Using the right perturbation: $\mathbf{R}\exp(\delta\boldsymbol{\phi}^{\wedge}) \approx \mathbf{R}(\mathbf{I} + \delta\boldsymbol{\phi}^{\wedge})$, the Jacobians of the edge residual are calculated

as

$$\begin{aligned}\frac{\partial \mathbf{r}_e(\mathbf{x}_K, \mathbf{p})}{\partial \mathbf{x}_K} &= [\frac{\partial \mathbf{r}_e(\mathbf{x}_K, \mathbf{p})}{\partial \mathbf{t}_K}, \ \frac{\partial \mathbf{r}_e(\mathbf{x}_K, \mathbf{p})}{\partial \mathbf{q}_K}] \\ &= [-\frac{1}{||\mathbf{p}_1 - \mathbf{p}_2||}\frac{\mathbf{r}^\top}{||\mathbf{r}||}(\mathbf{p}_1 - \mathbf{p}_2)^\wedge, \ \frac{1}{||\mathbf{p}_1 - \mathbf{p}_2||}\frac{\mathbf{r}^\top}{||\mathbf{r}||}(\mathbf{p}_1 - \mathbf{p}_2)^\wedge \mathbf{R}_K \mathbf{p}^\wedge],\end{aligned} \tag{5.12}$$

where $\mathbf{r} = \mathbf{r}_e(\mathbf{x}_{\text{op},K}, \mathbf{p})$ is the residual at an operating point. The quaternion is updated according to $\delta\mathbf{q} \approx [\frac{1}{2}\delta\boldsymbol{\phi}, 1]^\top$.

5.8.1.2 Jacobians of Planar Residuals

Using the left perturbation: $\mathbf{R}\exp(\delta\boldsymbol{\phi}^\wedge) \approx \mathbf{R}(\mathbf{I} + \delta\boldsymbol{\phi}^\wedge)$, the Jacobians of the planar residuals are calculated as

$$\frac{\partial \mathbf{r}_p(\mathbf{x}_K, \mathbf{p})}{\partial \mathbf{x}_K} = [\frac{\partial \mathbf{r}_p(\mathbf{x}_K, \mathbf{p})}{\partial \mathbf{t}_K}, \ \frac{\partial \mathbf{r}_p(\mathbf{x}_K, \mathbf{p})}{\partial \mathbf{q}_K}] = [\mathbf{w}^\top, \ -\mathbf{w}^\top \mathbf{R}_K \mathbf{p}^\wedge]. \tag{5.13}$$

5.8.2 Additional Experimental Results

5.8.2.1 Validation on Good Feature Selection

Fig. 5.8 presents the qualitative results of good features selected by the stochastic-greedy and the fully randomized method. The full features are also provided as a reference.

5.8.2.2 Performance of SLAM

As a complement to our manuscript, we present the complete quantitative results on OR sequences including: translation absolute trajectory error (ATE) and translation relative pose error (RPE) in Table 5.5.

Table 5.5: Translation ATE and RPE on OR sequences (the two best results are marked as bold text).

OR01 Method	Translation ATE	Translation RPE (per 137m)	Translation RPE (per 274m)	Translation RPE (per 412m)	Translation RPE (per 549m)	Translation RPE (per 686m)
M-LOAM-gf	**1.986** $\pm$ 0.013	2.674% $\pm$ 1.819%	2.654% $\pm$ 1.956%	2.604% $\pm$ 2.022%	2.075% $\pm$ 1.536%	**1.001**% $\pm$ 0.658%
M-LOAM-rnd	2.324 $\pm$ 0.028	**2.665**% $\pm$ 1.833%	**2.648**% $\pm$ 1.956%	2.609% $\pm$ 2.001%	2.089% $\pm$ 1.549%	1.015% $\pm$ 0.670%
M-LOAM-full	2.449 $\pm$ 0.014	2.668% $\pm$ 1.835%	2.650% $\pm$ 1.964%	2.614% $\pm$ 2.013%	2.097% $\pm$ 1.566%	**1.013**% $\pm$ 0.669%
A-LOAM	4.504 $\pm$ 0.055	3.670% $\pm$ 2.125%	3.186% $\pm$ 1.856%	2.961% $\pm$ 2.007%	2.272% $\pm$ 1.520%	1.251% $\pm$ 0.700%
F-LOAM	**2.174** $\pm$ 0.007	2.721% $\pm$ 1.900%	2.691% $\pm$ 1.971%	**2.123**% $\pm$ 1.631%	**1.531**% $\pm$ 1.037%	1.043% $\pm$ 0.701%
LEGO-LOAM	4.334 $\pm$ 0.000	**2.405**% $\pm$ 1.797%	**2.370**% $\pm$ 1.781%	**2.433**% $\pm$ 1.799%	**2.006**% $\pm$ 1.440%	1.069% $\pm$ 0.579%

OR02 Method	Translation ATE	Translation RPE (per 199m)	Translation RPE (per 399m)	Translation RPE (per 599m)	Translation RPE (per 799m)	Translation RPE (per 999m)
M-LOAM-gf	**2.473** $\pm$ 0.151	3.497% $\pm$ 2.628%	2.608% $\pm$ 2.104%	2.406% $\pm$ 1.984%	2.391% $\pm$ 1.950%	1.988% $\pm$ 1.664%
M-LOAM-rnd	**3.107** $\pm$ 0.104	**3.478**% $\pm$ 2.612%	**2.583**% $\pm$ 2.119%	2.376% $\pm$ 1.968%	2.385% $\pm$ 1.953%	2.003% $\pm$ 1.675%
M-LOAM-full	3.120 $\pm$ 0.076	3.468% $\pm$ 2.627%	2.587% $\pm$ 2.124%	2.379% $\pm$ 1.974%	2.396% $\pm$ 1.965%	2.006% $\pm$ 1.678%
A-LOAM	7.535 $\pm$ 0.410	4.748% $\pm$ 3.028%	3.744% $\pm$ 2.417%	3.238% $\pm$ 2.314%	2.973% $\pm$ 2.017%	2.478% $\pm$ 1.613%
F-LOAM	3.709 $\pm$ 0.044	3.487% $\pm$ 2.602%	3.001% $\pm$ 2.407%	**2.291**% $\pm$ 1.940%	**1.739**% $\pm$ 1.327%	**1.441**% $\pm$ 1.054%
LEGO-LOAM	4.369 $\pm$ 0.083	**3.209**% $\pm$ 2.450%	**2.340**% $\pm$ 1.869%	**2.123**% $\pm$ 1.678%	**2.149**% $\pm$ 1.687%	**1.808**% $\pm$ 1.414%

OR03 Method	Translation ATE	Translation RPE (per 190m)	Translation RPE (per 380m)	Translation RPE (per 570m)	Translation RPE (per 760m)	Translation RPE (per 950m)
M-LOAM-gf	**1.955** $\pm$ 0.105	1.913% $\pm$ 1.482%	2.193% $\pm$ 1.749%	1.956% $\pm$ 1.560%	1.836% $\pm$ 1.430%	1.875% $\pm$ 1.456%
M-LOAM-rnd	2.829 $\pm$ 0.158	**1.884**% $\pm$ 1.474%	2.158% $\pm$ 1.718%	1.921% $\pm$ 1.530%	**1.820**% $\pm$ 1.391%	**1.867**% $\pm$ 1.381%
M-LOAM-full	3.063 $\pm$ 0.062	1.885% $\pm$ 1.469%	**2.151**% $\pm$ 1.710%	**1.915**% $\pm$ 1.514%	**1.822**% $\pm$ 1.367%	1.872% $\pm$ 1.349%
A-LOAM	12.995 $\pm$ 0.006	7.507% $\pm$ 5.763%	7.496% $\pm$ 5.118%	6.641% $\pm$ 4.487%	6.117% $\pm$ 4.095%	5.570% $\pm$ 3.554%
F-LOAM	**2.019** $\pm$ 0.024	2.239% $\pm$ 1.727%	2.202% $\pm$ 1.753%	1.956% $\pm$ 1.570%	1.830% $\pm$ 1.423%	**1.756**% $\pm$ 1.372%
LEGO-LOAM	5.440 $\pm$ 0.102	**1.830**% $\pm$ 1.390%	**2.059**% $\pm$ 1.572%	**1.874**% $\pm$ 1.381%	1.894% $\pm$ 1.228%	1.983% $\pm$ 1.147%

OR04 Method	Translation ATE	Translation RPE (per 177m)	Translation RPE (per 354m)	Translation RPE (per 531m)	Translation RPE (per 708m)	Translation RPE (per 885m)
M-LOAM-gf	**1.720** $\pm$ 0.047	3.099% $\pm$ 1.657%	2.227% $\pm$ 1.352%	1.387% $\pm$ 0.961%	**1.070**% $\pm$ 0.805%	**0.866**% $\pm$ 0.641%
M-LOAM-rnd	2.242 $\pm$ 0.084	3.065% $\pm$ 1.649%	2.195% $\pm$ 1.360%	**1.380**% $\pm$ 0.961%	**1.081**% $\pm$ 0.823%	**0.898**% $\pm$ 0.673%
M-LOAM-full	2.239 $\pm$ 0.024	3.060% $\pm$ 1.644%	2.192% $\pm$ 1.354%	1.382% $\pm$ 0.959%	**1.081**% $\pm$ 0.823%	0.902% $\pm$ 0.671%
A-LOAM	**1.913** $\pm$ 0.057	3.094% $\pm$ 1.720%	2.290% $\pm$ 1.402%	1.414% $\pm$ 1.059%	1.142% $\pm$ 0.828%	0.936% $\pm$ 0.670%
F-LOAM	2.238 $\pm$ 0.062	**2.865**% $\pm$ 1.478%	**2.091**% $\pm$ 1.288%	**1.337**% $\pm$ 0.926%	1.078% $\pm$ 0.791%	0.915% $\pm$ 0.696%
LEGO-LOAM	6.224 $\pm$ 0.021	**2.878**% $\pm$ 1.484%	**2.090**% $\pm$ 1.324%	1.579% $\pm$ 0.943%	1.513% $\pm$ 0.792%	1.512% $\pm$ 0.703%

OR05 Method	Translation ATE	Translation RPE (per 198m)	Translation RPE (per 397m)	Translation RPE (per 596m)	Translation RPE (per 794m)	Translation RPE (per 993m)
M-LOAM-gf	**1.719** $\pm$ 0.027	2.502% $\pm$ 1.583%	1.994% $\pm$ 1.436%	1.941% $\pm$ 1.465%	1.787% $\pm$ 1.399%	1.769% $\pm$ 1.393%
M-LOAM-rnd	**2.387** $\pm$ 0.432	2.444% $\pm$ 1.581%	1.945% $\pm$ 1.426%	1.884% $\pm$ 1.431%	1.758% $\pm$ 1.372%	**1.738**% $\pm$ 1.368%
M-LOAM-full	2.567 $\pm$ 0.487	**2.417**% $\pm$ 1.593%	**1.929**% $\pm$ 1.430%	**1.875**% $\pm$ 1.434%	**1.755**% $\pm$ 1.376%	**1.736**% $\pm$ 1.369%
A-LOAM	6.446 $\pm$ 0.015	4.789% $\pm$ 3.222%	3.125% $\pm$ 1.865%	2.452% $\pm$ 1.609%	2.247% $\pm$ 1.566%	2.153% $\pm$ 1.497%
F-LOAM	2.469 $\pm$ 0.061	2.662% $\pm$ 1.516%	2.119% $\pm$ 1.421%	2.022% $\pm$ 1.455%	1.891% $\pm$ 1.417%	1.826% $\pm$ 1.395%
LEGO-LOAM	7.248 $\pm$ 0.260	**2.223**% $\pm$ 1.302%	**1.728**% $\pm$ 1.142%	**1.725**% $\pm$ 1.136%	**1.725**% $\pm$ 1.070%	1.787% $\pm$ 1.199%

Part III

Recognition

CHAPTER 6

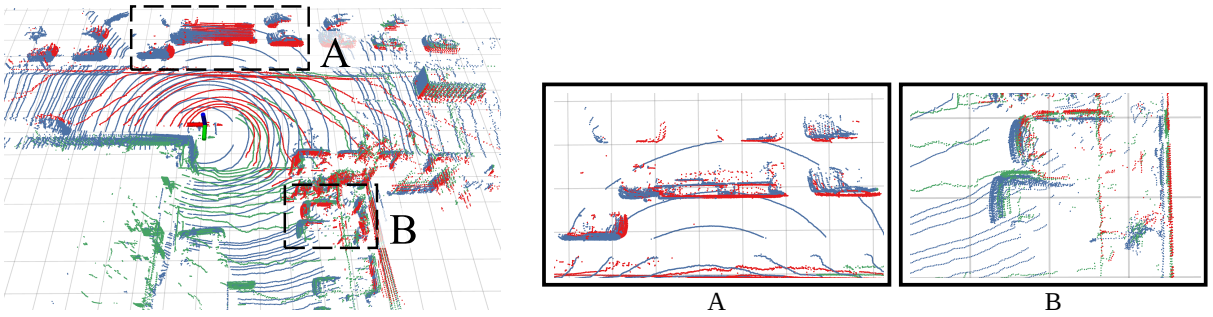
MLOD: AWARENESS OF EXTRINSIC PERTURBATION IN MULTI-LIDAR 3D OBJECT DETECTION FOR AUTONOMOUS DRIVING

Extrinsic perturbation always exists in multiple sensors. In this thesis, we focus on the extrinsic uncertainty in multi-LiDAR systems for 3D object detection. We first analyze the influence of extrinsic perturbation on geometric tasks with two basic examples. To minimize the detrimental effect of extrinsic perturbation, we propagate an uncertainty prior on each point of input point clouds, and use this information to boost an approach for 3D geometric tasks. Then we extend our findings to propose a multi-LiDAR 3D object detector called *MLOD*. *MLOD* is a two-stage network where the multi-LiDAR information is fused through various schemes in stage one, and the extrinsic perturbation is handled in stage two. We conduct extensive experiments on a real-world dataset, and demonstrate both the accuracy and robustness improvement of *MLOD*. The code, data and supplementary materials are available at: `https://ram-lab.com/file/site/mlod`.

