Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-2021-5139805

M.Sc. Thesis

Targetless Camera-LiDAR Calibration for Autonomous Systems

Bichi Zhang B.Sc.

Abstract

In recent decades, the field of autonomous driving has witnessed rapid development, benefiting from the development of artificial intelligence-related technologies such as machine learning. Autonomous perception in driving is a key challenge, in which multi-sensor fusion is a common feature. Due to the high resolution and rich information, the camera is one of the core perceptual sensor in autonomous systems. However, the camera provides no knowledge on distance (or depth), which is insufficient for the requirements of autonomous driving. On the other hand, LiDAR provides accurate distance measurements, however the information is sparse. The complementary characteristics of cameras and LiDAR have been exploited over the past decade for autonomous navigation. In order to be able to fuse the camera and LiDAR sensor system jointly, an efficient and accurate calibration process between sensors is essential. Conventional methods for calibrating the camera and LIDAR rely on deploying artificial objects, e.g., checkerboard, on the field. Given the impracticality of such solutions, targetless calibration solutions have been proposed over the past years, which require no human intervention and are readily applicable for various autonomous systems, e.g., automotive, drones, rovers, and robots.

In this thesis, we review and analyze several classic targetless calibration schemes. Based on some of their shortcomings, a new multi-feature workflow called MulFEA (Multi-Feature Edge Alignment) is proposed. MulFEA uses the cylindrical projection method to transform the 3D-2D calibration problem into a 2D-2D calibration problem and exploits a variety of LiDAR feature information to supplement the scarce LiDAR point cloud boundaries to achieve higher features similarity compared to camera images. In addition, a feature matching function with a precision factor is designed to improve the smoothness of the objective function solution space and reduce local optima. Our results are validated using the open-source KITTI dataset, and we compare our results with several existing targetless calibration methods. In many different types of roadway environments, our algorithm provides more reliable results regarding the shape of the objective function in the 6-DOF space, which is more conducive for the optimization algorithms to solve. In the end, we also analyze the shortcomings of our proposed solutions and put forward a prospect for future research in the field of joint camera-Lidar calibration algorithms.

Targetless Camera-LiDAR Calibration for Autonomous Systems

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Bichi Zhang B.Sc.
born in Hunan, China

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Targetless Camera-LiDAR Calibration for Autonomous Systems"** by **Bichi Zhang B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: October 12, 2021

Chairman: _____

Dr.ir. Richard Hendriks

Advisor: _____

Dr.Raj Thilak Rajan

Committee Members: _____

Dr. Stefano Speretta

_____

Abstract

In recent decades, the field of autonomous driving has witnessed rapid development, benefiting from the development of artificial intelligence-related technologies such as machine learning. Autonomous perception in driving is a key challenge, in which multi-sensor fusion is a common feature. Due to the high resolution and rich information, the camera is one of the core perceptual sensor in autonomous systems. However, the camera provides no knowledge on distance (or depth), which is insufficient for the requirements of autonomous driving. On the other hand, LiDAR provides accurate distance measurements, however the information is sparse. The complementary characteristics of cameras and LiDAR have been exploited over the past decade for autonomous navigation. In order to be able to fuse the camera and LiDAR sensor system jointly, an efficient and accurate calibration process between sensors is essential. Conventional methods for calibrating the camera and LIDAR rely on deploying artificial objects, e.g., checkerboard, on the field. Given the impracticality of such solutions, targetless calibration solutions have been proposed over the past years, which require no human intervention and are readily applicable for various autonomous systems, e.g., automotive, drones, rovers, and robots.

In this thesis, we review and analyze several classic targetless calibration schemes. Based on some of their shortcomings, a new multi-feature workflow called MulFEA (Multi-Feature Edge Alignment) is proposed. MulFEA uses the cylindrical projection method to transform the 3D-2D calibration problem into a 2D-2D calibration problem and exploits a variety of LiDAR feature information to supplement the scarce LiDAR point cloud boundaries to achieve higher features similarity compared to camera images. In addition, a feature matching function with a precision factor is designed to improve the smoothness of the objective function solution space and reduce local optima. Our results are validated using the open-source KITTI dataset, and we compare our results with several existing targetless calibration methods. In many different types of roadway environments, our algorithm provides more reliable results regarding the shape of the objective function in the 6-DOF space, which is more conducive for the optimization algorithms to solve. In the end, we also analyze the shortcomings of our proposed solutions and put forward a prospect for future research in the field of joint camera-Lidar calibration algorithms.

Acknowledgments

The two years of studying in Delft have passed. Looking back on the past two years, we have all experienced too much—a deadly virus, exhaustion from working at home, uncertainty in life, academic pressure, etc. However, I am thrilled that I can withstand these tests and finally finish my graduation thesis. During my journey, many people around me helped me and accompanied me constantly. It can be said that without their support, there would be no results I have achieved today.

First of all, I want to thank my daily supervisor, Dr. Raj Rajan. Starting from the extra project, he has been guiding me for more than a year. I started to explore the related knowledge of fusion sensing from scratch. When I had no idea on where to start, Raj recommends several novel papers to me. After I focus on extrinsic calibration, I reported the results to him every week. In many cases, the results were not satisfied, and my work was often stuck. And he gave me some valuable practical suggestions so that I can explore the problem step by step. Sometimes I just couldn't complete tasks on time, and he also gave a lot of support and understanding. In recent months, he has repeatedly helped me review my reports, and also gave me the opportunity to revise it repeatedly and strive for perfection. Many times the ideas he brought to me about scientific research and the methods of writing papers guided my scientific research exploration from a relatively high angle. I learned from him not only how to solve this specific problem, but also the commonly used scientific research methodologies. In addition, during our weekly meetings, we often exchange some interesting things in life, which are especially precious during the epidemic. It can be said that Raj is not only my mentor, but also my friend.

Likewise, I am very grateful to Dr. Richard Hendriks. Although we rarely met, he helped me a lot in the administrative process. I have encountered problems with my credits and IEP, and Richard helped me solve them in a timely manner so that I can concentrate on my final work. In addition, I am also very grateful to Dr. Stefano Speretta for taking time out of busy works and becoming one of my committee.

I want to thank several partners in our group who have moved forward together, including Martijn, Elke, Felix, Brenda, and Calum. We have talked about happy things during our weekly coffee break, which is very precious for me to communicate with my peers. In addition, we supported each other academically. One of the ideas in my thesis was inspired by a question that Martijn posed after a presentation where I reported on my progress. I am pleased to meet these friends in Raj's group.

I am also very grateful to some friends in my daily life. I often cook with my neighbors Yongkang and Fang. Of course, I have to mention Ximei, Yuanyuan, and Xiaoyao. We discussed various things about daily life from time to time, especially Ximei. I am delighted to enjoy dinners and watch movies with her. Two friends of mine are actively applying for Ph.D., Jianing and Yanbin. We also exchange opinions and share experiences from time to time. Some friends remotely encouraged me during the journey, especially Tanguy, Hao and Han. Some too many friends have supported me in the past two years. Although I cannot list their names one by one, I still want to express my gratitude. I hope we can keep in touch in the future and continue to walk along the

journey together.

Most importantly, I would like to thank my parents. As the only child in the family, it is always a pity that I cannot be reunited with my parents in the past two years. My parents and I have long video calls every week, and they try their best to understand the various difficulties I encountered and provide emotional and financial support. Although we are thousands of miles apart, we will always be a family of heart to heart.

In addition, I would also like to thank the Chinese Embassy for all kinds of support provided to expats so that we overseas students can feel the care of our motherland. I also thank TU Delft and CAS group for their efforts in difficult times for our students to receive a complete learning experience as much as possible.

In the past two years, there have been too many magical experiences and too much gratitude. It is always hard to say goodbye. I will carry on with the knowledge and skills I learned and the kindness I received deep in my heart.

Bichi Zhang B.Sc.
Delft, The Netherlands
October 12, 2021

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction

In the past years, with the rise of new technologies such as deep learning, the realization of artificial intelligence has become an increasingly widely discussed topic. Autonomous driving technology is one of the most attractive branches under the framework of artificial intelligence. More and more companies and research institutes are focusing on vehicles, an indispensable tool for human travel. Almost every day, people come into contact with all kinds of vehicles, and the various problems these machines bring, such as traffic jams, car accidents, etc., have also affect people's daily life, even lives. Autonomous driving is regarded as an essential means to solve these types of problems that caused by human drivers. Ideally, an autonomous system can optimally plan the path, automatically avoid obstacles, intelligently analyze the environment, etc.

Traditionally, an autonomous driving system consists of four modules: perception, positioning, planning (navigation), and control, as seen in figure 1.1. Among those modules, the perception module uses vision-based sensors to perceive the vehicle's surrounding environment to achieve a real-time understanding of the surrounding environment. Finally, it sends results to the planning module for further analysis to decide whether to perform specific actions. The localization module uses GNSS (Global Navigation Satellite System) or RTK (Real-Time Kinematic) technology for real-time positioning of the vehicle. Some systems also employ IMUs (Inertial Measurement Units) to track the current pose or state of the vehicles. Therefore, the planning module can perform optimal path-finding based on location information. The planning and navigation module combines the information collected by various sensors, and makes decisions. They include high-level path plannings, as well as low-level tasks, such as obstacles avoidance. The control module is mainly based on the results of the decision made by the planning module. It controls the vehicle to speed up, slow down or make turns given different orders from the decisions. In a word, it changes the states of the vehicle through control orders. In the past the orders were given by the drivers, now for autonomous systems they are given by the planning modules.

Figure 1.1: The basic structure of autonomous vehicles.

With some application of intelligent transport and smart city, planning and naviga-

tion will combine the traffic conditions, historical data, and road data of the entire city to make more thoughtful decisions. The planning and navigation module macroscopically plans the route and direction of the vehicle from a relatively high level. In contrast, the perception module is more specific. Its purpose is to help the vehicle recognize the road and surrounding environment to plan more wisely on local processes (for example, change lanes to overtake the car in front of the automated vehicle). In addition to SLAM (Simultaneous Localizing And Mapping) and decision-making (path planning), the core of realizing autonomous driving includes tasks like vehicles, pedestrians, obstacles and traffic light recognition, lane line maintenance, etc. These functionalities are all integrated within the perception module. The boost of computer vision technology brought by deep learning significantly promotes the reliability of robot perception. Figure 1.2 illustrates some typical perception tasks for driverless vehicles.