6.1 Introduction

6.1.1 Motivation

3D object detection is a indispensable module for autonomous driving since it enables vehicles to recognize key objects on roads such as cars and pedestrians from sensor data. Benefiting from LiDARs' activenss in distance measuring and robustness to ambient light, LiDAR-based detectors perform better than camera-based detectors in most of driving scenarios [54]. But LiDARs also have their own limitations: data sparsity and limited vertical field of view (FOV) [79]. An important reason is that LiDARs' points distribute loosely, inducing a mass of undetectable regions between two nearby scans. Therefore, this thesis considers multi-LiDAR systems, which is an advanced choice to setup LiDARs and presents excellent potential for 3D object detection. Compared with a single LiDAR,



(a) Multiple point clouds are merged with little extrinsic perturbation.

(b) The details in the region A and B of the point cloud.

Figure 6.1: (a) A point cloud example from the LYFT dataset [89]. It is merged by transforming point clouds perceived by top (blue), front-left (green), and front-right (red) LiDARs into the base frame. We simulate perturbation on the rotation of 1° in yaw. (b) The details in the region A and B with a zooming effect. Massive noisy points are induced by extrinsic perturbation, while they should be sharp or flat in perturbation-free situations.

multi-LiDAR systems enable a vehicle to maximize its perceptual awareness of environments and obtain sufficient measurements. The decreasing price of LiDARs also makes such systems accessible to many modern self-driving cars [79, 83, 89, 116, 173].

6.1.2 Challenges

Two challenges should be addressed to develop the multi-LiDAR object detection. One issue is the fusion of multi-LiDAR data. Current data fusion schemes all have pros and cons using convolutional neural network [29] Therefore, we need to perform extensive efforts in real tests to find the most suitable fusion method. Another challenge is that the extrinsic perturbation is inevitable for sensor fusion during long-term operation because of factors such as vibration, temperature drift, and calibration error [115]. Especially, wide baseline stereo cameras or vehicle-mounted multi-LiDAR systems suffer even more extrinsic deviations than the normal one. Even though online calibration has been proposed to handle this issue [111], life-long calibration is still unsolved [123]. On the other hand, current calibration methods typically require environmental or motion constraints with full observability. Otherwise, the resulting extrinsics may become suboptimal or unreliable. As identified in both Fig. 6.1 and Section 6.3, extrinsic perturbation is detrimental to the measurement accuracy even with a small change. But this problem is often neglected by the research community.

6.1.3 Contributions

To tackle these challenges, we propose a **M**ulti-**L**iDAR **O**bject **D**etector (*MLOD*) to predict states of objects on point clouds. Here, the class of objects is restricted to cars to simplify the problem. We design a two-stage procedure to estimate 3D bounding boxes of objects, and demonstrate two innovations in *MLOD*. First, we explore three fusion schemes to exploit multi-view point clouds in a stage-1 network for object proposal generation. These schemes perform LiDAR fusion at the input, feature, and output phases, respectively. Second, we develop a stage-2 network to capture extrinsic perturbation and refine the proposals. The experimental results demonstrate that *MLOD* outperforms single-LiDAR detectors up to 9.7AP in perturbation-free cases. When considering extrinsic perturbation, *MLOD* consistently improves the performance of its stage-1 counterpart. To the best of our knowledge, this is the first work to systematically study the multi-LiDAR object detection with consideration of extrinsic perturbation. Overall, our work has the following contributions:

1. We analyze the influence of extrinsic perturbation on multi-LiDAR-based geometric tasks, and also demonstrate that the usage of input uncertainty prior improves the robustness of an approach against such effect.

2. We propose a unified and effective two-stage approach for multi-LiDAR 3D object detection, where the multi-LiDAR information is fused in the first stage, and extrinsic perturbation is handled in the second stage.

3. We exhaustively evaluate the proposed approach in terms of accuracy and robustness under extrinsic perturbation on a real-world self-driving dataset.

4. We release the implementation code.[1]

6.2 Related Work

We briefly review methods on LiDAR-based object detection and uncertainty estimation of deep neural networks.

[1] `http://143.89.78.112:5000/sharing/90BpyDIuq`

6.2.1 3D Object Detection on Point Clouds

The LiDAR-based object detectors are generally categorized into grid-based [105, 192, 217] and point-based methods [25, 142, 167]. Li et al. proposes 3D-FCN [105] to apply 3D volumetric CNN on voxelized point clouds. But it commonly suffers a high computation cost due to dense convolution on sparse point clouds. Zhou et al. proposes VoxelNet [217], which is an end-to-end network, to learn features. But both of these methods commonly suffer a high computational cost using dense convolution. In contrst, SECOND [192] utilizes the spatially sparse convolution [58] to replace the 3D dense convolution layers. In this thesis, we adopt SECOND as our basic proposal generator and propose three schemes for multi LiDAR fusion in the first stage.

Regarding point-based methods, Qi et al. proposed PointNet [25] to learn features from the raw unordered point set with a symmetric function. And their follow-up work presents a set-abstraction block to capture local features [142]. Shi et al. presents PointRCNN [167] which exploits PointNet to learn point-wise features and extract foreground points for autonomous driving. In this thesis, we extend PointNet to design the stage-2 network of *MLOD* with an awareness of extrinsic perturbation.

Researchers also explored fusion methods that further exploit images to improve the LiDAR-based approaches. For example, MV3D [29] combines features from multiple views, including a bird-eye view and front view, to conduct classification and regression. Several methods [141, 190] separate the process of detection into two stages. They first projected region proposals generated by the camera-based detectors into 3D space, and then used PointNets to segment and to classify objects on point clouds. But these methods are aimed at data fusion from cameras and LiDARs and assume sensors to be well-calibrated. In contrast, our work focuses on multi-LiDAR fusion, and tries to minimize the negative effect of extrinsic perturbation for 3D object detection.

6.2.2 Uncertainty Estimation in Object Detection

The problem of uncertainty estimation is essential to the reliability of an algorithm, and has attracted much attention in recent years. As the two main types of uncertainties in deep neural networks: aleatoric and epistemic uncertainty, they are explained in [87,88]. In [87], the authors also demonstrated the benefits of modeling uncertainties for vision tasks. As an extension, Feng et al. [45] proposed an approach to capture uncertainties for

Table 6.1: Nomenclature

Notation	Explanation
$\{\}^b, \{\}^{l^i}$	Frame of the base and the i^{th} LiDAR.
$\mathcal{P}^{l^i}$	Raw point cloud captured by the i^{th} LiDAR.
$\mathcal{B}$	Set of estimated 3D bounding boxes.
$\mathbf{T}$	Transformation matrix in the *Lie group* $SE(3)$.
$\mathbf{R}$	Rotation matrix in the *Lie group* $SO(3)$.
$\mathbf{t}$	Translation vector in $\mathbb{R}^3$.
$\mathbf{\Theta}$	Measurement and extrinsic uncertainty prior.
$\mathbf{\Xi}$	Propagated covariance of each point.
α	Scaling parameter of $\mathbf{\Theta}$.

3D object detection. Generally, the data noise is modeled as a unique Gaussian variable in these works. However, the sources of data uncertainties in multi-LiDAR systems are much more complicated. In this thesis, we take both the measurement noise and extrinsic perturbation into account, and analyze their effect on multi-LiDAR-based object detection. Furthermore, we propagate the Gaussian uncertainty prior of both the extrinsics and measurement to model the input data uncertainty. This additional cue is utilized to improve the robustness of *MLOD* against extrinsic perturbation.

6.3 Notations and Problem Statement

The nomenclature is shown in Tab. 6.1. The transformation from $\{\}^b$ to $\{\}^{l^i}$ is denoted by $\mathbf{T}^b_{l^i}$. Our perceptual system consists of one primary LiDAR which is denoted by l^1 and multiple auxiliary LiDARs. The primary LiDAR commonly enjoys the largest FOV and thus defines the base frame. The auxiliary LiDAR compensates the lack of FOV to alleviate the occlusion problem and provids additional measurement to solve the sparsity drawback of one LiDAR. Extrinsics describe the spatial offsets from the base frame to frames of auxiliary LiDARs. With the extrinsics, all measurements or features from different LiDARs can be transformed into the base frame for data fusion. LiDARs are assumed to be synchronized that the relatively latency of multiple point clouds is tiny.

In this thesis, we focus on 3D object detection task for a multi-LiDAR system. We also consider the extrinsic perturbation, which means that the system may suffer the small but unexpected change on transformation over time. Our goal is to estimate a series of 3D bounding boxes covering objects of predefined classes. Each bounding box

$\mathbf{b} \in \mathcal{B}$ is parameterized as $[c, x, y, z, w, l, h, \gamma]$, c is the class of a bounding box, $[x, y, z]$ denotes a box's bottom center, $[w, h, l]$ represent the sizes along the $x-$, $y-$, and $z-$ axes respectively, as well as γ for the rotation of the 3D bounding box along the $z-$ axis.

6.4 Propagating Extrinsic Uncertainty on Points

We start with introducing preliminaries about the uncertainty representation and propagation. We then use an example to explain the negative effect on points caused by extrinsic perturbation. Finally, we also conduct a plane fitting test to show that the covariance prior can be utilized to make a fitting approach robust.

6.4.1 Uncertainty Representation and Propagation

We use the rotation matrix and translation to indicate a transformation matrix. We define a random variable for $\mathbb{R}^3$ with small perturbation according to

$$\mathbf{t} = \boldsymbol{\rho} + \bar{\mathbf{t}}, \quad \boldsymbol{\rho} \sim \mathcal{N}(\mathbf{0}, \mathbf{P}), \tag{6.1}$$

where $\bar{\mathbf{t}}$ is a noise-free translation and $\boldsymbol{\rho} \in \mathbb{R}^3$ is a zero-mean Gaussian variable with covariance $\mathbf{P}$. We can also define a rotation for $SO(3)$ as[2]

$$\mathbf{R} = \exp(\boldsymbol{\phi}^\wedge)\bar{\mathbf{R}}, \quad \boldsymbol{\phi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Phi}), \tag{6.2}$$

where $\bar{\mathbf{R}}$ is a noise-free rotation and $\boldsymbol{\phi} \in \mathbb{R}^3$ is a small zero-mean Gaussian variable with the covariance $\boldsymbol{\Phi}$. A point with the perturbation in $\mathbb{R}^3$ is written as

$$\mathbf{p} = \boldsymbol{\zeta} + \bar{\mathbf{p}}, \quad \boldsymbol{\zeta} \sim \mathcal{N}(\mathbf{0}, \mathbf{Z}), \tag{6.3}$$

where $\boldsymbol{\zeta}$ is zero-mean Gaussian with the covariance $\mathbf{Z}$. We can use $[\bar{\mathbf{t}}_{li}^b, \bar{\mathbf{R}}_{li}^b]$ to indicate the ground-truth extrinsics of a multi-LiDAR system, and $[\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\zeta}]$ to indicate both the extrinsic and measurement perturbation. By transforming $\mathbf{p} \in \mathcal{P}^i$ into $\{\}^b$, we have

$$\begin{aligned} \mathbf{y} \triangleq \mathbf{R}_{li}^b \mathbf{p} + \mathbf{t}_{li}^b &= \exp(\boldsymbol{\phi}^\wedge)\bar{\mathbf{R}}_{li}^b(\boldsymbol{\zeta} + \bar{\mathbf{p}}) + (\boldsymbol{\rho} + \bar{\mathbf{t}}_{li}^b) \\ &\approx \Big(\mathbf{I} + \boldsymbol{\phi}^\wedge\Big)\bar{\mathbf{R}}_{li}^b(\boldsymbol{\zeta} + \bar{\mathbf{p}}) + (\boldsymbol{\rho} + \bar{\mathbf{t}}_{li}^b), \end{aligned} \tag{6.4}$$

[2]The $\wedge$ operator turns a 3×1 vector into the corresponding skew symmetric matrix in the *Lie algebra* $\mathfrak{so}(3)$. The exponential map exp associates an element of $\mathfrak{so}(3)$ to a rotation in $SO(3)$. The closed-form expression for the matrix exponential can be found in [7].

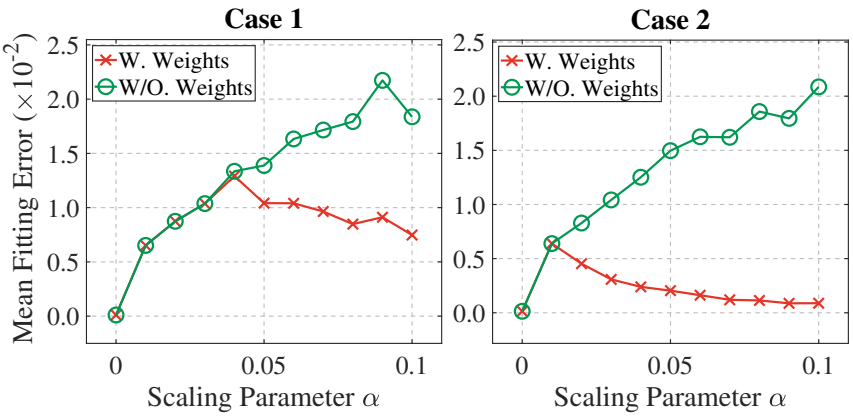


Figure 6.2: Mean fitting error $e_{\mathbf{x}} = ||\mathbf{x}_{gt} - \mathbf{x}_{est}||$ of methods with weights (use uncertainty prior) or without weights on two planar surface cases. The weighted method does better, and the error does not increase along with α.

where we have kept the first-order approximation of the exponential map. If we multiply out the equation and retain only those terms that are first-order in $\boldsymbol{\phi}$ or $\boldsymbol{\zeta}$, we have

$$\mathbf{y} \approx \mathbf{h} + \mathbf{H}\boldsymbol{\theta}, \tag{6.5}$$

where

$$\boldsymbol{\theta} = [\boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\zeta}], \quad \mathbf{h} = \bar{\mathbf{R}}^b_{li}\bar{\mathbf{p}} + \bar{\mathbf{t}}, \quad \mathbf{H} = [\mathbf{I} \quad -(\bar{\mathbf{R}}^b_{li}\bar{\mathbf{p}})^{\wedge} \quad \bar{\mathbf{R}}^b_{li}], \tag{6.6}$$

We embody the uncertainties of extrinsics and measurements into $\boldsymbol{\theta}$ that is subjected to a zero-mean Gaussian with 9×9 covariance $\boldsymbol{\Theta} = \text{diag}(\mathbf{P}, \boldsymbol{\Phi}, \mathbf{Z})$. Linearly transformed by $\boldsymbol{\theta}$, $\mathbf{y}$ is a Gaussian variable with mean and covariance as

$$\boldsymbol{\mu} \triangleq E[\mathbf{y}] = \mathbf{h}, \quad \boldsymbol{\Xi} \triangleq E\big[(\mathbf{y} - \boldsymbol{\mu})(\mathbf{y} - \boldsymbol{\mu})^{\top}\big] = \mathbf{H}\boldsymbol{\Theta}\mathbf{H}^{\top}. \tag{6.7}$$

where we follow [91] to use the trace, i.e., $\text{tr}(\boldsymbol{\Xi})$, as the criterion to quantify the magnitude of a covariance.

6.4.2 A Toy Example

In this section, a simple example of uncertainty propagation on a point is presented, where we quantify the data noise caused by both extrinsic and measurement perturbation. The associated covariance is propagated by passing the Gaussian uncertainty from extrinsics and measurements through a transformation. We set ground-truth extrinsics as with $[10, 10, 10]^{\circ}$ in roll, pitch, yaw and $[1, 1, 1]m$ along the $x-$, $y-$, and $z-$ axis respectively.

We set $\mathbf{p} = [10, 10, 10]^\top m$ as a point, and $\mathbf{\Theta}$ as a constant matrix[3]

$$\mathbf{\Theta} = \text{diag}^2\left\{\frac{1}{20}, \frac{1}{20}, \frac{1}{20}, \frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \frac{1}{50}, \frac{1}{50}, \frac{1}{50}\right\}, \tag{6.8}$$

where the first six diagonal entries can be multiplied by a scaling parameter α, allowing us to parametrically increase the extrinsic covariance in experiments. According to (6.7), we have an approximate expression

$$\mathbf{\Xi} \approx \alpha \times \begin{bmatrix} 2.36 & -1.24 & -1.20 \\ -1.24 & 2.41 & -1.18 \\ -1.20 & -1.18 & 2.49 \end{bmatrix}, \tag{6.9}$$

where the new position has a high deviation around $0.22m$ on each axis even for a small input covariance (i.e., $\alpha = 0.02$). This accuracy is totally unacceptable for autonomous driving.

6.4.3 Covariance Prior-Enhanced Plane Fitting

We denote $\mathcal{P}_m$ a merged point cloud which is obtained by transforming $\mathcal{P}^{l^1}$ and $\mathcal{P}^{l^2}$ into $\{\}^b$ with ground-truth extrinsics. We assume that there is a dominant plane in $\mathcal{P}_m$, and our task is to estimate the plane coefficients. They can be obtained by solving a linear system $\mathbf{Ax} = \mathbf{b}$ from $\mathcal{P}_m$, where row elements of $\mathbf{A}$ are the point coordinates, and $\mathbf{b}$ is set as an identity vector with least-squares methods.

Here, we show how the input uncertainty prior $\mathbf{\Theta}$ is utilized to acquire better fitting results. We define $\mathbf{W}$ as a diagonal matrix to weight the linear system. For a point $\mathbf{p}_j \in \mathcal{P}_m$, we propagate its associated covariance $\mathbf{\Xi}$ using (6.7). The corresponding entry (i.e., $\mathbf{W}_{jj}$) is set as the inverse of $\text{tr}(\mathbf{\Xi})$. After that, the fitting problem is turned into a weighted least-squares regression and the optimal results are solved: $\mathbf{x}^* = (\mathbf{A}^\top\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{W}\mathbf{b}$.

An experiment is conducted on a toy example to compare the performance of two fitting methods (i.e., with or without weights). We use the ground-truth extrinsics and the prior covariance in Section 6.4.2 to sample perturbation with 100 trials. We generate 10000 points which are subjected to zero-mean Gaussian noise with a standard deviation of $0.02m$ to produce $\mathcal{P}_m$ from a planar patch. $\mathcal{P}_m$ is randomly split into two equal parts

[3]The measurement noise is found in LiDAR datasheets. The extrinsic perturbation typically has the variances around $5cm$ in translation and $5°$ in rotation. This is based on our studies on multi-LiDAR systems [79, 116].

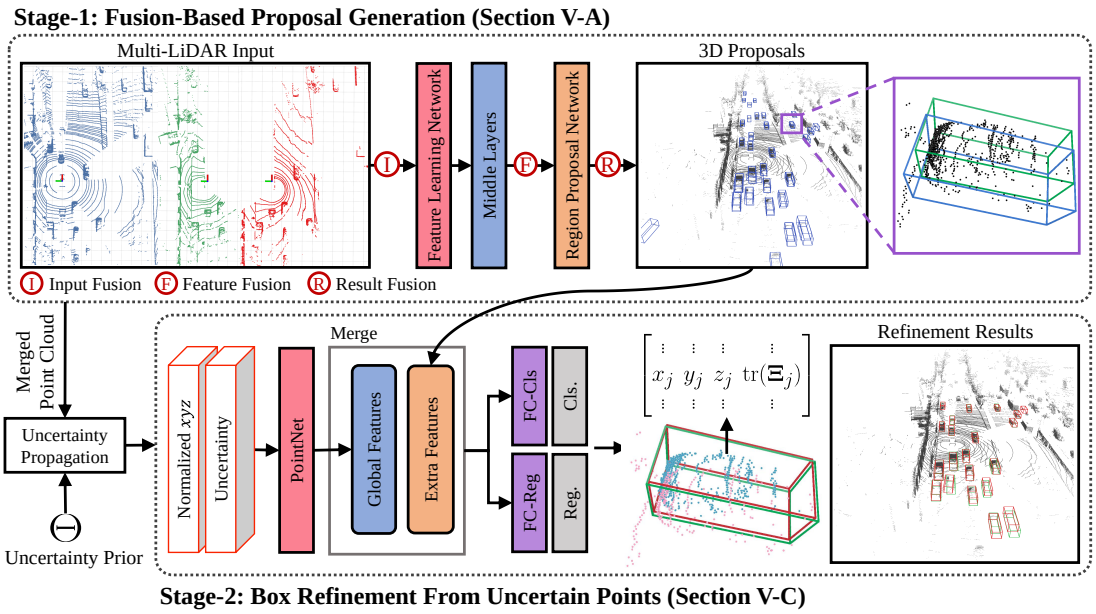


Figure 6.3: Overview of the proposed detector (*MLOD*) for multi-LiDAR 3D object detection. Red circles: the position where the fusion may be performed; Blue cubes: proposals in stage one; Red cubes: refinements in stage two; Green cubes: ground truths. We used the score and parameter of a proposal from the stage-1 network as the extra features. Global features are extracted by PointNet [25]. They are merged as the input for final classification and regression. For simplicity, we consider three LiDARs as an example; however, *MLOD* can be extensible to more.

to form $\mathcal{P}^{l^1}$ and $\mathcal{P}^{l^2}$. At each trial, we evaluate the fitting results of each method by comparing them to the ground truth as $e_{\mathbf{x}} = ||\mathbf{x}_{gt} - \mathbf{x}_{est}||$.

The mean fitting error on two different cases over $\alpha \in [0, 0.1]$ is shown in Fig. 6.2. We see that the weighted least-squares method does better in estimating the plane coefficients, and the mean fitting error does not increase along with α. It shows that the point-wise uncertainty information boosts algorithms' robustness in geometric tasks.

6.4.4 Reasoning

The above process of uncertainty propagation enables the mechanical perturbation on sensors, a practical issue in robotics, to be quantitatively analyzed. Current fusion-based approaches can apply this technique to evaluate the uncertainty of a point among all measurements and reduce outliers. But in practice, the value of α should be manually set or adaptively obtained from an online method [83]. Otherwise, a few correct measurements are discarded during operation.

6.5 Methodology

In this section, we extend our findings from the basic geometric tasks to multi-LiDAR-based 3D object detection. The overall structure of the proposed two-stage *MLOD* is illustrated in Fig. 6.3. We adopt SECOND [192], which is a sparse convolution improvement of VoxelNet [217], to generate 3D proposals in the first stage. [4] It consists of three components: a feature learning network (**FLN**) for feature extraction; middle layers (**ML**) for feature embedding with sparse convolution; and a region proposal network (**RPN**) for box prediction and regression. Readers are referred to [192,217] for more details about the network structures. We first present three species of 3D proposal generation with different multi-LiDAR fusion schemes: *Input Fusion*, *Feature Fusion* and *Result Fusion*. Then, we introduce the architecture of our stage-2 network to tackle the extrinsic uncertainty and refine the proposals.

6.5.1 Proposal Generation With Three Fusion Schemes

According to stages at which the information from multiple LiDARs is fused, we propose three general fusion schemes, which we call *Input Fusion*, *Feature Fusion*, and *Result Fusion*. These approaches are developed from SECOND with several modifications.

6.5.1.1 Input Fusion

The fusion of point clouds is performed at the input stage. We transform raw point clouds perceived by all LiDARs into the base frame to obtain the fused point cloud, and then feed it to the network as the input.

6.5.1.2 Feature Fusion

To enhance the feature interaction, the LiDAR data is also fused in the feature level. The extracted features from the FLN and ML are transformed into the base frame, and then fused by adopting the maximum value as

$$\mathcal{F}_{\text{fused}} = \mathcal{F}^{l^1} \oplus (\mathbf{T}^b_{l^2}\mathcal{F}^{l^2}) \oplus \cdots \oplus (\mathbf{T}^b_{l^I}\mathcal{F}^{l^I}), \tag{6.10}$$

[4] We use SECOND V1.5 in our experiments: `https://github.com/traveller59/second.pytorch`. Different from the original SECOND [192], the FLN of SECOND V1.5 is simplified by computing the mean value of points within each voxel, which consumes less memory.

where $\mathcal{F}^{l^i}$ are the extracted features, $\oplus$ denotes the max operator, and I is the number of LiDARs.

6.5.1.3 Result Fusion

The result fusion takes the box proposals and the associated points as inputs, and produces a set of boxes with high scores. We transform all box proposals into the base frame, and then filter them as

$$\mathcal{B}_{\text{fused}} = \mathcal{B}^{l^1} \oplus (\mathbf{T}^b_{l^2}\mathcal{B}^{l^2}) \oplus \cdots \oplus (\mathbf{T}^b_{l^I}\mathcal{B}^{l^I}), \tag{6.11}$$

where $\oplus$ denotes the non-maximum suppression (NMS) on 3D intersection-over-Union (IoU). After transformation, each object is associated with several candidate boxes, and these boxes with low confidences are filtered by the NMS.

6.5.2 Box Refinement From Uncertain Points

Although the stage-1 fusion-based network generates promising proposals, its capability in handling uncertain data uncertainty is fragile. Therefore, we propose a stage-2 module to improve the robustness of *MLOD*. A straight-forward idea is to eliminate the highly uncertain points according to their associated covariances. But this method is sensitive to a pre-set threshold. Inspired by [167], it is more promising that training a neural network to embed features and refine the proposals with an awareness of uncertainty.