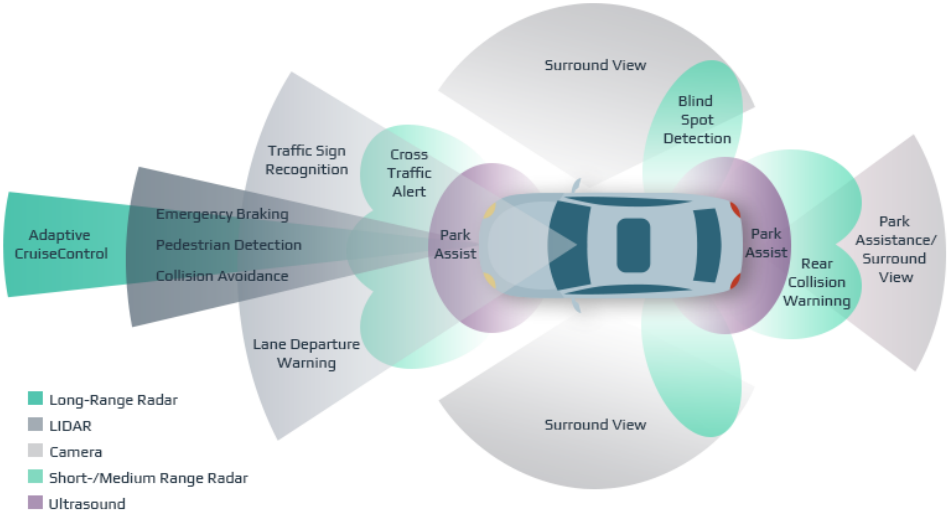


Figure 1.2: An example of perception tasks that automate the vehicle [24].

1.2 Perceptual Sensors and Data Fusion

The perception module is considered to be the eyes of the autonomous driving system. Ensuring these eyes can work functionally, efficiently, and stably has been challenging for autonomous-driving system-design engineers. A relatively reliable technical solution at the moment is to set up multiple types of sensing sensors, even to achieve redundancy, thus securing that the perception ability to the outside world can always be maintained. In many cases, due to technical limitations, the accuracy of the sensor's perception of the environment cannot meet the requirements. Therefore, using the information of multiple sensors to make a more comprehensive understanding of the external environment can effectively improve the system's reliability.

A simple example is the combination of camera and LiDAR. The camera can obtain a relatively stable perception ability during daytime and make more accurate classifications of obstacles. However, at night, due to some factors such as exposure time and

weak ambient light, its perception performance is quite affected. LiDAR is an active sensor. It emits and receives reflected laser beams to perceive the environment and obtain information from the outside world. This method will not be disturbed by day or night. Although the combined camera and LiDAR are redundant for the perception module, when the reliability of one or several sensors is reduced, the remaining sensors can still keep the perception module functional. The fusion of multiple sensors ensures that the device's essential functions will not be seriously affected. This idea is also one of the reasons why multi-sensor fusion technology is comparably reliable.

From another perspective, the reason for multi-sensor fusion is that there are complementarities between multiple sensors in terms of information. The most common camera-LiDAR fusion system is a typical combination. Among them, the type of information provided by the camera and the LiDAR are complementary. LiDAR can obtain more accurate 3D position information, which is an unparalleled advantage compared to the 2D plane image returned by the camera. Conversely, the camera can obtain richer color and texture information, just like a human eye. Unfortunately, a single optical camera cannot directly provide the target's distance information or depth information. Fusing these two kinds of sensors also makes the perception dimension of the autonomous driving system have a more remarkable improvement.

Although in the above discussion, we often take the examples of cameras and LiDARs. However, the perception module of the autonomous driving system does not only have these two kinds of sensors. Millimeter-wave radar (Mmw-Radar) or ultrasonic sensors are also commonly used in autonomous driving (for example, parking distance control, which has been applied for many years). We list a table to summarize the characteristics of popular sensors for vehicle perception, as seen in table 1.1.

Sensor	**Range**	**Functions**	**Advantages**	**Drawbacks**
Camera	50 m	Perceiving the surroundings based on computer vision algorithms	Low cost, mature technology, rich information about the patterns of objects	heavily dependent on ambient light, easily affected by bad weathers
Mmw-Radar	250 m	Perceive a wide range of objects, often used in adaptive cruise system	Robust to weather and time, long-range detection, high accuracy	No imaging ability, detection valid only in 2D, not sensitive to horizontal objects, low quality on human detection
LiDAR	100 m	Obstacle detection, plane detection, object tracking	High accuracy, high resolution, fast response time	High cost, complex production process, sensitive to extreme weather
Ultrasonic Radar	3-5 m	Low speed scenarios, mainly for parking systems	Low cost, high accuracy in short range, robust to light	Only feasible for low speed and short range cases, signals are sensitive to interference

Table 1.1: Comparison of Popular Perceptual Sensors [49] [50]

From the perspective of level classification, there are currently three fusion levels,

namely low-level fusion, middle-level fusion, and high-level fusion. Low-level fusion is known as raw-data fusion. It directly fuses the raw data from sensors without advanced processes like feature extraction. It is conceivable that this level of fusion can essentially be understood as this system is a super sensor, which can simultaneously collect LiDAR point clouds and camera images. As for the data processing, it is directly handed over to the subsequent algorithm for further analysis. Middle-level fusion is feature-level fusion. That is, feature extraction is performed on the original data of the camera and LiDAR, and decisions of object classification are made by other algorithms based on the fused features. High-level fusion usually refers to decision-level fusion. The data of the camera and LiDAR are processed separately for target recognition, and the recognized result is finalized (for example, whether it is a specific target). Due to the requirements of different goals, different levels of fusion take specific types of information into account. Therefore, their algorithms also have specific characteristics. Nevertheless, the ultimate goal of different fusion levels is to perceive the environment better.

1.3 Sensor Calibration

A critical point for achieving multi-sensor information fusion is to calibrate multiple sensors in advance. The calibration process of sensors can be roughly divided into two categories: temporal calibration (also known as time synchronization) and spatial calibration (also known as sensor alignment [7]). Sensor time synchronization is necessary for sensor fusion because different sensors may have different system clocks and sampling frequencies. Therefore, their sampling times may not be the same. The time synchronization allows their results to match at the same time frame. In engineering, time synchronization is divided into two schemes: hardware synchronization and software synchronization. Hardware synchronization refers to the simultaneous sampling of sensors by adding hardware triggers during system design, while software synchronization calculates the time difference between frames through a software algorithm, thereby realizing time synchronization through frame alignment.

Sensor spatial calibration refers to aligning the sensor's coordinate axes through software algorithms, which is one of the reasons that it is also called as sensor alignment. The spatial calibration solves such a problem: how to assure that an object A detected by the laser point cloud exactly corresponds to an object B on the camera image? To clearly express the idea, we can raise another question from a simple example: How to confirm that two coordinates coincide as a single point in space? Generally speaking, when the coordinates of these two points are equal in the same coordinate system, they must coincide. Similarly, there are now two sensor coordinate systems. A single point may be denoted with different coordinates in different coordinate systems, but their coordinates should coincide through a specific conversion algorithm. In this case, if object A coincides with object B the correct coordinate transformation, A and B should be the same object in the real world. Therefore, the results of LiDAR and camera can be errorlessly merged, and the question of spatial calibration raised above has a theoretical answer. It can be noticed that the problem of spatial calibration is actually to find the specific conversion algorithm, which is the function that can accurately describe the spatial relationship between sensor coordinate systems. Note that in some papers

which mainly apply optical cameras as one of the sensors, the spatial calibration between the sensor and the outside world (or other sensors) also known as extrinsic calibration, where extrinsic means the relationship between the sensor and the external world (other sensors).

Most of the extrinsic calibration methods use artificial objects to ease the difficulty of feature matching. Traditional methods use checkerboards and require multi-view frames [51]. Some methods use multiple boards to avoid working in multiple frames, with people moving target around to provide multiple frames [30] [9] [16]. Later on, some other shapes of boards and objects are developed. They apply designated geometric shapes to avoid using onboard patterns [1] [34]. Although the target-based methods have longer development, human intervention is the major drawback with respect to the level of automation. Therefore, targetless methods are more attractive in applications.

1.4 Research Topic and Contributions

This thesis aims at developing a target-less extrinsic calibration algorithm that requires no specific target to achieve the goal of parameter estimation. To achieve this aim, main question listed below must be discovered.

- How to associate information from camera and LiDAR without any known target?

In order to answer the question, data models for camera and LiDAR sensors are formulated. By exploring some state-of-the-art methods, their advantages and disadvantages are discussed. And then, we develop a novel pipeline called MulFEA (Multi-Feature Edge Alignment) that answers the main question while improves the performance compared to current methods. In order to verify the performance, the experiments are carried out based on KITTI dataset [14]. Our work provides several general contributions:

- The imaging model of both camera and LiDAR are derived mathematically. The camera-LiDAR joint calibration model is formulated.
- Three state-of-the-art methods are recreated and analyzed. Some of their limitations are discussed.
- An algorithm called MulFEA is developed in order to solve the camera-LiDAR calibration problem. MulFEA is able to work in a natural environment without any calibration tools. It reduces the total amount of calculation by minimizing the processing during optimization. In this algorithm, the 3D-2D calibration problem is converted into a 2D-2D problem. The occlusion effect of LiDAR realizations are reduced. MulFEA exploits multiple features from LiDAR, improving the overall performance. It uses a new criteria that measures the performance of extrinsic calibration is developed.
- An analysis of the MulFEA is made. Some of the limitations and possible improvements are presented.

1.5 Structure of the Thesis

In this thesis [1], we mainly study the target-less extrinsic calibration of monocular cameras and LiDARs, where extrinsic calibration refers to estimating and correcting the calibration parameters between sensors' coordinate systems. In Chapter 2, we will briefly introduce the entire camera-LiDAR calibration development process, starting from and some key achievements in the recent development. We will also analyze the current mainstream algorithms and our choices in this project. In the Chapter 3, we will explain the imaging models and characteristics of cameras and LiDAR sensors from a mathematical perspective. According to the models, we will also derive the specific mathematical problems for joint calibration, and the 6-DOF (6 degrees of freedom) extrinsic parameters to be estimated. In the Chapter 4, we will analyze and reproduce three classic highly-cited methods in the field of automatic calibration without targets and mainly use the KITTI data set to test the three algorithms. Some of their limitations are discussed. In Chapter 5, we will briefly introduce the overall MulFEA framework and ideas, and then dive into multiple small sections to describe the details. At the same time, we will concretely illustrate some results in each stage. The experiments will be carried out and presented in Chapter 6. Firstly, we introduce the content and types of data related to the KITTI Dataset. We will also present the data processing mode from the perspective of programming, including how to analyze the raw data from sensors and add features to each point in our pipeline. We will select some typical frames to test our algorithm and compare the single-frame performance of different algorithms. Meanwhile, we will also extend our algorithm to multiple frames and demonstrated better optimization capabilities. In the concluding chapter, we will review the entire thesis, summarize our contribution to this project, discuss the limitations of the current framework and more possible improvements, and propose reasonable suggestions on future solutions to this problem.