We thus use a deep neural network to deal with this problem. The network takes a series of points of each proposal (a $2m$ margin along $x-$, $y-$, and $z-$ axes is expanded) generated by the stage-1 network, and refines the stage-1 proposals. In addition to three-dimensional coordinates, each point is also embedded with its uncertain quantity. As defined in (6.7), we use the trace, i.e., $\text{tr}(\mathbf{\Xi})$, to quantify the uncertainties. We employ PointNet [25] as the backbone to extract global features. We also encode extra features, including the scores and parameters of each proposal, to concatenate with the global features. At the end of the network, fully connected layers are used to provide classification scores and refinement results with two heads. To reduce the variance of input, we normalize the coordinates of each point of a proposal as

$$\mathbf{p}_{\text{n}} = \mathbf{T}\mathbf{p} \odot \mathbf{s}, \tag{6.12}$$

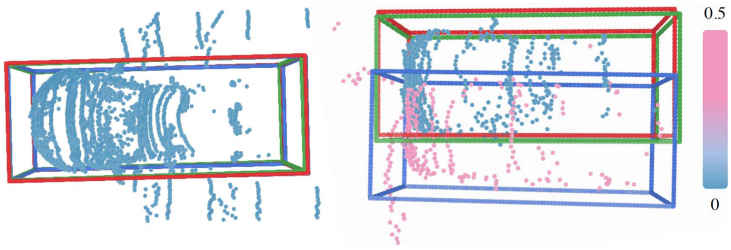


Figure 6.4: Illustration of the *MLOD* results on the proposals and merged point cloud given by the stage-1 network. The color of each point represents its uncertainty quantity defined as the trace of the associated covariance. Blue to pink color indicates low to high uncertainty. The estimated and ground-truth bounding boxes are also marked with different colors: blue as the proposals in stage one, red as refinements in stage two, and green as ground truths.

where

$$\mathbf{T} = \begin{bmatrix} cos(\gamma) & sin(\gamma) & 0 & -x \\ -sin(\gamma) & cos(\gamma) & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} 1/l & 1/w & 1/h & 1 \end{bmatrix}^\top, \tag{6.13}$$

where $\mathbf{T}$ and $\mathbf{s}$ are defined in terms of the parameters of a object proposal, $\odot$ is the Hadamard product, and both $\mathbf{p}_\text{n}$ and $\mathbf{p}$ are represented in homogeneous coordinates. The output of the classification head is a binary variable indicating the probability of objectiveness. We regress residuals of the bounding box based on the proposal instead of directly regressing the final 3D bounding box. The regression target of the stage-2 network is

$$\mathbf{u}_i \triangleq \big[\frac{x - x_p}{l_p}, \frac{y - y_p}{w_p}, \frac{z - z_p}{h_p}, \frac{l - l_p}{l_p}, \frac{w - w_p}{w_p}, \frac{h - h_p}{h_p}, \sin(\gamma - \gamma_p), \cos(\gamma - \gamma_p)\big], \tag{6.14}$$

where $\mathbf{u}_i \in \mathbb{R}^8$ are regression outputs for the i^{th} positive proposal respectively. We adopt the classification and regression loss which are defined in [201] as

$$\mathcal{L}_\text{cls} = \sum_i f_\text{cls}(p_i), \quad \mathcal{L}_\text{reg} = \sum_i f_\text{reg}(\mathbf{u}_i, \mathbf{u}_i^*), \tag{6.15}$$

where p_i represents the posterior probability of objectiveness, $\mathcal{L}_\text{cls}$ is defined as the classification loss, $f_\text{cls}(\cdot)$ denotes the focal loss [114], $\mathcal{L}_\text{reg}$ is defined as the normalized regression loss, and $f_\text{reg}(\cdot)$ denotes the smooth-L1 loss. Since we have the observation that the regression outputs $\mathbf{u}_i$ should not be large when the uncertainty is small, we add a regularization term to penalize large $\mathbf{u}_i$ in the low uncertainty cases:

$$\mathcal{L}_\text{uct} = \sum_i \exp\left(1 - f\big[\underset{j}{MAX}\ \text{tr}(\mathbf{\Xi}_j)\big]\right) \cdot ||\mathbf{u}_i'||_2, \tag{6.16}$$

where $\text{tr}(\mathbf{\Xi}_j)$ is the point-wise uncertainty of the j^{th} point within the i^{th} proposal, MAX is the max operator, $\mathbf{u}_i^{'}$ is the regressed residual without the last cosine term for the the i^{th} proposal, and $f(\cdot)$ clamps a value into the range $[10^{-3}, 0.5]$ to stabilize training. Finally, we define the loss function $\mathcal{L}$ for training the stage-2 network as

$$\mathcal{L} = \mathcal{L}_{\text{reg}} + \ \eta\mathcal{L}_{\text{cls}} + \ \lambda\mathcal{L}_{\text{uct}}, \tag{6.17}$$

where η and λ are hyper-parameters to balance the weight of the classification loss and the uncertainty regularizer. We set $\eta = 2$ and $\lambda = 0.005$ in our experiments.

Fig. 6.4 visualizes the refinement results of our stage-2 network, which shows that our method produces similar results to the ground truths even if many points are uncertain. More quantitative examples are provided in Section 6.6.3.

6.6 Experiment

In this section, we evaluate our proposed *MLOD* on LYFT multi-LiDAR dataset [89] in terms of accuracy and robustness under different levels of extrinsic perturbation. In particular, we aim to answer the following questions:

1. Can a multi-LiDAR object detector perform better accuracy than single-LiDAR methods?

2. Can *MLOD* improve the robustness of a multi-LiDAR object detector under extrinsic perturbation?

6.6.1 Implementation Details

6.6.1.1 Dataset

The LYFT dataset [89] provides a large amount of data collected in a variety of environments for the task of 3D object detection. Three 40-beam and calibrated Hesai LiDARs are mounted on the top, front-left and front-right position of the vehicle platform. The top LiDAR is set as the primary LiDAR, which is denoted by l^1. The left and right LiDARs are auxiliary LiDARs, which are denoted by l^2 and l^3 respectively. This setting is according to LiDAR's field of views (FOV). We select all multi-LiDAR data samples for our experiment. The data contain **4031** samples, **2500** of which are for training as

well as validation and **1531** of which are for testing. The testing scenes are different from those in the training and validation sets.

6.6.1.2 Metric

Following the KITTI evaluation metrics [54], we compute the average precision on 3D bounding boxes (AP_{3D}) to measure the detection accuracy. We compute the AP_{3D} for 360° around the vehicle instead of evaluating only the 90° front view. According to the distance of objects, we set three different evaluation difficulties: *easy* ($< 20m$), *moderate* ($< 30m$), and *hard* ($< 50m$).

6.6.1.3 Extrinsic Perturbation Injection

We inject extrinsic perturbation on the original LYFT dataset to generate another set of data for robustness evaluation. We take $\mathbf{\Theta}$ defined in (6.8) and adjust $\alpha \in [0, 0.1]$ with a 0.02 interval to simulate different levels of perturbation. To remove the effects from outliers, we bound the sampled perturbation within the σ position. Although the maximum value of α is small, but the effect of input perturbation on points is obvious. The noisy extrinsics are obtained by adding the sampled perturbation to the ground-truth extrinsics according to (6.1) and (6.2).

6.6.1.4 Baseline Declaration

We declare all cases of our methods which are tested in the following experiments as

- *LiDAR-Top*, *LiDAR-Left*, *LiDAR-Right*: single-LiDAR objeect detectors based on SECOND with the input data which are captured by the top LiDAR, front-left LiDAR and front-right LiDAR, respectively.
- *Input Fusion*, *Feature Fusion*, *Result Fusion*: multi-LiDAR objeect detetors with different fusion schemes, which are introduced in Section 6.5.1.
- *MLOD-I*, *MLOD-F*, *MLOD-R*: *MLOD* with *Input Fusion*, *Feature Fusion*, *Result Fusion* as its stage-1 network respectively, which are described in Section 6.5.2.

- *MLOD-I (OC)*, *MLOD-F (OC)*, *MLOD-R (OC)*: Variants of *MLOD* with an online calibration method to reduce extrinsic perturbation. This method is implemented with a point-to-plane ICP approach [140].

6.6.1.5 Training of Stage-One Network

We follow [192] to train the stage-1 network. During training, we conduct dataset sampling as in [192] and an augmentation of random flips on the $y-$ axis as well as translation sampling within $[-0.5, 0.5]m$, $[-0.5, 0.5]m$, $[-0.3, 0.3]m$ on the $x-$, $y-$, and $z-$ axes respectively, as well as rotation around the $z-$ axis between $[-5, 5]^\circ$. In each epoch, the augmented data accounts for 30% of the whole training data. In *Result Fusion* and *Feature Fusion*, we directly take the calibrated and merged point clouds from different LiDARs as input. In *Result Fusion*, we calibrate all the proposal generators with temperature scaling [60].

6.6.1.6 Training of Stage-Two Network

To train our stage-2 network, we use all cases of the well-trained stage-1 networks to generate proposals. We use the training set to create proposals and inject uncertainties by selecting different α like the above section. In addition, we force the ratio between uncertainty-free samples and uncertain samples to be about 1:1.5 to stabilize the training process. A proposal is considered to be positive if its maximum IoU with ground-truth boxes is above 0.6, and is treated as negative if its maximum 3D IoU is below 0.45. During training, we conduct data augmentation of random flipping, scaling with a scale factor sampled from $[0.95, 1.05]$, translation along each axis between $[-0.1, 0.1]m$ and rotation around the z- axis between $[-3, 3]^\circ$. We also randomly sample 1024 points within the proposals to increase their diversity.

6.6.2 Results on the Multi-LiDAR LYFT Dataset

We evaluate both single-LiDAR and multi-LiDAR detectors on the LYFT test set, as reported in Table 6.2. In this experiment, we do not consider any extrinsic perturbation between LiDARs. *LiDAR-Top* performs the best among all single-LiDAR detectors. This is because the top LiDAR has a complete 360° horizontal field of view (FOV). The left

Table 6.2: Average Precision on the multi-LiDAR LYFT test set

Case	AP_{3D} IoU $\geqslant 0.7$			AP_{3D} IoU $\geqslant 0.5$		
	easy	mod.	hard	easy	mod.	hard
LiDAR-Top	**62.7**	**52.9**	**37.8**	**89.8**	**80.1**	**61.8**
LiDAR-Left	41.1	37.0	27.5	62.6	53.9	43.1
LiDAR-Right	40.9	31.5	23.4	53.9	52.4	35.5
Input Fusion	71.9	61.8	45.0	89.4	87.9	**61.9**
MLOD-I	**72.4**	**62.7**	**45.6**	**89.6**	**88.3**	61.8
Feature Fusion	71.1	60.2	44.1	89.6	87.4	**61.6**
MLOD-F	**72.0**	**61.8**	**44.9**	**89.7**	**88.2**	**61.6**
Result Fusion	71.4	61.6	44.5	89.5	87.7	**62.0**
MLOD-R	**72.1**	**62.3**	**45.2**	**89.6**	**88.0**	61.7

and right LiDARs, which are partially blocked by the vehicle, only have a 225° horizontal FOV. With sufficient measurements and decreasing occlusion areas, all multi-LiDAR detectors outperform *LiDAR-Top*, and the accuracy improvement gains up to **9.8**AP. Table 6.2 also shows that all *MLOD* variants perform better or comparable to their one-stage counterparts. This proves that our stage-2 network refines the stage-1 proposals in perturbation-free cases.

Single-LiDAR detectors roughly cost $40ms$ for each frame, while *Input Fusion*, *Feature Fusion*, and *Result Fusion* (the stage-1 network) cost around $50ms$, $45ms$, and $57ms$, respectively. The stage-2 network mainly costs $20ms$ for each frame. Thus, the overall network can run at real time.

6.6.3 Robustness Under Extrinsic Perturbation

In this section, we evaluate the robustness of our proposed methods under extrinsic perturbation. Three levels of perturbation are performed: no perturbation ($\alpha = 0$), moderate perturbation ($\alpha = 0.02$, $\alpha = 0.04$) and high perturbation ($\alpha = 0.1$). An example is displayed in Fig. 6.7(a) ($\alpha = 0.04$), where massive noisy points ($50cm$ misplacement) appear at $35m$ away. This is a typical phenomenon of extrinsic perturbation in multi-LiDAR systems. We randomly select **300** data samples from the test set with 10 trials to conduct evaluations at each level.

In Fig. 6.5, the detection results in terms of the means and variances as AP_{3D}(IoU $\geqslant$ 0.7) are detailed. We see that all variants of *MLOD* (*MLOD-I*, *MLOD-F*, *MLOD-R*)

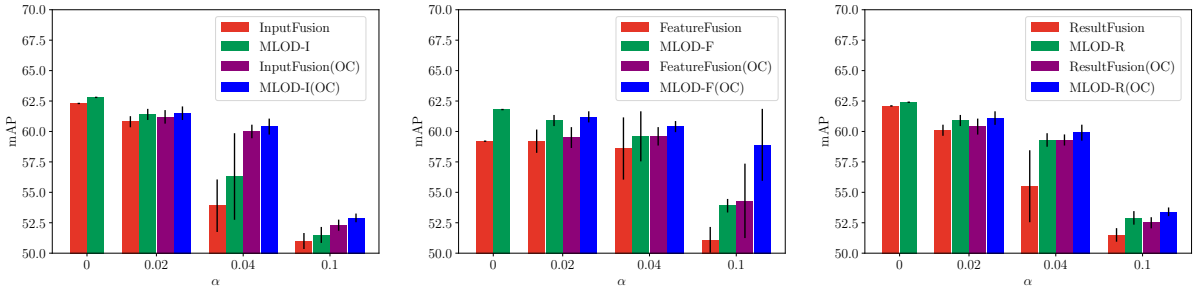


Figure 6.5: Mean and variance of accuracy (AP_{3D} IoU $\geqslant 0.7$) under different level of the extrinsic perturbation.

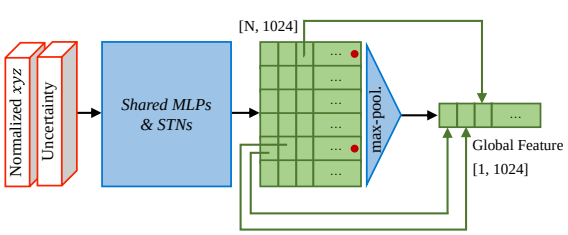
perform better or comparable to their one-stage counterparts. Online calibration baselines (*Input Fusion (OC)*, *Feature Fusion (OC)*, *Result Fusion (OC)*) demonstrate better results than those without online calibration. With the assistance of *MLOD*, their performance is further enhanced. *Feature Fusion* and its variants perform better than the others. We explain that *Feature Fusion* fuses data in a high dimensional embedding space, and each feature is related to an area of the cognitive region. Regarding *LiDAR-Top*, *MLOD*'s variants also perform better or comparable results. A qualitative result is illustrated in Fig. 6.7, and more results are shown in the supplementary materials.

We also conduct a sensitivity analysis of extrinsic uncertainty with a fixed α as input in the supplementary material. The same conclusion still holds. To further explore the reason behind the performance of *MLOD*, we investigate the characteristics of active points. These points are activated by PointNet before the max-pooling layer (Fig. 6.6a). We compute the difference between the input points and active points on the proportion of uncertain points[5] for each sample. Fig. 6.6b plots a histogram of the proportional difference overall input samples. It shows that the network deactivates highly uncertain points, and explain why *MLOD* is robust.

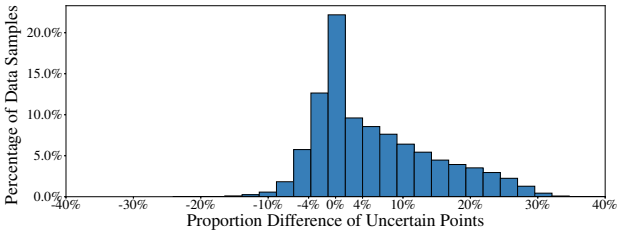
6.7 Conclusion

In this thesis, we analyze the extrinsic perturbation effect on multi-LiDAR-based 3D object detection. We propose a two-stage network to both fuse the data from multiple LiDARs and handle extrinsic perturbation after data fusion. We conduct extensive

[5]The proportion of the uncertain points is defined as $\frac{|\mathcal{P}_u|}{|\mathcal{P}|}$, where $\mathcal{P}_u \in \mathcal{P}$ is the set of uncertain points with the uncertain quantity > 0.05.



(a) The details of PointNet [25].



(b) Histogram of proportional differences between input uncertain points and active uncertain points.

Figure 6.6: (a) Active points (Red dotted) form the global features after applying the max-pooling symmetric function. (b) The proportional differences larger than 0.04 (4%) occupy about 44.0%, and those less than -0.04 (-4%) occupy about 8.5%. Less uncertain points exist in the active points.

experiments on a real-world dataset and discuss the results in different levels of extrinsic perturbation. In the perturbation-free situation, we show the multi-LiDAR fusion approaches obtain better accuracy than single-LiDAR detectors. Under extrinsic perturbation, *MLOD* performs great robustness with the assistance of the uncertainty prior. A future direction concerns the combination with sensors in various modalities, e.g., LiDAR-camera-radar setups, for developing a more accurate object detector.

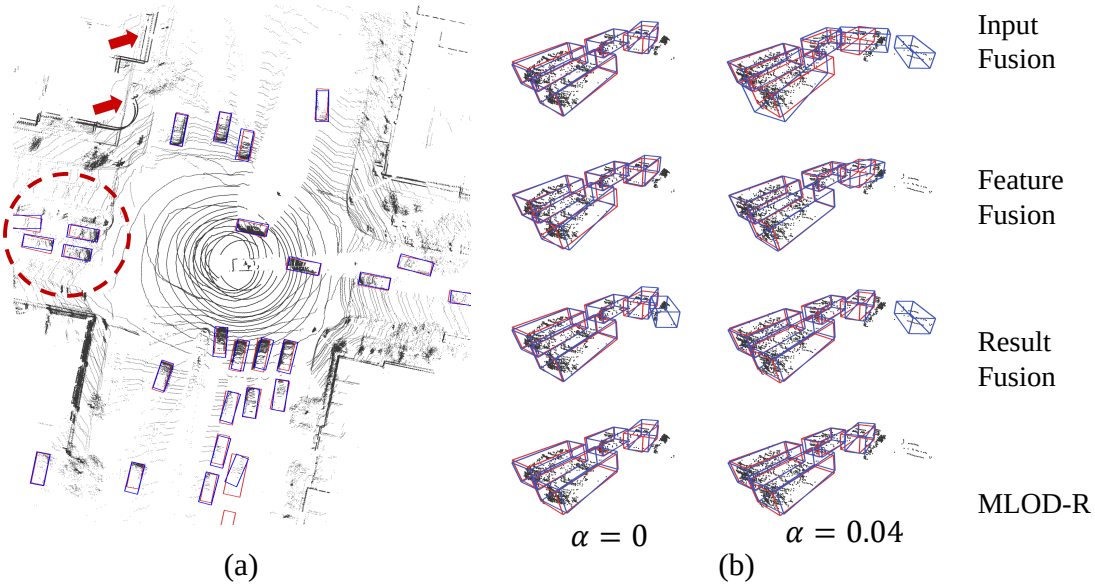


Figure 6.7: Visualization of the fusion schemes results (For comparison, here we sample the extrinsic perturbation for all the ($\alpha = 0.04$) cases at the σ position according to $\mathbf{\Theta}$. (a) Bird's-eye view of *MLOD*'s detection results when $\alpha = 0.04$. The extrinsic perturbation is observed from where the red arrows indicate. (b) A close-up view of results estimated by different fusion schemes within the red circle. Left to right: $\alpha = 0$, $\alpha = 0.04$. *Input Fusion*, *Feature Fusion* and *Result Fusion* suffer from false positives or inaccurate boxes caused by extrinsic perturbation and inaccurate box location. Compared with its counterpart (*Result Fusion*), *MLOD-R* eliminates the false positives and refines the 3D boxes.

Part IV

Conclusion and Future Works

CHAPTER 7

CONCLUSION AND FUTURE WORKS

7.1 Conclusion

This thesis presents contributions to the SOTA LiDAR perception algorithms of autonomous robots. In Chap. 1, I briefly introduce the background, review the main features and algorithms of sensors for robotics, and formulate the LiDAR perception. I am motivated to develop the multi-LiDAR perception system by addressing key challenges step by step ranging from extrinsic calibration, SLAM, and 3D object detection. In Chap. 2, I introduce an offline, dual-LiDAR extrinsic calibration method using three linearly independent planar surfaces. This method is good at accuracy while requiring a well-constrained calibration target. Then in Chap. 3, an offline hybrid calibration method that takes advantage of motion features for initialization and appearance cues for refinement, is presented. In Chap. 4, I follow the initialization-refinement paradigm to propose an online procedure to achieve flexible extrinsic calibration by combining with SLAM techniques. Finally, a complete online system called *M-LOAM* for multi-LiDAR extrinsic calibration, odometry, and mapping is presented. Then in Chap. 5, I accelerate the optimization process by proposing a greedy-based feature selection algorithm, leading to both lower latency and better performance of M-LOAM. Finally, I integrate the previous research with SOTA learning approaches to propose a multi-LiDAR object detector for autonomous driving. I have released the implementation of these methods B.

Although all methods shown in this thesis are developed based on LiDAR-only systems, they are general and extensible to fusion other sensor modalities. Regarding the impact, my works on calibration and SLAM have been partially applied in several industrial products [116]. The hybrid method in Chap. 3 is integrated into the MATLAB LiDAR toolbox[1]. My works also inspire the novel design of multi-LiDAR systems for various applications [27, 66, 117, 131, 132].

[1] `https://www.mathworks.com/help/lidar/ug/multi-lidar-calibration-workflow.html`

7.2 Limitations

This thesis presents solutions to several essential perception problems using the multi-LiDAR system. However, several limitations still exist.

1. The system utilizes point cloud registration as the key algorithm to estimate states. As a typical non-convex problem, registration requires a good initial guess. But LiDARs produce a low-frequency data stream, making this problem sometimes challenging. An example is that when a robot moves and turns at a high frequency, the system barely tracks its poses. Typical cases can be seen in the state estimation on drones, AR/VR devices, and legged robots.

2. The system extracts simple edge and planar features from environments. However, these features present drawbacks in real tests. They only provide constraints in their perpendicular directions. In a long tunnel, where all planes are mostly parallel, the system may fail. Another example is that such features do not have enough recognition power to enable robust matching across frames with large viewpoint changes. As compared with surfel-based or visual features, they are less useful for tasks such as place recognition and relocalization.

3. LiDARs cannot provide texture information of surroundings. The built global map is purely geometric and lacks rich semantic information, which may limit the navigation performance of agents.

7.3 Future Works

To address these limitations, more cutting-edge topics beyond the current LiDAR-only perception system are worth future research. In what follows, I briefly state the plan for my future research.

7.3.1 Sensor Fusion for Robust Perception

The LiDAR-only sensor presented in this thesis provides insufficient information if the targeted problems become more complex. For example, visual texture information is needed for the scene reconstruction task. This motivates me to seek the fusion of

LiDARs with sensors in other modalities to achieve robust perception. Besides IMUs and cameras, novel sensors such as event cameras can be also considered. I will exploit such a multi-sensor system to address fundamental perception problems such as image denoising, feature extraction, depth estimation, as well as dense mapping.

7.3.2 Exploiting Semantic Constraints for SLAM

Most methodologies reported in this thesis are designed on the traditional geometric framework while neglecting the high-level semantics from the scene. As the development of learning approaches in segmentation and scene understanding, the algorithm design for SLAM that utilizes geometric primitives should be rethought. Based on the multi-sensor system, I plan to develop a complete SLAM system by incorporating both geometric and semantic constraints. I will answer an important question in this domain: *how to make use of semantic constraints to improve SLAM in data association and map building?*

7.3.3 Semantics for High-Level Navigation Tasks

The semantic information extracted from the perception modules can also evolve the design of high-level navigation methodologies, like path planning. For instance, the high-definition map consisting of traffic scene information is useful for localization and decision making. For legged robots, the elevation map can assist the subsequent motion control. Some robots may require semantic and dynamic understanding, including motion forecasting of people, to safely move around without colliding. These research results will facilitate several robotic applications ranging from autonomous logistics transportation, home service, and photography.

REFERENCES

[1] Sameer Agarwal and Keir Mierle. Others,ceres solver,, 2015.

[2] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[3] Siddharth Agarwal, Ankit Vora, Gaurav Pandey, Wayne Williams, Helen Kourous, and James McBride. Ford multi-av seasonal dataset. arXiv preprint arXiv:2003.07969, 2020.

[4] Alex M Andrew. Multiple view geometry in computer vision. Kybernetes, 2001.

[5] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. IEEE Transactions on pattern analysis and machine intelligence, (5):698–700, 1987.

[6] Timothy D Barfoot. State estimation for robotics. Cambridge University Press, 2017.

[7] Timothy D Barfoot and Paul T Furgale. Associating uncertainty with three-dimensional poses for use in estimation problems. IEEE Transactions on Robotics, 30(3):679–693, 2014.

[8] Dan Barnes, Matthew Gadd, Paul Murcutt, Paul Newman, and Ingmar Posner. The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset. arXiv preprint arXiv:1909.01300, 2019.

[9] Dan Barnes, Matthew Gadd, Paul Murcutt, Paul Newman, and Ingmar Posner. The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 6433–6438. IEEE, 2020.

[10] Jens Behley and Cyrill Stachniss. Efficient surfel-based slam using 3d laser range data in urban environments. In Robotics: Science and Systems, 2018.

[11] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslamlearning a compact, optimisable representation for dense visual slam. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2560–2568, 2018.