[1] This study is part of the ADACORSA project [6].

2 Related Works

In this chapter, we discuss the development and some related works of the extrinsic calibration between camera and LiDAR. The essence of the camera-LiDAR joint calibration problem is to estimate the relative pose between these two sensors. The original question can be traced back to the intrinsic and extrinsic calibration of the monocular camera [52]. In the process of monocular camera calibration, the traditional scheme uses the calibration board with a chessboard pattern to obtain the intrinsic projection matrix, and then solve the rigid extrinsic transformation from camera coordinate system to world coordinate system according to the different poses of various calibration boards. The pose of the world coordinate system is usually predefined. In the problem of camera-LiDAR joint calibration, the known preset world coordinate system has become an unknown LiDAR coordinate system, so obtaining the transformational relationship between camera and LiDAR has become the main problem of extrinsic calibration.

2.1 Target-based Methods

Zhang and Pless firstly introduced the checkerboard to calibrate the single line LiDAR and monocular camera in 2004 [51]. They use the geometric characteristics of the calibration board and establish the relationship with the calibration plane coordinate system by calculating the normal vector of the checkerboard. On the other hand, the camera can also build a mathematical connection with the calibration board. The positional relationship between these planes and lines produces constraints on extrinsic parameters. By repeatedly configuring the position of the calibration board, the matching of multiple groups of 2D points and 3D points can also be found, and the problem can be solved by the constrained optimization method. However, due to the limitation of plane-line correspondence, this method needs multiple groups of pose to solve. Unnikrishnan et al. utilize the plane-plane correspondence but still requires at least three sets of different poses [46]. In order to reduce repeatedly moving the checkerboard to create multi-pose correspondence, Geiger et al. place multiple checkerboards with different positions in the same field of view to create the connection between matching points and planes [17]. Since there are multiple calibration boards in the scene, their method only needs one shot. Generally speaking, the calibration board is only exploited to create plane and object edge properties. Many people also try to design objects with different shapes to improve the performance of the calibration algorithm. Pereira et al. use a shpere as a calibration object [34]. Others use all different types of objects, such as rings [38], V-shape objects [9] [42], cuboid [37], trihedron [19]. The underlying idea, still mainly based on plane-line, plane-point, and plane-plane correspondence. The demand for labor makes it difficult for these methods to go out of the laboratory or factory. When the calibration parameters drift after a long-term utilization, the customers of automatic

driving products cannot solve the problem themselves. Therefore, the ideal autonomous systems are demanded to calibrate perceptual sensors by adaptively relying on various features of arbitrary environment automatically.

2.2 Targetless Methods

In the recent decade, with the rise of technologies and concepts such as machine learning, electric vehicles, and higher-level automated driving, the targetless calibration approaches, which is also of research significance, has only begun to be widely studied. Since target-less calibration can theoretically configure the extrinsic calibration parameters of the sensor in any environment, compared to the target-based method that requires a calibration board or a specific calibration object, it can improve the robustness of the autonomous driving system and reduce the human intervention.

2.2.1 Statistical-Feature-based Methods

One of the highly cited methods is based on mutual information, which was proposed by Pandey et al. in 2012 [32]. Statistical-based methods such as mutual information and correlation coefficients are usually used in image registration problems. Pandey applies this idea to the targetless camera-LiDAR calibration. It maximizes the mutual information between the grayscale intensities of the camera image and the reflectivity values of the LiDAR. However, due to the relatively significant difference between the modalities of these two sensors, the reflectivity of the LiDAR is not always consistent with the change of the camera's intensity values. This leads to the problem that extrinsic parameters to be estimated are often stuck in the local optima during the optimization process. In 2016, Irie et al. extended feature-based statistical algorithms to higher dimensions [21]. The LiDAR features are expanded from reflectivity to more, including discontinuities and normal vector features. On the image side, the algorithm considers three-dimensional RGB information and edge intensity as image features. The author uses square-loss mutual information to replace ordinary mutual information and reduces the influence of outliers by eliminating logarithm. The overall calibration results witness a smoother shape in terms of the cost function. Since the LiDAR they use is a high-density 3D lidar rather than a multi-line LiDAR commonly used in autonomous driving systems, its application in long-distance sparse information scenarios needs further verification. A critical drawback is that their algorithm contains massive data preprocessing, and the total computational efficiency is relatively low.

2.2.2 Edge-Alignment-based Methods

The algorithm based on depth discontinuity proposed by Levinson and Thrun in 2013 tries to solve target-less extrinsic calibration parameters from another standpoint [25]. They use the unique sensing capabilities of sensors to capture the edge information of any object that is ubiquitous in the road environment. This algorithm matches the edge information obtained by the depth difference of the scanning points between each line of the LiDAR and the difference of the camera grayscale image. Theoretically, when the

correct calibration parameters are given, the object's edge should coincide in the LiDAR and camera coordinate systems. Their algorithm is in online fashion, and its advantage is its calculation speed. However, similar to previous methods, due to the arbitrariness of the environment, when the number of frames provided is small, the algorithm will also face the phenomenon of local optima. Nevertheless, since its primary purpose is to deal with slight sensor drift or disturbance, this algorithm can handle this problem in a relatively small interval. Taylor et al. use gradient orientation measurement to match the LiDAR reflectivity map with the camera grayscale image [44]. Another novel algorithm based on edge matching is the idea of solving the calibration-fusion joint problem, which is proposed by Castorena et al. in 2016 [5]. This algorithm first projects the laser point cloud onto the camera image plane and uses projected sparse points for depth completion. Through the dense depth map, the boundary information of the 3D laser point cloud is transformed into the boundary information of the two-dimensional depth map. A loss function similar to the cosine correlation coefficient is applied to match the depth map's boundary with the camera image's boundary. However, since its optimization step includes a relatively redundant depth map completion step, the performance of the entire algorithm is inefficient. Similarly, due to the lack of proper processing of the laser point cloud in this method, the final matching effect also has a relatively large room for improvement.

2.2.3 Ego-Motion-based Methods

Recently, with the in-depth study of SLAM problems, some methods exploit the trajectory matching based on vehicle's ego-motion models [35] [7]. In their algorithm, the commonly used process is to calculate visual-based odometry and laser-based odometry separately and then match their path with Gaussian process. The method proposed by Wang et al. uses SfM (Structure from Motion) to reconstruct the scene in 3D, mark the SIFT feature points, and then optimize the external parameters by minimizing the projection error of the feature points between different frames [48]. Motion-based methods are generally less accurate than feature-based methods because the estimation error of the odometry in the natural environment will also bring errors to the calibration.

2.2.4 Neural-Network-based Methods

Some recent algorithms use deep learning and neural networks to solve calibration problems. Zhu et al. use PSP-Net to partition the vehicles in the image semantically and then use the semantic map area to match the point cloud with all ground points removed [54]. Similarly, Ma et al. use BiSeNet-v2 to semantically segment the lane lines and street light poles in the image [27]. To extract the same objects in the point cloud, they select the street light poles filtered by RANSAC and the lane lines by filtering the difference in reflectivity. Finally, the match is performed between the image with poles and lanes and the point cloud with poles and lanes. In the existing methods, there are also neural networks designed explicitly for camera-LiDAR calibration tasks. RegNet is a network that uses CNN to infer 6-degree-of-freedom extrinsic parameters between sensors [40]. The idea comes from image-image registration. CalibNet draws on the structure of ResNet, reconstructs the laser point cloud with depth values, and

uses photometric loss and point cloud distance loss to optimize network parameters [22]. However, the neural network-based method requires a large amount of data for pre-training the network and has relatively high computational power requirements.

The existing calibration problems have various attempts with advantages and disadvantages. In this work, we propose a novel workflow MulFEA, which is based on edge alignment. It improves and supplements the shortcomings of several current edge-alignment methods, and uses multiple features to improve the reliability of the results.

3 Problem Formulation

Calibration between sensors is a process that unifies the measurements from different sensors. For example, a camera-LiDAR system is widely seen for robots as perception tasks. A calibration process is crucial to utilize the acquired data correctly. The calibration problem can be categorized into two main aspects: spatial calibration and temporal calibration. Spatial calibration focuses on the geometry between sensors in the space, while temporal calibration is more of a synchronization step between sensors with different timestamps. In this work, our attention is on the extrinsic calibration between camera and LiDAR, which belongs to the spatial calibration category.

There are also two types of calibration with respect to sensors: intrinsic and extrinsic calibration. The intrinsic calibration determines the internal mapping relationship of the sensor, such as the camera's focal length and the distortion coefficient. In contrast, the extrinsic calibration describes the transformation relationship between the coordinate systems of sensors, such as rotation and translation. In other words, the aim of extrinsic calibration between camera and LiDAR is to explore a mathematical formulation that describes the spatial relationship between two sensors.

In this chapter, the problem of camera-LiDAR extrinsic calibration will be explained and formulated mathematically. To explicitly describe the calibration of these two different sensor modalities, we will first introduce the different imaging models of the camera system and LiDAR sensor in section 3.1 and 3.2. Then we will explain the spatial alignment of different modalities and formulate the mathematical model of extrinsic calibration in section 3.3.

3.1 Camera Model

3.1.1 From Lenses to Pinhole

Optical camera is one of the most common measuring sensors that is widely used nowadays. It captures the ambient light in the environment through its lens system and outputs digital images by saving all the information on coupled charge device (CCD). Modern cameras are complex and comprise of multiple sets of lenses, in order to capture more light, provide wider range, and record more details. Figure 3.1.1 shows a complex camera mechanism.