[12] Bruno Bodin, Harry Wagstaff, Sajad Saecdi, Luigi Nardi, Emanuele Vespa, John Mawer, Andy Nisbet, Mikel Luján, Steve Furber, Andrew J Davison, et al. Slambench2: Multi-objective head-to-head benchmarking for visual slam. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1–8. IEEE, 2018.

[13] Igor Bogoslavskyi and Cyrill Stachniss. Fast range image-based segmentation of sparse 3d laser scans for online operation. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 163–169. IEEE, 2016.

[14] Michael Bosse, Gabriel Agamennoni, Igor Gilitschenski, et al. Robust estimation and applications in robotics. Now Publishers, 2016.

[15] Michael Bosse, Robert Zlot, and Paul Flick. Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. IEEE Transactions on Robotics, 28(5):1104–1119, 2012.

[16] Michael Broome, Matthew Gadd, Daniele De Martini, and Paul Newman. On the road: Route proposal from radar self-supervised by fuzzy lidar traversability. AI, 1(4):558–585, 2020.

[17] Martin Brossard, Silvére Bonnabel, and Axel Barrau. A new approach to 3d icp covariance estimation. IEEE Robotics and Automation Letters, 5(2):744–751, 2020.

[18] Keenan Burnett, Angela P Schoellig, and Timothy D Barfoot. Do we need to compensate for motion distortion and doppler effects in spinning radar navigation? IEEE Robotics and Automation Letters, 6(2):771–778, 2021.

[19] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on robotics, 32(6):1309–1332, 2016.

[20] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. IEEE Transactions on Robotics, 2021.

[21] Luca Carlone and Sertac Karaman. Attention and anticipation in fast visual-inertial navigation. IEEE Transactions on Robotics, 35(1):1–20, 2018.

[22] Sarah H Cen and Paul Newman. Precise ego-motion estimation with millimeter-wave radar under diverse and challenging conditions. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 6045–6052. IEEE, 2018.

[23] Sarah H Cen and Paul Newman. Radar-only ego-motion estimation in difficult settings via graph matching. In 2019 International Conference on Robotics and Automation (ICRA), pages 298–304. IEEE, 2019.

[24] Andrea Censi. An accurate closed-form estimate of icp's covariance. In Proceedings 2007 IEEE international conference on robotics and automation, pages 3167–3172. IEEE, 2007.

[25] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 77–85, 2017.

[26] Homer H Chen. A screw motion approach to uniqueness analysis of head-eye geometry. In Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 145–151. IEEE, 1991.

[27] Pengxin Chen, Wenzhong Shi, Sheng Bao, Muyang Wang, Wenzheng Fan, and Haodong Xiang. Low-drift odometry, mapping and ground segmentation using a backpack lidar system. IEEE Robotics and Automation Letters, 6(4):7285–7292, 2021.

[28] Steven W Chen, Guilherme V Nardari, Elijah S Lee, Chao Qu, Xu Liu, Roseli Ap Francelin Romero, and Vijay Kumar. Sloam: Semantic lidar odometry and mapping for forest inventory. IEEE Robotics and Automation Letters, 5(2):612–619, 2020.

[29] Xiaozhi Chen et al. Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE CVPR, pages 1907–1915, 2017.

[30] Xieyuanli Chen, Thomas Läbe, Andres Milioto, Timo Röhling, Olga Vysotska, Alexandre Haag, Jens Behley, Cyrill Stachniss, and FKIE Fraunhofer. Overlapnet: Loop closing for lidar-based slam. In Proc. of Robotics: Science and Systems (RSS), 2020.

[31] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguère, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4530–4537. IEEE, 2019.

[32] Dong-Geol Choi, Yunsu Bok, Jun-Sik Kim, and In So Kweon. Extrinsic calibration of 2-d lidars using two orthogonal planes. IEEE Transactions on Robotics, 32(1):83–98, 2015.

[33] Dong-Geol Choi, Yunsu Bok, Jun-Sik Kim, and In So Kweon. Extrinsic calibration of 2-d lidars using two orthogonal planes. IEEE Transactions on Robotics, 32(1):83–98, 2016.

[34] Siddharth Choudhary, Vadim Indelman, Henrik I Christensen, and Frank Dellaert. Information-based reduced landmark slam. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 4620–4627. IEEE, 2015.

[35] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. Introduction to algorithms. MIT press, 2009.

[36] Konstantinos Daniilidis. Hand-eye calibration using dual quaternions. The International Journal of Robotics Research, 18(3):286–298, 1999.

[37] Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset. In 2019 International Conference on Robotics and Automation (ICRA), pages 6713–6719. IEEE, 2019.

[38] David Droeschel, Jörg Stückler, and Sven Behnke. Local multi-resolution representation for 6d motion estimation and mapping with a continuously rotating 3d laser scanner. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 5221–5226. IEEE, 2014.

[39] Kevin Eckenhoff, Yulin Yang, Patrick Geneva, and Guoquan Huang. Tightly-coupled visual-inertial localization and 3-d rigid-body target tracking. IEEE Robotics and Automation Letters, 4(2):1541–1548, 2019.

[40] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. The farthest point strategy for progressive image sampling. IEEE Transactions on Image Processing, 6(9):1305–1315, 1997.

[41] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. IEEE transactions on pattern analysis and machine intelligence, 40(3):611–625, 2017.

[42] EUSPA. What is gnss. `https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss`, 2021.

[43] Davide Falanga, Kevin Kleber, Stefano Mintchev, Dario Floreano, and Davide Scaramuzza. The foldable drone: A morphing quadrotor that can squeeze and fly. IEEE Robotics and Automation Letters, 4(2):209–216, 2018.

[44] Rui Fan, Jianhao Jiao, Jie Pan, Huaiyang Huang, Shaojie Shen, and Ming Liu. Real-time dense stereo embedded in a uav for road inspection, 2019.

[45] Di Feng et al. Leveraging heteroscedastic aleatoric uncertainties for robust real-time lidar 3d object detection. In 2019 IEEE Intelligent Vehicles Symposium (IV), pages 1280–1287. IEEE, 2019.

[46] Alejandro Fontan, Javier Civera, and Rudolph Triebel. Information-driven direct rgb-d odometry. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4929–4937, 2020.

[47] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. IEEE Transactions on Robotics, 33(1):1–21, 2016.

[48] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. IEEE Transactions on Robotics, 33(2):249–265, 2016.

[49] David Forsyth and Jean Ponce. Computer vision: A modern approach. Prentice hall, 2011.

[50] Matthew Gadd, Daniele De Martini, and Paul Newman. Unsupervised place recognition with deep embedding learning over radar videos. arXiv preprint arXiv:2106.06703, 2021.

[51] G. Gallego, T. Delbruck, G. M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. Event-based vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1–1, 2020.

[52] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics, 28(5):1188–1197, 2012.

[53] Chao Gao and John R Spletzer. On-line calibration of multiple lidars on a mobile vehicle platform. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 279–284. IEEE, 2010.

[54] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. The International Journal of Robotics Research, 32(11):1231–1237, 2013.

[55] Andreas Geiger, Frank Moosmann, Ömer Car, and Bernhard Schuster. Automatic camera and range sensor calibration using a single shot. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 3936–3943. IEEE, 2012.

[56] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, et al. A2d2: Audi autonomous driving dataset. arXiv preprint arXiv:2004.06320, 2020.

[57] Gene H Golub and Charles F Van Loan. Matrix computations, volume 3. JHU press, 2013.

[58] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. arXiv preprint arXiv:1706.01307, 2017.

[59] W Shane Grant, Randolph C Voorhies, and Laurent Itti. Finding planes in lidar point clouds for real-time registration. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4347–4354. IEEE, 2013.

[60] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In International Conference on Machine Learning, pages 1321–1330, 2017.

[61] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. Feature extraction: foundations and applications, volume 207. Springer, 2008.

[62] L. Han, F. Gao, B. Zhou, and S. Shen. Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4423–4430, 2019.

[63] Mengwen He, Huijing Zhao, Jinshi Cui, and Hongbin Zha. Calibration method for multiple 2d lidars system. In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 3034–3041. IEEE, 2014.

[64] Mengwen He, Huijing Zhao, Franck Davoine, Jinshi Cui, and Hongbin Zha. Pairwise lidar calibration using multi-type 3d geometric features in natural scene. In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pages 1828–1835. IEEE, 2013.

[65] Garrett Hemann, Sanjiv Singh, and Michael Kaess. Long-range gps-denied aerial inertial navigation with lidar localization. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1659–1666. IEEE, 2016.

[66] Lionel Heng. Automatic targetless extrinsic calibration of multiple 3d lidars and radars. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 10669–10675. IEEE, 2020.

[67] Lionel Heng, Gim Hee Lee, and Marc Pollefeys. Self-calibration and visual slam with a multi-camera system on a micro aerial vehicle. Autonomous robots, 39(3):259–277, 2015.

[68] Lionel Heng, Bo Li, and Marc Pollefeys. Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pages 1793–1800. IEEE, 2013.

[69] Akshay Hinduja, Bing-Jui Ho, and Michael Kaess. Degeneracy-aware factors with applications to underwater slam. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1293–1299, 2019.

[70] Ziyang Hong, Yvan Petillot, and Sen Wang. Radarslam: Radar based large-scale slam in all weathers. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5164–5170. IEEE, 2020.

[71] Radu Horaud and Fadi Dornaika. Hand-eye calibration. The international journal of robotics research, 14(3):195–210, 1995.

[72] Chong Huang, Fei Gao, Jie Pan, Zhenyu Yang, Weihao Qiu, Peng Chen, Xin Yang, Shaojie Shen, and Kwang-Ting Cheng. Act: An autonomous drone cinematography system for action scenes. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 7039–7046. IEEE, 2018.

[73] Huaiyang Huang, Haoyang Ye, Yuxiang Sun, and Ming Liu. Gmmloc: Structure consistent visual localization with gaussian mixture models. IEEE Robotics and Automation Letters, 5(4):5043–5050, 2020.

[74] Kaihong Huang and Cyrill Stachniss. On geometric models and their accuracy for extrinsic sensor calibration. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1–9. IEEE, 2018.

[75] ICRA. Radar in robotics. `https://sites.google.com/view/radar-robotics/home`, 2021.

[76] Viorela Ila, Lukas Polok, Marek Solony, Pavel Smrz, and Pavel Zemcik. Fast covariance recovery in incremental nonlinear least-squares solvers. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 4636–4643. IEEE, 2015.

[77] Kaijin Ji, Huiyan Chen, Huijun Di, Jianwei Gong, Guangming Xiong, Jianyong Qi, and Tao Yi. Cpfg-slam: a robust simultaneous localization and mapping based on lidar in off-road environment. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 650–655. IEEE, 2018.

[78] Jiao Jianhao, Yun Peng, Tai Lei, and Liu Ming. Mlod: Awareness of extrinsic perturbation in multi-lidar 3d object detection for autonomous driving. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

[79] J. Jiao, Y. Yu, Q. Liao, H. Ye, R. Fan, and M. Liu. Automatic calibration of multiple 3d lidars in urban environments. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 15–20, 2019.

[80] Jianhao Jiao, Qinghai Liao, Yilong Zhu, Tianyu Liu, Yang Yu, Rui Fan, Lujia Wang, and Ming Liu. A novel dual-lidar calibration algorithm using planar surfaces. In 2019 IEEE Intelligent Vehicles Symposium (IV), pages 1499–1504. IEEE, 2019.

[81] Jianhao Jiao, Qinghai Liao, Yilong Zhu, Tianyu Liu, Yang Yu, Rui Fan, Lujia Wang, and Ming Liu. A novel dual-lidar calibration algorithm using planar surfaces. In IEEE Intelligent Vehicles Symposium (IV), 2019.

[82] Jianhao Jiao, Haoyang Ye, Yilong Zhu, and Ming Liu. Supplementary material to: Robust odometry and mapping for multi-lidar systems with online extrinsic calibration. Technical report, 2020.

[83] Jianhao Jiao, Haoyang Ye, Yilong Zhu, and Ming Liu. Robust odometry and mapping for multi-lidar systems with online extrinsic calibration. IEEE Transactions on Robotics, 2021.

[84] Jianhao Jiao, Yilong Zhu, Haoyang Ye, Huaiyang Huang, Peng Yun, Linxin Jiang, Lujia Wang, and Ming Liu. Greedy-based feature selection for efficient lidar slam. In 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021.

[85] Wolfgang Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography, 34(5):827–828, 1978.

[86] Michael Kaess and Frank Dellaert. Covariance recovery from a square root information matrix for data association. Robotics and Autonomous Systems, 57(12):1198–1210, 2009.

[87] A Kendall, V Badrinarayanan, and R Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In British Machine Vision Conference 2017, BMVC 2017, 2017.

[88] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In Advances in neural information processing systems, pages 5574–5584, 2017.

[89] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 av dataset 2019. urlhttps://level5.lyft.com/dataset/, 2019.

[90] Giseop Kim, Yeong Sang Park, Younghun Cho, Jinyong Jeong, and Ayoung Kim. Mulran: Multimodal range dataset for urban place recognition. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 6246–6253. IEEE, 2020.

[91] Youngji Kim and Ayoung Kim. On the uncertainty propagation: Why uncertainty on lie groups preserves monotonicity? In 2017 IROS, pages 3425–3432. IEEE, 2017.

[92] Nathan P Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In IROS, volume 4, pages 2149–2154. Citeseer, 2004.

[93] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1–8. IEEE, 2018.

[94] Vijay Kumar and Nathan Michael. Opportunities and challenges with autonomous micro aerial vehicles. The International Journal of Robotics Research, 31(11):1279–1291, 2012.

[95] Rainer Kümmerle, Giorgio Grisetti, and Wolfram Burgard. Simultaneous calibration, localization, and mapping. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3716–3721. IEEE, 2011.

[96] David Landry, François Pomerleau, and Philippe Giguere. Cello-3d: Estimating the covariance of icp in the real world. In 2019 International Conference on Robotics and Automation (ICRA), pages 8190–8196. IEEE, 2019.

[97] Itai Lang, Asaf Manor, and Shai Avidan. Samplenet: Differentiable point cloud sampling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7578–7588, 2020.

[98] Cedric Le Gentil, Teresa Vidal-Calleja, and Shoudong Huang. 3d lidar-imu calibration based on upsampled preintegrated measurements for motion distortion correction. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 2149–2155. IEEE, 2018.

[99] Cedric Le Gentil, Teresa Vidal-Calleja, and Shoudong Huang. In2lama: Inertial lidar localisation and mapping. In 2019 International Conference on Robotics and Automation (ICRA), pages 6388–6394. IEEE, 2019.

[100] Cedric Le Gentil, Teresa Vidal-Calleja, and Shoudong Huang. In2laama: Inertial lidar localization autocalibration and mapping. IEEE Transactions on Robotics, 2020.

[101] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. Science robotics, 5(47), 2020.

[102] Stefan Leutenegger, Paul Furgale, Vincent Rabaud, Margarita Chli, Kurt Konolige, and Roland Siegwart. Keyframe-based visual-inertial slam using nonlinear optimization. Proceedings of Robotis Science and Systems (RSS) 2013, 2013.

[103] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research, 34(3):314–334, 2015.

[104] Jesse Levinson and Sebastian Thrun. Automatic online calibration of cameras and lasers. In Robotics: Science and Systems, volume 2, 2013.

[105] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1513–1518. IEEE, 2017.

[106] Mingyang Li, Hongsheng Yu, Xing Zheng, and Anastasios I Mourikis. High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation. In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 409–416. IEEE, 2014.

[107] Qinghai Liao, Zhenyong Chen, Yang Liu, Zhe Wang, and Ming Liu. Extrinsic calibration of lidar and camera with polygon. In 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 200–205. IEEE, 2018.

[108] Qinghai Liao, Zhenyong Chen, Yang Liu, Zhe Wang, and Ming Liu. Extrinsic calibration of lidar and camera with polygon. In IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, 2019.

[109] Qinghai Liao, Ming Liu, Lei Tai, and Haoyang Ye. Extrinsic calibration of 3d range finder and camera without auxiliary object or human intervention, 2017.

[110] 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), pages 3126–3131, 2020.

[111] Jiarong Lin, Xiyuan Liu, and Fu Zhang. A decentralized framework for simultaneous calibration, localization and mapping with multiple lidars. arXiv preprint arXiv:2007.01483, 2020.

[112] Jiarong Lin and Fu Zhang. R3live: A robust, real-time, rgb-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package. arXiv preprint arXiv:2109.07982, 2021.

[113] Jiarong Lin, Chunran Zheng, Wei Xu, and Fu Zhang. R2live: A robust, real-time, lidar-inertial-visual tightly-coupled state estimator and mapping. arXiv preprint arXiv:2102.12400, 2021.

[114] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision, pages 2980–2988, 2017.

[115] Yonggen Ling and Shaojie Shen. High-precision online markerless stereo extrinsic calibration. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1771–1778. IEEE, 2016.

[116] Tianyu Liu, Qinghai Liao, Lu Gan, Fulong Ma, Jie Cheng, Xupeng Xie, Zhe Wang, Yingbing Chen, Yilong Zhu, Shuyang Zhang, et al. The role of the hercules autonomous vehicle during the covid-19 pandemic: An autonomous logistic vehicle for contactless goods transportation. 2021.

[117] Xiyuan Liu, Chongjian Yuan, and Fu Zhang. Fast and accurate extrinsic calibration for multiple lidars and cameras. arXiv preprint arXiv:2109.06550, 2021.

[118] Xiyuan Liu and Fu Zhang. Extrinsic calibration of multiple lidars of small fov in targetless environments. IEEE Robotics and Automation Letters, 6(2):2036–2043, 2021.

[119] Zheng Liu and Fu Zhang. Balm: Bundle adjustment for lidar mapping. arXiv preprint arXiv:2010.08215, 2020.

[120] David G Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91–110, 2004.

[121] Thomas Lowe, Soohwan Kim, and Mark Cox. Complementary perception for handheld slam. IEEE Robotics and Automation Letters, 3(2):1104–1111, 2018.

[122] Simon Lynen, Markus W Achtelik, Stephan Weiss, Margarita Chli, and Roland Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In 2013 IEEE/RSJ international conference on intelligent robots and systems, pages 3923–3929. IEEE, 2013.

[123] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. The International Journal of Robotics Research, 36(1):3–15, 2017.

[124] Martin Magnusson, Achim Lilienthal, and Tom Duckett. Scan registration for autonomous mining vehicles using 3d-ndt. Journal of Field Robotics, 24(10):803–827, 2007.

[125] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In 2017 IEEE International Conference on Robotics and automation (ICRA), pages 4628–4635. IEEE, 2017.

[126] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[127] Shenyu Mou, Yan Chang, Wenshuo Wang, and Ding Zhao. An optimal lidar configuration approach for self-driving cars. arXiv preprint arXiv:1805.07843, 2018.

[128] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. IEEE transactions on robotics, 31(5):1147–1163, 2015.

[129] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE transactions on robotics, 33(5):1255–1262, 2017.

[130] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In 2011 10th IEEE international symposium on mixed and augmented reality, pages 127–136. IEEE, 2011.

[131] Thien-Minh Nguyen, Shenghai Yuan, Muqing Cao, Thien Hoang Nguyen, and Lihua Xie. Viral slam: Tightly coupled camera-imu-uwb-lidar slam. arXiv preprint arXiv:2105.03296, 2021.

[132] Thien-Minh Nguyen, Shenghai Yuan, Muqing Cao, Lyu Yang, Thien Hoang Nguyen, and Lihua Xie. Miliom: Tightly coupled multi-input lidar-inertia odometry and mapping. IEEE Robotics and Automation Letters, 6(3):5573–5580, 2021.

[133] Zhanpeng Ouyang, Lan Hu, Yukan LU, Zhirui Wang Wang, Xin Peng, and Laurent Kneip. Online calibration of exterior orientations of a vehicle-mounted surround-view camera system. In 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020.

[134] Gaurav Pandey, Shashank Giri, and Jame R Mcbride. Alignment of 3d point clouds with a dominant ground plane. In Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, pages 2143–2150. IEEE, 2017.

[135] Gaurav Pandey, James R McBride, Silvio Savarese, and Ryan M Eustice. Automatic extrinsic calibration of vision and lidar by maximizing mutual information. Journal of Field Robotics, 32(5):696–722, 2015.

[136] Chanoh Park, Peyman Moghadam, Jason Williams, Soohwan Kim, Sridha Sridharan, and Clinton Fookes. Elasticity meets continuous-time: Map-centric dense 3d lidar slam. arXiv preprint arXiv:2008.02274, 2020.

[137] Yeong Sang Park, Young-Sik Shin, and Ayoung Kim. Pharao: Direct radar odometry using phase correlation. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 2617–2623. IEEE, 2020.

[138] Andreas Pfrunder, Paulo VK Borges, Adrian R Romero, Gavin Catt, and Alberto Elfes. Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3d lidar. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2601–2608. IEEE, 2017.

[139] François Pomerleau, Andreas Breitenmoser, Ming Liu, Francis Colas, and Roland Siegwart. Noise characterization of depth sensors for surface inspections. In 2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI), pages 16–21. IEEE, 2012.

[140] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing icp variants on real-world data sets. Autonomous Robots, 34(3):133–148, 2013.

[141] Charles R Qi, Wei Liu, Chenxia Wu, et al. Frustum pointnets for 3d object detection from rgb-d data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 918–927, 2018.

[142] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Advances in Neural Information Processing Systems, pages 5099–5108, 2017.

[143] Chao Qin, Haoyang Ye, Christian E Pranata, Jun Han, and Ming Liu. Lins: A lidar-inerital state estimator for robust and fast navigation. 2020.

[144] Tong Qin, Tongqing Chen, Yilun Chen, and Qing Su. Avp-slam: Semantic visual mapping and localization for autonomous vehicles in the parking lot. arXiv preprint arXiv:2007.01813, 2020.

[145] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. IEEE Transactions on Robotics, 34(4):1004–1020, 2018.

[146] Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors. arXiv preprint arXiv:1901.03638, 2019.

[147] Tong Qin and Shaojie Shen. Online temporal calibration for monocular visual-inertial systems. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3662–3669. IEEE, 2018.

[148] Kejie Qiu, Tong Qin, Wenliang Gao, and Shaojie Shen. Tracking 3-d motion of dynamic objects using monocular visual-inertial sensing. IEEE Transactions on Robotics, 35(4):799–816, 2019.

[149] Kejie Qiu, Tong Qin, Jie Pan, Siqi Liu, and Shaojie Shen. Real-time temporal and rotational calibration of heterogeneous sensors using motion correlation analysis. IEEE Transactions on Robotics, 2020.

[150] Jan Quenzel, Nils Papenberg, and Sven Behnke. Robust extrinsic calibration of multiple stationary laser range finders. In Automation Science and Engineering (CASE), 2016 IEEE International Conference on, pages 1332–1339. IEEE, 2016.

[151] Jan Razlaw, David Droeschel, Dirk Holz, and Sven Behnke. Evaluation of registration methods for sparse 3d laser scans. In 2015 European Conference on Mobile Robots (ECMR), pages 1–7. IEEE, 2015.

[152] Kuka Robotics. Kuka navigation solution. `Available:http://www.kuka-robotics.com/res/robotics/Products/PDF/EN/KUKANavigation_Solution_EN.pdf`, 2016.

[153] Zheng Rong and Nathan Michael. Detection and prediction of near-term state estimation degradation via online nonlinear observability analysis. In 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pages 28–33. IEEE, 2016.

[154] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 1689–1696. IEEE, 2020.

[155] Jörg Röwekämper, Michael Ruhnke, Bastian Steder, Wolfram Burgard, and Gian Diego Tipaldi. Automatic extrinsic calibration of multiple laser range sensors with little overlap. In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pages 2072–2077. IEEE, 2015.

[156] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In 2011 International Conference on Computer Vision, pages 2564–2571, 2011.

[157] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In Robotics and automation (ICRA), 2011 IEEE International Conference on, pages 1–4. IEEE, 2011.

[158] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1352–1359, 2013.

[159] Davide Scaramuzza. Tutorial on event-bbased cameras. `http://rpg.ifi.uzh.ch/docs/scaramuzza/Tutorial_on_Event_Cameras_Scaramuzza.pdf`, 2020.

[160] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. Annual Review of Control, Robotics, and Autonomous Systems, 2018.

[161] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In Robotics: science and systems, volume 2, page 435, 2009.

[162] Jacopo Serafin and Giorgio Grisetti. Nicp: Dense normal based point cloud registration. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 742–749. IEEE, 2015.

[163] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4758–4765. IEEE, 2018.

[164] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5135–5142. IEEE, 2020.

[165] Tixiao Shan, Brendan Englot, Carlo Ratti, and Daniela Rus. Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. arXiv preprint arXiv:2104.10831, 2021.

[166] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–779, 2019.

[167] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–779, 2019.

[168] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction to autonomous mobile robots. MIT press, 2011.

[169] Joan Sola. Quaternion kinematics for the error-state kalman filter. arXiv preprint arXiv:1711.02508, 2017.

[170] Haoran Song, Wenchao Ding, Yuxuan Chen, Shaojie Shen, Michael Yu Wang, and Qifeng Chen. Pip: Planning-informed trajectory prediction for autonomous driving. In European Conference on Computer Vision, pages 598–614. Springer, 2020.

[171] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In 2012 IEEE/RSJ

international conference on intelligent robots and systems, pages 573–580. IEEE, 2012.

[172] Tyler H Summers, Fabrizio L Cortesi, and John Lygeros. On submodularity and controllability in complex dynamical networks. IEEE Transactions on Control of Network Systems, 3(1):91–101, 2015.

[173] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2446–2454, 2020.

[174] Richard Szeliski. Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

[175] Tim Y Tang, Daniele De Martini, and Paul Newman. Get to the point: Learning lidar place recognition and metric localisation using overhead imagery. 2021.

[176] Zachary Taylor and Juan Niet. Motion-based calibration of multimodal sensor extrinsics and timing offset estimation. IEEE Transactions on Robotics, 32(5):1215–1229, 2016.

[177] Zachary Taylor and Juan Nieto. Motion-based calibration of multimodal sensor arrays. In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pages 4843–4850. IEEE, 2015.