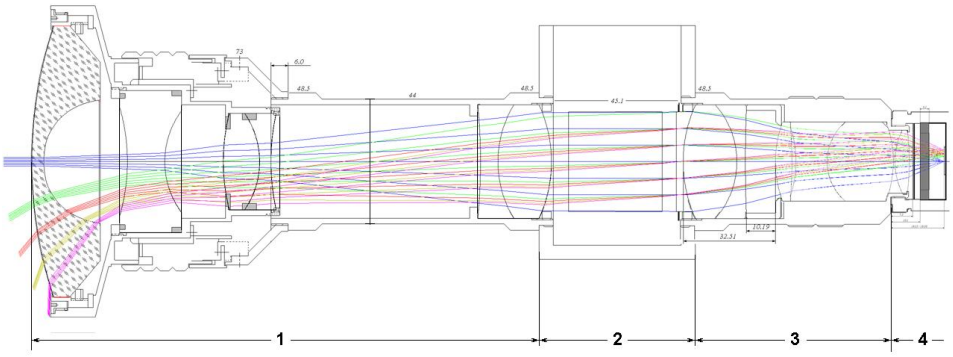


Figure 3.1: The NORUSCA II all-sky camera mechanics [41]

These kind of advanced camera models are difficult to analyze. To generally analyze the optical model, we start with introducing the basic optical lens paradigm, shown in Fig 3.2

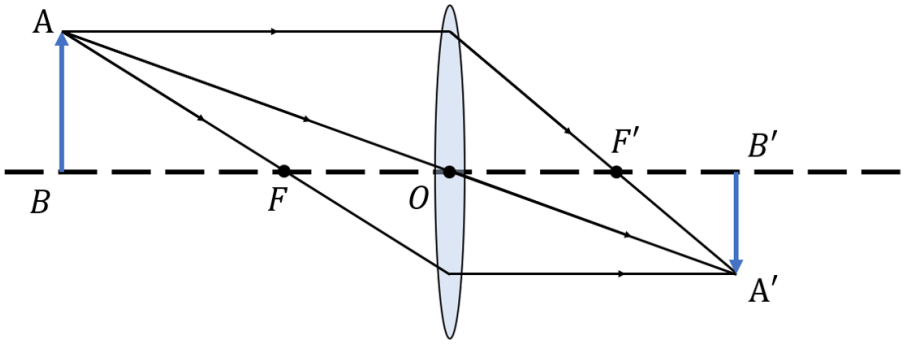


Figure 3.2: The convex lens model

The arrow AB represents an object, and its image $A'B'$ is on the other side of the lens. O is called the optical center, and both F and F' are the focal points. The straight-line BB' is the optical axis of the lens. Three rays $AF'A'$, AOA' and AFA' are the equivalent light paths. The underlying principle can be expressed in (3.1):

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v} \ , \tag{3.1}$$

where $u = OB$ represents the distance between the lens and the object, and $v = OB'$ is the distance between the lens and the image. $f = OF$ is the focal length. This is a nonlinear expression, describing the relationship between the object distance and the image distance.

The equivalent convex lens imaging model can represent the actual camera model. The CCD imaging plane is placed near the focal point. The pinhole camera model is widely used to describe the imaging process in most camera sensing tasks. It simplifies the general camera model and mimics the light path inside the lens sets. It approximates

the equivalent light path using a linear approach. If we only concentrate on the ray AOA' and discard the rest, it will become the pinhole camera model. The mechanism is simplified to a single straight light path passing through the optical center. In this model, even the lenses can be simplified as a tiny aperture since we only consider the light going through the optical center, where the light path will not be affected by the lens at all. This is also known as the camera obscura effect. Figure 3.3 illustrates the pinhole camera model.

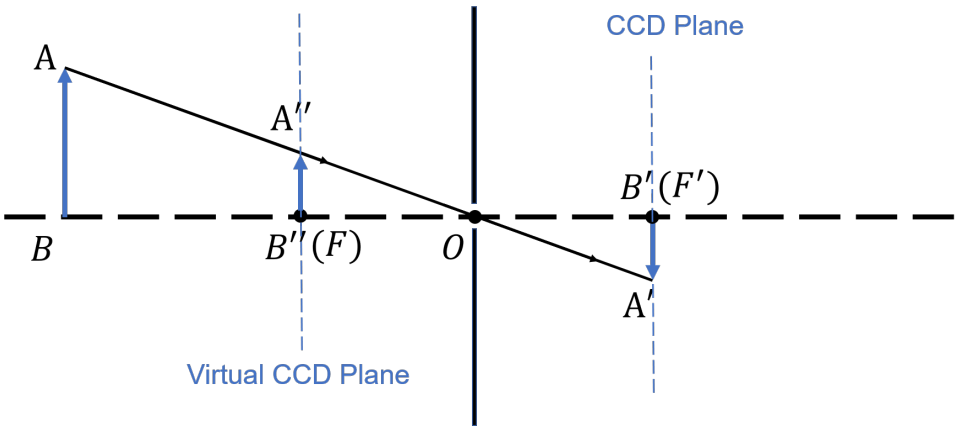


Figure 3.3: The pinhole model

In the pinhole camera model, the relationship between the object and the image is linear. From Fig 3.3, we can clearly see that

$$\frac{AB}{A'B'} = \frac{OB}{OB'}$$

by properties of similar triangles. The focal length in the pinhole camera model no longer represents the convergence points of light. Instead, it only shows the distance between the optical center to the CCD plane. To simplify the model even further, we can use the symmetric property to assume a virtual CCD plane on the same side as the object. Therefore, the right part of the figure can be discarded as well from our diagram. The similarity properties between triangles, in this case, still hold. Furthermore, if we expand our model into 3D space, these similar triangles also remain.

3.1.2 Camera Imaging Model in 3D

In this section, the 3D camera imaging model will be derived mathematically. The concept of pinhole camera model is still applicable in 3D cases. Figure 3.4 presents the geometric model of camera imaging procedure.

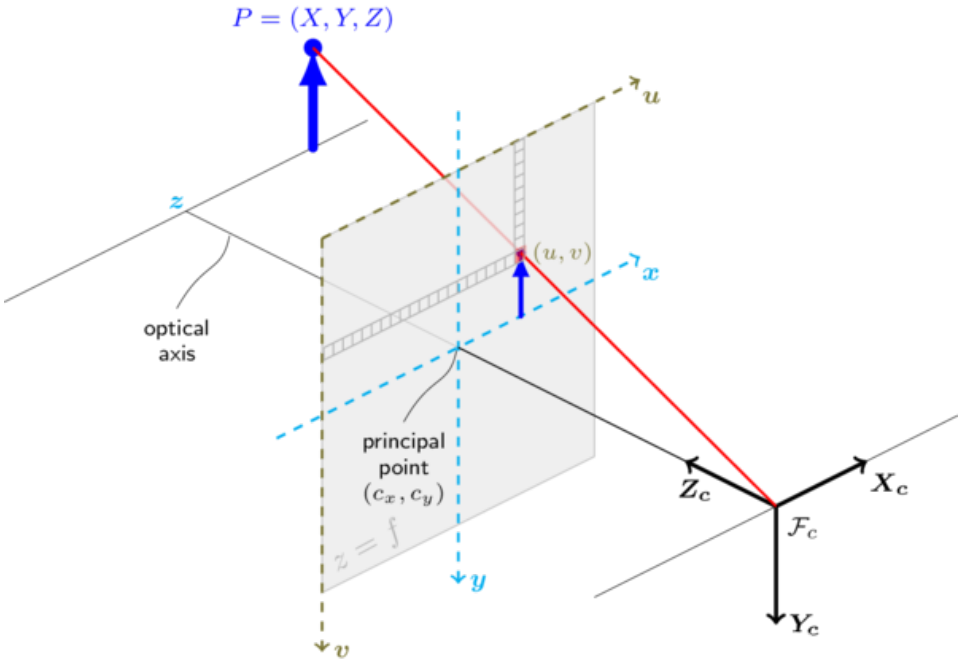


Figure 3.4: The camera imaging model [8]

The imaging process is a mathematical function that converts a 3D point in the space to a 2D pixel on the digital image. Four coordinate systems are used in the derivation of intrinsic model to describe the imaging process explicitly. The four coordinate systems are:

- The 3D **world coordinate** $\mathcal{C}_w$, which is the Cartesian coordinate of the 3D real world. Its center can be defined arbitrarily in the space.
- The 3D **camera coordinate** $\mathcal{C}_c$, which uses the same metric as world coordinate (for example, meters) but the origin is located at the optical center $\mathcal{F}_c$. The axes are fixed to the imaging and optical axis, where $\mathcal{Z}_c$ is pointing to the front direction of the camera and perpendicular to the imaging plane. According to the right-hand rule of the coordinate system, the $\mathcal{X}_c$ and $\mathcal{Y}_c$ are parallel to the horizontal and vertical edges of the CCD imaging plane, respectively.
- The 2D **image coordinate** $\mathcal{C}_i$, where the origin locates at the center of the imaging plane inside the camera, i.e., the CCD device. The axes are parallel to the edge of the rectangle imaging device, denoted as x_i and y_i. It often uses meters or millimeters as metric.
- The 2D **pixel coordinate** $\mathcal{C}_p$, where the metric is discrete pixels and the origin is the first pixel at the top left of the image. The axis u is along the top edge of the image and v is directing to the left edge of the image.

The projection from the 3D camera coordinate $\mathcal{C}_c$ to the 2D pixel coordinate $\mathcal{C}_p$ is known as the camera intrinsic model. It is inherent to the camera itself, containing some intrinsic parameters, i.e., focal length, size of each pixel on the CCD plane, etc. They are

assumed static no matter how the external environment changes. The mapping between 3D world coordinate $\mathcal{C}_w$ and 3D camera coordinate $\mathcal{C}_c$ is called the camera's extrinsic model. It describes the geometric relationship between the outside world and the camera. In this section, we only focus on the intrinsic model and derive the mathematical transform.

Imagine we have a point P_c in 3D space, given the coordinate (x_c, y_c, z_c) in camera coordinate system (noted with c as subscripts). Based on the similar triangles comprised by the light (red line in the image above) and the optical axis in the projection model, we can build the equations of point P_c and the corresponding point $P_i(x_i, y_i)$ projected on the image plane:

$$\begin{cases} \frac{x}{X} = \frac{f}{Z} \\ \frac{y}{Y} = \frac{f}{Z} \end{cases}, \tag{3.2}$$

where f is the focal length of the pinhole camera model, indicating the distance from the principal point to $\mathcal{F}_c$, the center of the pinhole. In this way, we build the connection between coordinates systems $\mathcal{C}_c$ and $\mathcal{C}_i$.