[178] Zachary Taylor, Juan Nieto, and David Johnson. Automatic calibration of multimodal sensor systems using a gradient orientation measure. In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pages 1293–1300. IEEE, 2013.

[179] Alex Teichman, Stephen Miller, and Sebastian Thrun. Unsupervised intrinsic calibration of depth sensors via slam. In Robotics: Science and Systems, volume 248, page 3, 2013.

[180] Sebastian Thrun. Probabilistic robotics. Communications of the ACM, 45(3):52–57, 2002.

[181] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. Journal of field Robotics, 23(9):661–692, 2006.

[182] Martin Velas, Michal Spanel, and Adam Herout. Collar line segments for fast odometry estimation from velodyne point clouds. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 4486–4495. IEEE, 2016.

[183] Guowei Wan, Xiaolong Yang, Renlan Cai, Hao Li, Yao Zhou, Hao Wang, and Shiyu Song. Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 4670–4677. IEEE, 2018.

[184] Yizhou Wang, Zhongyu Jiang, Xiangyu Gao, Jenq-Neng Hwang, Guanbin Xing, and Hui Liu. Rodnet: Radar object detection using cross-modal supervision. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 504–513, 2021.

[185] GC Weiffenbach. Tropospheric and ionospheric propagation effects on satellite radio-doppler geodesy. In Electromagnetic Distance Measurement, pages 339–352. University of Toronto Press, 2019.

[186] Jeremy N Wong, David J Yoon, Angela P Schoellig, and Timothy D Barfoot. A data-driven motion prior for continuous-time trajectory estimation on se (3). IEEE Robotics and Automation Letters, 5(2):1429–1436, 2020.

[187] Y. Wu, Y. Zhang, D. Zhu, Y. Feng, S. Coleman, and D. Kerr. Eao-slam: Monocular semi-dense object slam based on ensemble data association. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4966–4973, 2020.

[188] Guoyang Xie, Tao Xu, Carsten Isert, Michael Aeberhard, Shaohua Li, and Ming Liu. Online active calibration for a multi-lrf system. In 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pages 806–811. IEEE, 2015.

[189] Yuanfan Xie, Rui Shao, Popo Guli, Bo Li, and Liang Wang. Infrastructure based calibration of a multi-camera and multi-lidar system using apriltags. In 2018 IEEE

Intelligent Vehicles Symposium (IV), pages 605–610. IEEE, 2018.

[190] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 244–253, 2018.

[191] Bohuan Xue, Jianhao Jiao, Yilong Zhu, Linwei Zhen, Dong Han, Ming Liu, and Rui Fan. Automatic calibration of dual-lidars using two poles stickered with retro-reflective tape. In 2019 IEEE International Conference on Imaging Systems and Techniques (IST), pages 1–6. IEEE, 2019.

[192] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. Sensors, 18(10):3337, 2018.

[193] Heng Yang and Luca Carlone. A polynomial-time solution for robust registration with extreme outlier rates. arXiv preprint arXiv:1903.08588, 2019.

[194] Shichao Yang and Sebastian Scherer. Cubeslam: Monocular 3-d object slam. IEEE Transactions on Robotics, 35(4):925–938, 2019.

[195] Zhenfei Yang, Tianbo Liu, and Shaojie Shen. Self-calibrating multi-camera visual-inertial fusion for autonomous mavs. In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pages 4984–4991. IEEE, 2016.

[196] Zhenfei Yang and Shaojie Shen. Monocular visual-inertial fusion with online initialization and camera-imu calibration. In 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pages 1–8. IEEE, 2015.

[197] Haoyang Ye, Yuying Chen, and Ming Liu. Tightly coupled 3d lidar inertial odometry and mapping. In 2019 IEEE International Conference on Robotics and Automation (ICRA), 2019.

[198] Haoyang Ye, Huaiyang Huang, and Ming Liu. Monocular direct sparse localization in a prior 3d surfel map. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 8892–8898. IEEE, 2020.

[199] Huan Yin, Yue Wang, and Rong Xiong. Improved radar localization on lidar maps using shared embedding. arXiv preprint arXiv:2106.10000, 2021.

[200] Yang Yu, Wenliang Gao, Chengju Liu, Shaojie Shen, and Ming Liu. A gps-aided omnidirectional visual-inertial state estimator in ubiquitous environments. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7750–7755. IEEE, 2019.

[201] Peng Yun, Lei Tai, Yuan Wang, Chengju Liu, and Ming Liu. Focal loss in 3d object detection. IEEE Robotics and Automation Letters, 4(2):1263–1270, 2019.

[202] Peng Yun, Lei Tai, Yuan Wang, Chengju Liu, and Ming Liu. Focal loss in 3d object detection. IEEE Robotics and Automation Letters, 4(2):1263–1270, April 2019.

[203] Ji Zhang, Chen Hu, Rushat Gupta Chadha, and Sanjiv Singh. Maximum likelihood path planning for fast aerial maneuvers and collision avoidance. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2805–2812. IEEE, 2019.

[204] Ji Zhang, Michael Kaess, and Sanjiv Singh. On degeneracy of optimization-based state estimation problems. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 809–816. IEEE, 2016.

[205] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In Robotics: Science and Systems, volume 2, page 9, 2014.

[206] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 2174–2181. IEEE, 2015.

[207] Ji Zhang and Sanjiv Singh. Enabling aggressive motion estimation at low-drift and accurate mapping in real-time. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 5051–5058. IEEE, 2017.

[208] Jun Zhang, Mina Henein, Robert Mahony, and Viorela Ila. Vdo-slam: A visual dynamic object-aware slam system. arXiv preprint arXiv:2005.11052, 2020.

[209] Zhengyou Zhang. A flexible new technique for camera calibration. IEEE Transactions on pattern analysis and machine intelligence, 22(11):1330–1334, 2000.

[210] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7244–7251. IEEE, 2018.

[211] Shibo Zhao, Zheng Fang, Haolai Li, and Sebastian Scherer. A robust laser-inertial odometry and mapping method for large-scale highway environments. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019.

[212] Y. Zhao and P. A. Vela. Good feature matching: Toward accurate, robust vo/vslam with low latency. IEEE Transactions on Robotics, 36(3):657–675, 2020.

[213] Yipu Zhao, Justin S Smith, and Patricio A Vela. Good graph to optimize: Cost-effective, budget-aware bundle adjustment in visual slam. arXiv preprint arXiv:2008.10123, 2020.

[214] Weikun Zhen and Sebastian Scherer. Estimating the localizability in tunnel-like environments using lidar and uwb. In 2019 International Conference on Robotics and Automation (ICRA), pages 4903–4908. IEEE, 2019.

[215] Linwei Zheng, Yilong Zhu, Bohuan Xue, Ming Liu, and Rui Fan. Low-cost gps-aided lidar state estimation and map building. In 2019 IEEE International Conference on Imaging Systems and Techniques (IST), pages 1–6. IEEE, 2019.

[216] Lipu Zhou, Zimo Li, and Michael Kaess. Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5562–5569. IEEE, 2018.

[217] Yin Zhou and Oncel 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, pages 4490–4499, 2018.

[218] Zhibo Zhou, Ming Yang, Chunxiang Wang, and Bing Wang. Roi-cloud: A key region extraction method for lidar odometry and localization. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 3312–3318. IEEE, 2020.

[219] Yilong Zhu, Bohuan Xue, Linwei Zheng, Huaiyang Huang, Ming Liu, and Rui Fan. Real-time, environmentally-robust 3d lidar localization. In 2019 IEEE International Conference on Imaging Systems and Techniques (IST), pages 1–6. IEEE, 2019.

[220] Xingxing Zuo, Patrick Geneva, Yulin Yang, Woosik Lee, Yong Liu, and Guoquan Huang. Lic-fusion: Lidar-inertial-camera odometry. In 2013 IEEE/RSJ international conference on intelligent robots and systems. IEEE, 2019.

APPENDIX A

LIST OF PUBLICATIONS

Journal Papers

1. Bowen Yang, **Jianhao Jiao**, Lujia Wang, Ming Liu, Game-Theoretic Local Planner for Interactive Crowd Navigation, submitted to *IEEE Robotics and Automation Letters (RAL)*, 2021.

2. Yang Yu, Peng Yun, **Jianhao Jiao**, Bohuan Xue, Ming Liu, Accurate and Robust Visual Localization System in Large-scale Dynamic Environments, submitted to *IEEE/ASME Transactions on Mechatronics*, 2021.

3. Jin Wu, Yu Zheng, Zhi Gao, Yi Jiang, Xiangcheng Hu, Yilong Zhu, **Jianhao Jiao**, Ming Liu, Quadratic Pose Estimation Problems: Unified Solutions, Solvability/Observability Analysis and Uncertainty Description in A Globally Optimal Framework, conditionally accpeted to *IEEE Transaction on Robotics (TRO)*, 2021.

4. Huaiyang Huang, Yuxiang Sun, Wu Jin, **Jianhao Jiao**, Xiangcheng Hu, Linwei Zheng, Lujia Wang, Ming Liu, On Bundle Adjustment for Multiview Point Cloud Registration, *IEEE Robotics and Automation Letters (RAL)*, 2021.

5. **Jianhao Jiao**, Haoyang Ye, Yilong Zhu, Ming Liu, Robust Odometry and Mapping for Multi-LiDAR Systems with Online Extrinsic Calibration, *IEEE Transaction on Robotics (TRO)*, 2021.

6. Tianyu Liu, Qinghai Liao, etc, **Jianhao Jiao**, Ming Liu, Hercules: An Autonomous Logistic Vehicle for Contact-less Goods Transportation During the COVID-19 Outbreak, *IEEE Robotics and Magazine (RAM)*, 2020.

Conference Papers

1. Sukai Wang, **Jianhao Jiao**, Peide Cai, Ming Liu, R-PCC: A Baseline for Range Image-based Point Cloud Compression, submitted to *IEEE International Confer-*

ence on Robotics and Automation (ICRA), 2022.

2. **Jianhao Jiao**, Yilong Zhu, Haoyang Ye, etc, Ming Liu, Greedy-Based Feature Selection for Efficient LiDAR SLAM, *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

3. **Jianhao Jiao***, Peng Yun*, Lei Tai, Ming Liu, MLOD: Awareness of Extrinsic Perturbation in Multi-LiDAR 3D Object Detection for Autonomous Driving, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

4. M Usman Maqbool Bhutta, Shoaib Aslam, Peng Yun, **Jianhao Jiao**, Ming Liu, Smart-Inspect: Micro Scale Localization and Classification of Smart Phone Glass Defects for Industrial Automation, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

5. Yilong Zhu, Dong Han, **Jianhao Jiao**, etc, Ming Liu, Rui Fan, Road Curb Detection Using A Novel Tensor Voting Algorithm, *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019, **Best Paper Finalist**.

6. Bohuan Xue, **Jianhao Jiao**, Yilong Zhu, etc, Ming Liu, Automatic Calibration of Dual-LiDARs Using Two Poles Stickered with Retro-Reflective Tape, *IEEE International Conference on Imaging Systems and Techniques (IST)*, 2019.

7. **Jianhao Jiao**, Yang Yu, Qinghai Liao, Haoyang Ye, Ming Liu, Automatic Calibration of Multiple 3D LiDARs in Urban Environments, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

8. **Jianhao Jiao***, Qinghai Liao*, Ming Liu, et al, A Novel Dual-Lidar Calibration Algorithm Using Planar Surfaces, *IEEE Intelligent Vehicles Symposium (IV)*, 2019.

9. Rui Fan, Mohammud Bocus, **Jianhao Jiao**, Ming Liu, et al, Road Crack Detection Using Deep Convolutional Neural Network and Adaptive Thresholding, *IEEE Intelligent Vehicles Symposium (IV)*, 2019.

10. **Jianhao Jiao**, Rui Fan, Han Ma, Ming Liu, Using DP Towards A Shortest Path Problem-Related Application, *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Workshop Papers

1. **Jianhao Jiao**, Huaiyang Huang, Liang Li, Zhijian He, Yilong Zhu, Ming Liu, Comparing Representations in Tracking for Event Camera-based SLAM, *IEEE/CVF Conference on Computer Vision and Pattern Recognition, Event-based Vision Workshop* (*CVPRW*), 2021.

2. Rui Fan, **Jianhao Jiao**, Jie Pan, Huaiyang Huang, Shaojie Shen, Ming Liu, Real-Time Dense Stereo Embedded in An UAV for Road Inspection, *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), UAV Vision Workshop*, 2019.

APPENDIX B

LIST OF OPEN-SOURCE PACKAGES

1. PS-Calib: Dual-LiDAR extrinsic calibration using three planar surfaces

   `https://github.com/ram-lab/lidar_appearance_calibration`

2. Auto-Calib: Hybrid method for multi-LiDAR extrinsic calibration

   `https://github.com/ram-lab/MLC`

3. M-LOAM: A robust and online system for multi-LiDAR extrinsic calibration, odometry, and mapping

   `https://ram-lab.com/file/site/m-loam`

4. MLOD: A robust multi-LiDAR 3D object detector with the awareness of extrinsic perturbation

   `http://143.89.78.112:5000/sharing/90BpyDIuq`