The relationship between image coordinate system $\mathcal{C}_i$ and pixel coordinate system $\mathcal{C}_p$ is a 2D-2D rigid transform. Note that these two systems use different metrics: one is the meter, and another is the pixel. We denote k and l as horizontal and vertical scaling parameters from meters to pixels, which defining the transform as:

$$\begin{cases} u = \frac{x_i}{k} + c_x \\ v = \frac{y_i}{l} + c_y \end{cases}, \tag{3.3}$$

where c_x and c_y are the bias from the center of imaging plane to the origin of the pixel coordinate system along the x and y axes, respectively. The physical interpretation of k and l is the horizontal and vertical size of each pixel on the CCD plane. In other words, a single pixel has k meters long and l meters wide. Note that u, v, c_x and c_y are all in number of pixels, not meters anymore. The corresponding focal lengths in pixels f_x and f_y can also be defined by the scaling parameters k and l, as

$$\begin{cases} f_x = \frac{f}{k} \\ f_y = \frac{f}{l} \end{cases}. \tag{3.4}$$

If we gather the equations above into matrix form, a more elegant expression can be concluded. The transform from pixel coordinate system $\mathcal{C}_p$ to image coordinate system $\mathcal{C}_i$ is shown in (3.5).

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{k} & 0 & c_x \\ 0 & \frac{1}{l} & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}. \tag{3.5}$$

The same procedure can be applied to the mapping from camera coordinate system $\mathcal{C}_c$ to the image coordinate system $\mathcal{C}_i$ as well, shown in (3.6).

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f_x \cdot k}{z_c} & 0 & 0 \\ 0 & \frac{f_y \cdot l}{z_c} & 0 \\ 0 & 0 & \frac{1}{z_c} \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{3.6}$$

In summary, the whole camera imaging process, the transform from 3D points to 2D pixels is illustrated by the intrinsic model, shown in (3.7), where the matrix is called the intrinsic matrix.

$$z_c \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{3.7}$$

We can see that the parameters k and l are dropped in the process. The remaining 4 parameters f_x, f_y, c_x and c_y compose the intrinsic matrix. With the intrinsic matrix at hand, a point in 3D can be projected onto the image explicitly. In other words, if we have a point (x_c, y_c, z_c) in 3D, we can know its corresponding 2D location on the digital image. Generally speaking, this is how the camera sensor captures information from its field of view.

The derivations above are a basic intrinsic model case, where no distortion and skewness happens during the imaging process. For further derivations with regard to more intrinsic parameters with more complicated models, readers can be referred to [43].

3.2 LiDAR Model

LiDAR stands for light detection and ranging. This technique is widely used in distance measuring and environment sensing tasks, especially for robot perceptions. A common LiDAR system comprises a scanner, a set of emitters, receivers, and signal processing circuits. The LiDAR in figure 3.5 is an example of a modern LiDAR system. It includes 64 laser emitters located at the side part of the device. The receiver contains two big lenses, diverting all the receiving laser beams to different sensing components for different processes according to the incident angles.

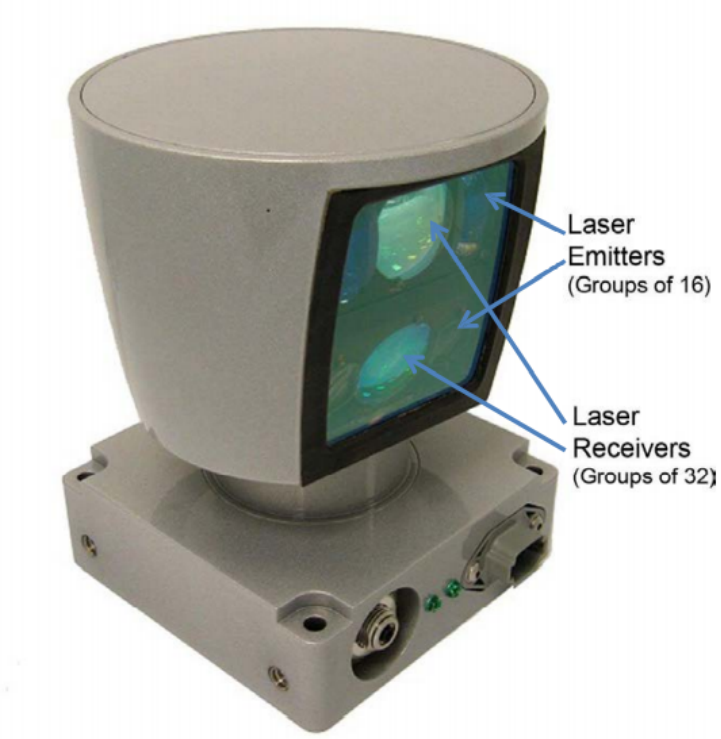


Figure 3.5: The Velodyne HDL-64 LiDAR [20]. The laser emitters emit laser beams, meanwhile the receivers in the middle of the window receive the reflections of them. The head of the LiDAR is spinning, so that the window will cover 360 degree of the surrounding environment.

The laser possesses advantages such as spatial coherence, which allows the laser to collimate and focus within a narrow light path over a long distance. LiDAR systems can measure the range by collecting the reflected laser beams by exploiting the laser's quality. There are also some other methods, i.e., FMCW (Frequency Modulated Continuous Wave) LiDAR borrows the idea of the Radar system, estimating the range by calculating the phase and frequency shifts of the modulated signal. The majority of the LiDAR systems on autonomous vehicles use the time of flight (TOF) technique for ranging because it is fast in terms of the speed of distance measuring. Although the precision is centimeter-level, which is not as good as FMCW LiDAR at millimeter level, it is still enough for significant tasks in autonomous driving scenarios.

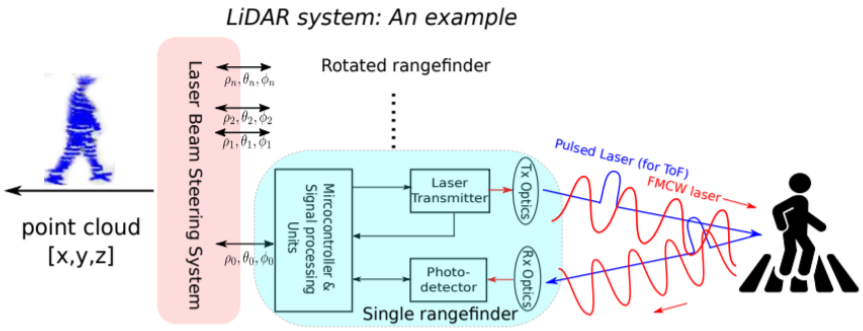


Figure 3.6: A example of LiDAR workflow [26].

In TOF LiDAR, the laser emitter emits very short laser pulses, and diffuse reflection occurs after encountering the target. A tiny part of the light returns along the original path and is received by the receiver. The target distance is obtained by calculating the time difference between laser emission and reception. Equation (3.8) shows the mathematical expression of measured distance value.

$$r = \frac{1}{2n} c\Delta t \tag{3.8}$$

In the equation above, c is the light speed, n is the refractive index of the propagation medium (for air, it can be approximated as 1), Δt is the time of flight between transmitting and receiving, and r is the distance between the LiDAR and the object.

After illuminating the object, part of the laser beam is reflected directly back to the LiDAR due to the diffusion of the surface of the object. Inside the receiver module in LiDAR, the photon-detecting sensor collects the received laser beams and converts them into digital signals. The power of the digital signal of a single impulse beam received P_r is defined as [47]:

$$P_r = E_p \frac{c\eta A_r}{2r^2} \cdot \beta \cdot T_r \tag{3.9}$$

, where E_p is the energy of the transmitted pulsed laser, c is the light speed. A_r denotes the area of receive aperture at a range r. β represents the surface reflectivity, which is strongly related to the surface's property and the incident angle. η is the overall system efficiency. The transmission loss coefficient T_r is related to the atmospheric quality in general. For example, laser faces huge losses during some foggy weather.

The receiving module also collects the azimuth and elevation angles. Therefore, with the range value r, the coordinate of a single point can be presented in a polar coordinate system. The reflectance value of LiDAR measurement is defined by the ratio of the power of the received laser and the emitted laser. By rotating the system, LiDAR is able to perceive the surroundings in 360°. When the emitter works at a particular sampling rate, the beam will form a circle in a sparse pattern.

The vertical field of view is obtained by stacking laser emitters vertically. Generally, in the market, 16 beams, 32 beams and 64 beams are commonly seen. Since the laser emitters are placed in a nearly-parallel way but with small angle between each other,

a wide-angle of view (sector area) can be covered. For autonomous vehicles, most of the setups are putting the LiDAR on the top of the automotive, so this type of LiDAR usually covers more lower areas than the upper parts, as shown in figure 3.7.

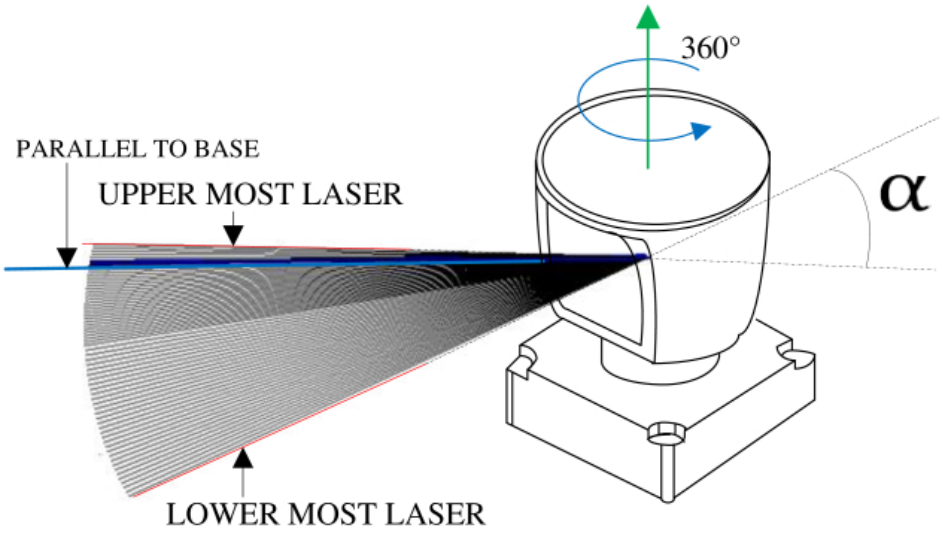


Figure 3.7: An example of field of view of LiDAR [20].

All the raw measurements collected from the receiver module are processed in the digital circuits. For each point, the time of flight and angles are converted into a Cartesian coordinate with 3D position information (x, y, z). The corresponding reflectivity value of the laser is also considered. In LiDAR coordinate system $\mathcal{C}_l$, x axis points toward the front, y to the left, and z to the up direction. In a single frame of LiDAR realization, we assume there are n points in total. With all the points coordinates (x, y, z) and reflectivity values r, the data model is therefore a matrix of n which stacks all the points from (x_0, y_0, z_0, r_0) to (x_n, y_n, z_n, r_n). This collection of points is also known as a point cloud.

3.3 Extrinsic Calibration: Combination of Modalities

3.3.1 Motivation

In the camera's intrinsic model, we illustrate how the camera captures the information and stores it into pixel form as an image. However, when analyzing images, which are the camera measurements, we cannot directly derive any useful distance information since all the intrinsic parameters are in pixels rather than meters (note that the vertical and horizontal sizes of the pixel are not part of the intrinsic parameters). More importantly, given a point (u, v) in the image, the 3D corresponding point (x_c, y_c, z_c) cannot be readily inferred without any additional information because the intrinsic model, in this case, is ill-posed. The lack of depth limits the camera-solely system if it is meant to perform as a perception module. Nevertheless, aside from this disadvantage, the camera images provide rich information on textures, patterns, colors, etc. These images are accessible

for humans to perceive and valuable for some perceptual tasks like classification and recognition.

Generally speaking, point clouds are sparse since vertically, there are limitations on the number of laser emitters, while horizontally, the sparsity is due to the speed of the rotation and the sampling rate of emitters. The disadvantage of sparse information is that there is little detailed information included. However, the point cloud contains the distance information that images do not have, which compensates for the shortcoming of the camera system.

Combining the information from both cameras and LiDAR systems is crucial since the lack depth information can be provided by LiDAR measurements. On the other hand, the deficiency of texture, color, and pattern information in the LiDAR point cloud is solved if we use that information from the camera measurements. The fusion of these two types of sensors exploits the complementary characteristics of different sensors and achieves better performances on autonomous perception tasks.

3.3.2 Extrinsic Calibration Model

Before utilizing any sensor fusion techniques, an important step is to register their coordinate systems. This step is also known as extrinsic calibration, a prerequisite for fusing information from different sensors. It builds a connection between camera images and LiDAR point clouds. With the extrinsic calibration model, each point in LiDAR coordinate system $\mathcal{C}_l$ can be linked with a corresponding counterpart in the digital image. In other words, extrinsic calibration of the camera and LiDAR answers an essential question: How to find the corresponding coordinate (x_c, y_c, z_c) in the camera system of a point (x_l, y_l, z_l) in LiDAR coordinate system $\mathcal{C}_l$?

As introduced in previous sections, both camera coordinate system $\mathcal{C}_c$ and LiDAR coordinate system $\mathcal{C}_l$ are 3D Cartesian coordinate systems with the same metric in meters. In any situation, this feature will not change. Thus, the extrinsic calibration can be described explicitly by rigid transform, which is only composed by rotation and translation, with shape and other qualities invariant. For any free rotation in 3D space, there are at least three parameters required, which are the three rotation angles along each coordinate axis. Similarly, for translation, there are also three parameters needed. Therefore, this transform is defined by six parameters. Three are the rotation angles along axes, while the rest three stand for the translations along axes.

In the following paragraphs, the extrinsic calibration model will be presented in a mathematical form. The extrinsic calibration parameter can be defined as

$$\boldsymbol{\theta} = (r_x, r_y, r_z, t_x, t_y, t_z) \tag{3.10}$$

, where r_x, r_y, r_z stands for rotation and t_x, t_y, t_z for translation. The rotation is achieved by using rotation matrix R. To simplify the notation, we denote c_x as $\cos r_x$, and s_x as $\sin r_x$. This simplification applies to rotations along y and z axes as well. The rotation along each axis can be defined as:

$$\boldsymbol{R_x}(\boldsymbol{\theta}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_x & s_x \\ 0 & -s_x & c_x \end{bmatrix}, \quad \boldsymbol{R_y} = \begin{bmatrix} c_y & 0 & -s_y \\ 0 & 1 & 0 \\ s_y & 0 & c_y \end{bmatrix}, \quad \boldsymbol{R_z} = \begin{bmatrix} c_z & s_z & 0 \\ -s_z & c_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The final rotation matrix $\boldsymbol{R}$ is therefore expressed as:

$$\begin{aligned}\boldsymbol{R} &= \boldsymbol{R_x R_y R_z} \\ &= \begin{bmatrix} c_y c_z & -c_y s_z & s_y \\ c_x s_z + c_z s_x s_y & c_x c_z - s_x s_y s_z & -c_y s_x \\ s_x s_z - c_x c_z s_y & c_z s_x + c_x s_y s_z & c_x c_y \end{bmatrix}\end{aligned} \tag{3.11}$$

The rotation matrix is an orthonormal matrix. This brings some useful characteristics, such that the transpose of $\boldsymbol{R}$ equals to the inverse of $\boldsymbol{R}$ and the determinant of $\boldsymbol{R}$ is 1:

$$\begin{cases} \boldsymbol{R}^\top = \boldsymbol{R}^{-1} \\ |\boldsymbol{R}| = 1 \end{cases} \tag{3.12}$$

The translation process is straightforward. It can be achieved by simply add some translation parameters at the corresponding coordinate. For instance, a point (x, y, z) can just be moved to $(x+t_x, y+t_y, z+t_z)$ by adding the translation parameters (t_x, t_y, t_z). Combining both rotation and translation, the conversion between camera coordinate system $\mathcal{C}_c$ and the LiDAR coordinate system $\mathcal{C}_l$ can be presented as follows:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \boldsymbol{R} \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} + \boldsymbol{t} \tag{3.13}$$

It can also be expressed using homogeneous coordinates. By augmenting with a forth term 1 in the coordinate, the translation term is merged into the matrix multiplication with rotation. Therefore, the process in (3.13) with both matrix multiplication and addition can be simplified as matrix multiplication only. We denote the transformation matrix as $\boldsymbol{T}$, where

$$\boldsymbol{T}(\boldsymbol{\theta}) = \begin{bmatrix} c_y c_z & -c_y s_z & s_y & t_x \\ c_x s_z + c_z s_x s_y & c_x c_z - s_x s_y s_z & -c_y s_x & t_y \\ s_x s_z - c_x c_z s_y & c_z s_x + c_x s_y s_z & c_x c_y & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

Therefore, using homogeneous coordinates, the rotation and translation processes are unified. It significantly reduces the complexity of the expression, especially when performing multiple times of rotation and translation. In this way, the rigid transform is described explicitly by the 4×4 transformation matrix $\boldsymbol{T}$. The transformation matrix $\boldsymbol{T}$ is also known as $SE(3)$ group (special Euclidean group). This group preserves the closed form for matrix multiplication. The inverse of the matrix intuitively represents the inverse of the rotation and translation process. Most importantly, the relative relationship between 2 points in a coordinate system remains static after the transformation. The extrinsic transform between LiDAR coordinate system $\mathcal{C}_l$ and camera coordinate system $\mathcal{C}_c$ is derived in (3.15).

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ z_l \\ 1 \end{bmatrix} = \boldsymbol{T} \begin{bmatrix} x_l \\ y_l \\ z_l \\ 1 \end{bmatrix} \tag{3.15}$$

When the intrinsic matrix also considered in our tranformation model, an elegant connection between the LiDAR point cloud and camera image is built. Equation (3.16) describes the relationship between camera image and LiDAR points with camera intrinsic model and joint extrinsic calibration model.

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \boldsymbol{KT} \begin{bmatrix} x_l \\ y_l \\ z_l \\ 1 \end{bmatrix} \tag{3.16}$$

$\boldsymbol{K}$ and $\boldsymbol{T}$ represents the intrinsic matrix and extrinsic matrix, respectively. Note that the missing depth z_c in camera imaging system is solved by multiplying the last row of $\boldsymbol{K} \cdot \boldsymbol{T}$ and the LiDAR measurements. This indicates the problem of lack of distance in camera system is solved when LiDAR information is available. In this way, the information from different sensors is fused since the association is built between the point cloud and image pixels. This process can also be interpreted as projecting laser points onto the camera image. The 3D position information is provided as another feature while the advantages of the 2D images preserve.

Normally, the number of input LiDAR points are enormous. Therefore, (3.16) can also be reformulated to process the whole point cloud at once. The input point cloud is denoted as $\boldsymbol{\psi} \in \mathbb{R}^{N \times 3}$, where N is the total number of points in the point cloud. Note that the normalizing term z_i^c for homogeneous coordinates is different for every single point.

$$\begin{bmatrix} z_1^c u_1 & z_1^c v_1 & z_1^c \\ \cdots & \cdots & \cdots \\ z_N^c u_N & z_N^c v_N & z_N^c \end{bmatrix} = \left(\boldsymbol{KT} \begin{bmatrix} \boldsymbol{\psi}^\top \\ \mathbf{1}^\top \end{bmatrix} \right)^\top = \left(\boldsymbol{KT} \begin{bmatrix} x_1^l & \cdots & x_N^l \\ y_1^l & \cdots & y_N^l \\ z_1^l & \cdots & z_N^l \\ 1 & \cdots & 1 \end{bmatrix} \right)^\top \tag{3.17}$$

Our goal, in this work, is to estimate the unknown extrinsic matrix $\boldsymbol{T}$. In traditional computer vision literature, this problem is known as camera pose estimation. Commonly, providing enough point-pairs on each side of the equation can be solved as P-n-P (perspective-n-points) problems. Some well-built methods like Direct Linear Transform (DLT) and Bundle Adjustment (BA) provide robust solutions for this problem.

For target-based methods, some known objects in the field of view are crucial for building connections between points in 3D and pixels in 2D. Checkerboards are a type of ordinary objects that are widely applied in the calibration process. In the engineering field, the intrinsic matrix of a camera is also unknown for users, while a checkerboard is necessary for intrinsic calibration, and it is convenient to extend its usage to both intrinsic and extrinsic calibration tasks. The matching points are the corner points of the checkerboard. Some methods use the geometry of the checkerboard to estimate the pose for sensors as well, for example, the edges and the normal vector of the board itself. Note that for LiDAR, patterns on the board are no longer clearly and precisely visible. Therefore, the shape of the object is not particularly determined as squares. As long as the features are easy to acquire for both camera and LiDAR, the choice of object is trivial. In experimental environment settings for extrinsic calibration tasks, the objects are usually put in the middle of the air, with little interfere from the ambiance. It is

ideal for calibration and the objects are easy to detect. In figure (3.8), an example of the target-based experimental environment is illustrated.

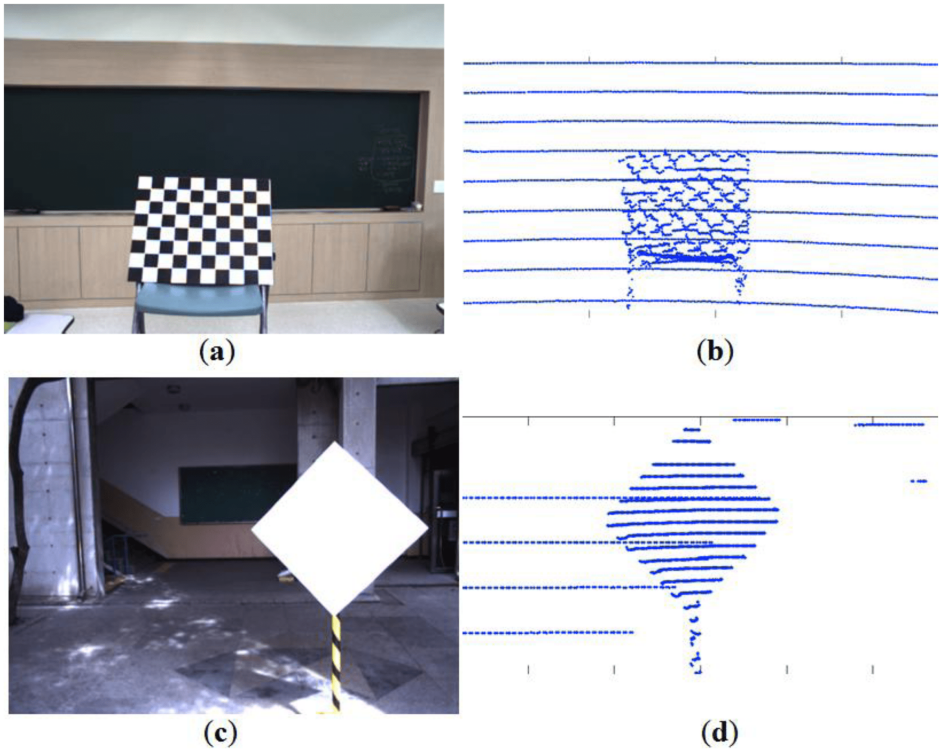


Figure 3.8: Velodyne scanning on both checker board and monochromatic board [33].

We can observe that except for the board, the remaining environment are noiseless, especially for LiDAR measurements. This artificial environment is hardly seen in real-world scenarios. Therefore most of the methods for target-based calibration are infeasible to apply to target-less scenarios. or targetless calibration methods, a general requirement is to extract features in natural scenes without any preset objects. It is a challenging topic since the modalities of camera and LiDAR are different, and the natural scenes are dynamic and arbitrary. In this case, the ideal experimental environments in target-based methods no longer exist.

3.4 Conclusion

This section has discussed the two major sensor models: intrinsic camera model and LiDAR model, respectively. Along with the mathematical derivation, we have explored the different modalities of the data model of these two types of sensors. We have also specified the particular problem for the extrinsic calibration of cameras, and LiDARs, including the 6-DOF rigid transform parameters of $SE(3)$ Lie group in three-dimensional space. The essential part is to correctly register the laser points and the corresponding camera image pixels. The discussion of the different experimental environments of traditional target-based methods and targetless methods is made.

In the next chapter (chapter 4), some seminal works addressing the abovementioned challenges are implemented and tested with in-depth analysis. Different types of methods for solving the problem from different perspectives are also discussed.

State-of-the-art Methods: Implementation and Analysis

4

The essence of the target-less approach of solving the extrinsic calibration problems is matching the point cloud and the image given arbitrary inputs. The types of features exploited and the metric choices are crucial, which answer the questions "what to match" and "how to match". Many attempts have been made to solve this problem. Those methods can be roughly categorized as motion-based registration and feature-based registration. As mentioned earlier, motion-based methods are less accurate while more computational power is needed [31]. Therefore, in this chapter, our focus is on the feature-based approaches. Three seminal works on feature-based methods will be reviewed and explained [32] [25] [5] in section 4.1, 4.2 and 4.3, respectively, and their results will be fully reproduced and analyzed.

4.1 Maximizing Dependence: A Mutual Information Approach

Mutual information (MI) is a metric that measures the statistical dependence between two random variables. In the multi-modal image registration field, mutual information has been widely applied since 1997 as a dependence indicator [28]. The assumption can be briefly explained as data acquired by multiple sensors on the same object should have a correlation. In 2012, Pandey et al. borrowed the idea of mutual information from image registration and formulated a mutual information-based extrinsic calibration pipeline for camera and LiDAR joint calibration scenario [32]. The MI assumption states that the reflectivity values of LiDAR measurements and its corresponding intensity values of the camera image should have a stronger correlation when the calibration parameters are correct. In this section, we will explain and analyze the approach mathematically and evaluate the results.

4.1.1 Preliminary Knowledge on Mutual Information

Before starting with the MI formulation for the calibration problem, it is necessary to explain the basic knowledge of mutual information. The mutual information is calculated by the sum of the entropy of both random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ substituted by their joint entropy. The term entropy, also known as Shannon entropy, is the measure of the uncertainty possessed in a random variable. Below we present the mathematical definitions in 4.1 4.2 and 4.3.

Assume $\boldsymbol{X}$ and $\boldsymbol{Y}$ are random variables. The entropy $H(\boldsymbol{X})$ is defined as

$$H(\boldsymbol{X}) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \tag{4.1}$$

The joint entropy of a pair of random variables is similar to the single variable form,

which is

$$H(\boldsymbol{X}, \boldsymbol{Y}) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y) \ , \tag{4.2}$$

where $p(x)$ is the marginal probability density of $\boldsymbol{X}$ and $p(x, y)$ is the joint probability density function. Mutual information is defined mathematically as:

$$\mathrm{MI}(\boldsymbol{X}, \boldsymbol{Y}) = H(\boldsymbol{X}) + H(\boldsymbol{Y}) - H(\boldsymbol{X}, \boldsymbol{Y}) \tag{4.3}$$

In this way, mutual information can also be interpreted as the amount of information that one possesses contains another. The larger the mutual information is, the stronger the bond is between the two random variables.

4.1.2 Mutual Information for Joint Calibration Problem

The assumption of exploiting the statistical correlation between the intensity of camera images and LiDAR reflectivities is based on some observations. For example, vegetations tend to provide higher reflectivity values while most of them have a green color. It is also true that white objects often reflect more than dark objects since black materials absorb more energy from the light. The deep-down theory of these types of phenomena is physics-related. Therefore, correlation-based methods, like the mutual information formulation, are more or less based on the experiences and physics that we just mentioned.

In order to build the connection between the intensity of the pixels on the image and the reflectivity values on the point cloud, it is necessary to firstly associate these two types of modalities by projecting the point cloud in 3D onto the image plane in 2D. Suppose the inputs are point cloud $\boldsymbol{\psi} \in \mathbb{R}^{N \times 3}$ and the input image $\mathbf{u} \in \mathbb{R}^{M_x, M_y}$. As explained in (3.17), given the calibration parameter, all the points can be projected onto the image plane given the calibration parameters $\boldsymbol{\psi}$. Suppose all the projected points from 3D to 2D as $\tilde{\boldsymbol{\psi}}$, we denote the corresponding intensity values of these points on 2D image as shown in (4.4). Therefore, each point contains two feature data, which are the reflectivity value $\boldsymbol{X}$ of LiDAR and the grayscale intensity value $\boldsymbol{Y}$ of the image.

$$\boldsymbol{Y} = \mathbf{u}(\tilde{\boldsymbol{\psi}}) \tag{4.4}$$

In practice, more than half of the projected points are physically located outside of the range of the image. These outliers are discarded due to the lack of corresponding pixels. Let the number of the rest of the points be n. The raw reflectivity values are scaled between 0 and 1, as the ratio of the intensity of the reflections. The reflectivity values are rescaled to range $[0, 255]$ and rounded as integers to assort with the intensity of the grayscale image. Let us denote the group of intensities and the scaled reflectivities as random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ in (4.5).

$$\begin{cases} \boldsymbol{X} = \{X_1, \cdots, X_n\} \\ \boldsymbol{Y} = \{Y_1, \cdots, Y_n\} \end{cases} \ , \tag{4.5}$$

It is difficult to directly obtain the actual probability of a particular intensity value. Given all the observations from $\boldsymbol{X}$ and $\boldsymbol{Y}$, a feasible approach is to calculate the fre-

quency of a particular intensity value as its probability.

$$P(X = k) = \frac{f_k}{n}, \tag{4.6}$$

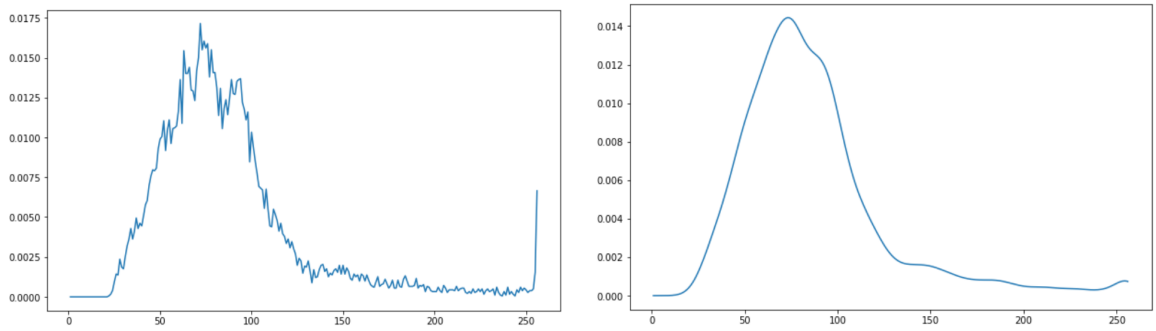
where the total number of pixels with intensity k is

$$f_k = \sum_{i=1}^{n} \delta\left(x_i - k\right), \quad k = 0, \cdots, 255 \tag{4.7}$$

In this way, the histograms for each random variable $\boldsymbol{X}$ and $\boldsymbol{Y}$ can be made. These types of estimations directly from the observations is statistically random and non-smooth. It is possible to reduce the estimation error of the probabilities by applying the KDE (Kernel Density Estimation) and smooth the step-like densities provided by the histogram. For multi-variable case, the KDE of the probability is calculated as

$$\hat{P}\left(X = k_x, Y = k_y\right) = \frac{1}{n} \sum_{i=1}^{n} K_\Omega \left(\begin{bmatrix} X \\ Y \end{bmatrix} - \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \right) \tag{4.8}$$

where the K_Ω is the 2D Gaussian kernel. To illustrate the improvement of KDE, we plot both histograms before and after the process of KDE.

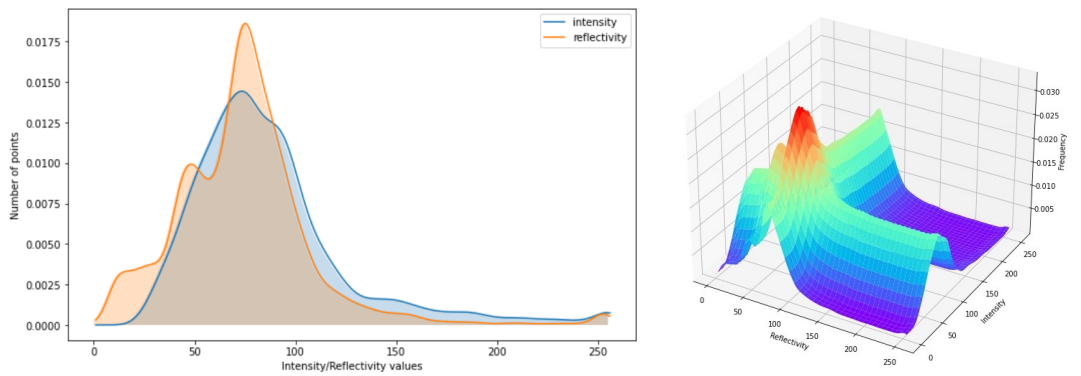


(a) The histogram of intensity values before KDE.

(b) The histogram of intensity values after KDE.

Figure 4.1: The comparison of histograms before KDE and after KDE.

To visualize the distributions both camera intensity and laser reflectivity, we present their probability density functions in figure 4.2. The left one simultaneously shows the shape of both distribution scaled from 0 to 255. The right one in 3D shows the joint distribution of both intensities and reflectivities after the kernel density estimation. The similarity between these two distributions ensures the assumption of mutual information formulation on solving the joint calibration problem.



(a) The histograms of both intensity and reflectivity values after KDE.

(b) An illustration of the KDE of the joint distribution.

Figure 4.2: The illustration of histograms in 2D and 3D.

With both the marginal and joint probabilities estimated, the mutual information parameterized by the extrinsic parameter $\boldsymbol{\theta}$ can be calculated as well. Therefore, the goal is to solve the maximization problem

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathrm{MI}(X, Y; \boldsymbol{\theta}) \tag{4.9}$$

since the assumption is when the calibration parameter is correct, the mutual information should be maximized. The parameter $\boldsymbol{\theta}$, as introduced in (3.10), is composed by the 6-DOF rotation and translation parameters.

Common choices including the gradient-based, Hessian-based optimizer, and heuristic methods can be selected in order to successfully optimize the target function (4.9). In our case, the choice is not particularly important as long as the objective function provides a smooth shape of the curve. To keep simplicity, we choose the Barzilai and Borwein steepest gradient ascent method [2].

The tricky part of using gradient ascent method is the gradient is hard to be derived through an analytical manner, since the histograms and the probabilities cannot be directly related to the calibration parameters in the form of functions. Therefore, the feasible approach is to use the numerical solution of the gradient.

$$\mathbf{G} = \nabla \mathrm{MI}(X, Y; \boldsymbol{\theta}) = \frac{\mathrm{MI}(\boldsymbol{\theta} + \Delta\mathbf{h}) - \mathrm{MI}(\boldsymbol{\theta} - \Delta\mathbf{h})}{2 \cdot \Delta\mathbf{h}} \tag{4.10}$$

The numerical derivatives can be calculated given a small parameter $\Delta\mathbf{h}$. Using Barzilai and Borwein's formulation [2], the update equation of the gradient ascent method is defined as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \gamma_k \frac{\mathbf{G}_k}{\|\mathbf{G}_k\|} \tag{4.11}$$

where k indicates the k-th step in the iteration. The adaptive step size is also defined as

$$\gamma_k = \frac{\boldsymbol{s}_k^\top \boldsymbol{s}_k}{\boldsymbol{s}_k^\top \boldsymbol{g}_k} \tag{4.12}$$

where $\boldsymbol{s}_k = \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$ and $\boldsymbol{g}_k = \mathbf{G}_k - \mathbf{G}_{k-1}$. The whole algorithm is presented in the algorithm 1.

Algorithm 1: The Mutual Information Formulation [32]

Input: image $\mathbf{u}$, point cloud $\boldsymbol{\psi}$ with reflectivity values $\boldsymbol{X}$, initial guess $\boldsymbol{\theta}_0$
Output: $\hat{\boldsymbol{\theta}}$

while $\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\| > \texttt{threshold}$ **do**
- Calculate projection matrix $\mathbf{T}(\boldsymbol{\theta})$ (3.14).
- Batch projection: $\tilde{\boldsymbol{\psi}} = \boldsymbol{K}\boldsymbol{T}\boldsymbol{\psi}$ (3.17).
- Corresponding reflectivity: $\boldsymbol{Y} = \mathbf{u}(\tilde{\boldsymbol{\psi}})$ (4.4).
- Histograms calculation: Hist($\boldsymbol{X}$), Hist($\boldsymbol{Y}$) and Hist($\boldsymbol{X}, \boldsymbol{Y}$) (4.6) (4.7).
- Perform KDE: $p(\boldsymbol{X})$, $p(\boldsymbol{Y})$ and $p(\boldsymbol{X}, \boldsymbol{Y})$ (4.8).
- Calculate $H(\boldsymbol{X})$, $H(\boldsymbol{Y})$ and $H(\boldsymbol{X}, \boldsymbol{Y})$ (4.1) and (4.2).
- Calculate entropy: MI$(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\theta})$ (4.3).
- Calculate $\mathbf{G} = \nabla \text{MI}(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{\theta})$ (4.10).
- Calculate the γ (4.12).
- Update: $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \gamma \frac{\mathbf{G}_k}{\|\mathbf{G}_k\|}$ as described in (4.11).

end

Therefore, through calculating the histograms and mutual information in each step within the optimization, the estimated parameter $\boldsymbol{\theta}$ is gradually updated until it is close to the maximum.

However, several drawbacks limit this algorithm. First of all, due to the non-smoothness of the 6-dimensional surface, the algorithm easily gets stuck at some local maxima. This can be observed through the 2D surface of rotational parameters r_x and r_y in Fig 4.3.

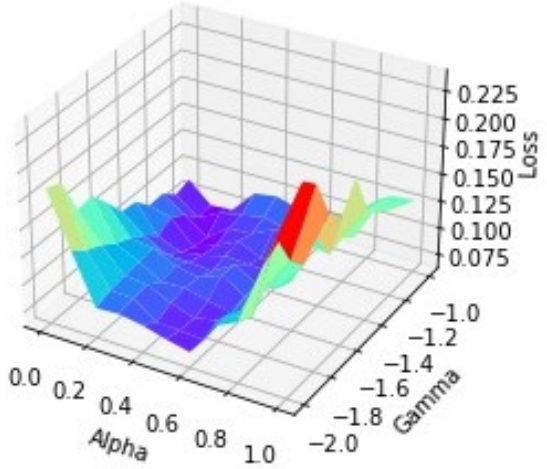


Figure 4.3: The surface of rotational parameters r_x and r_y. The ground truth is at the middle of the grid, where $r_x = 0.47$ and $r_y = -1.55$. The objective function has multiple local maxima which are not located at the ground truth. Therefore the algorithm will converge to the wrong point and provide a biased estimation.

This problem will lead to the biased estimation of the extrinsic parameters. In their real-field experiments, incorporating more than 40 frames of the data will reduce the bias to a relatively low level, while the calculation will consume more time, especially for the joint histogram with more than a million data points. Some differences between this experiment and the one from the original paper should be declared. Their system setup uses an omnidirectional camera, which consists of 5 cameras covering different directions. Therefore it covers the whole panoramic environment as LiDAR does, while in our experiment, we only apply one monocular camera from the KITTI dataset. The limitation of the amount of data and smaller field of view may also degrade the results of our implementation compared to the original one.

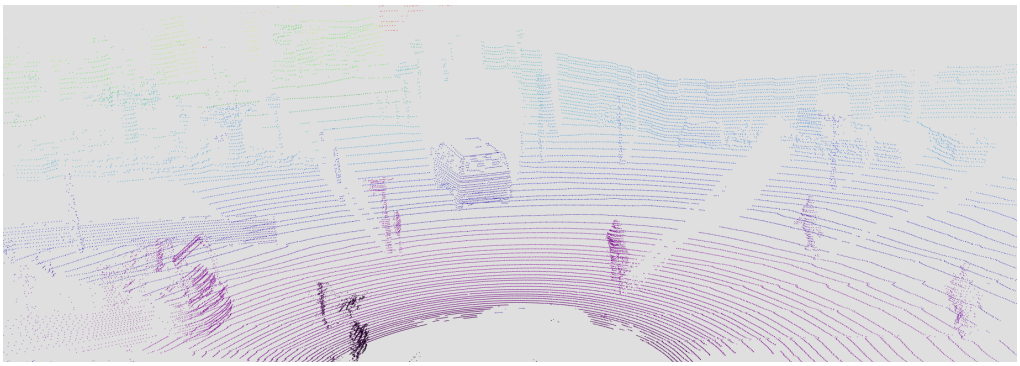
4.2 Feature Matching: An Edge Alignment Approach

Another seminal method of camera-LiDAR target-less calibration, which has an impact on a lot of later methods, is proposed by Levinson and Thrun in 2013 [25]. The critical insight of this method is to match the geometric features in both camera image and LiDAR point cloud, rather than exploiting the statistical correlation between pixel intensities and LiDAR reflectivities, as mentioned in the previous section. The underlying assumption of this method is intuitively convincing: The depth discontinuity should maximally match the edge of the image when the calibration parameters are correct.

Generally speaking, objects can be recognized through camera photos because there is a clear difference between the object and the surrounding environment, which is especially realized through the intensity difference of the pixels. Similarly, under normal circumstances, for the observer, the object's distance to the observer is also different from the surrounding environment. Therefore, the depth collected by LiDAR can also be exploited and distinguished to determine the object's boundary.

This edge alignment method exploits the common geometric features existing in both camera images and LiDAR point clouds. It also works when sometimes the reflectivity data from LiDAR are not reliable or even when some types of LiDAR that do not provide reflectivity values. In Fig 4.4, a comparison of both point cloud and camera image on an ordinary scene of a road scenario is presented.

(a) The camera grayscale image.



(b) The point cloud visualization on the same region.

Figure 4.4: The comparison of a 2D image and a 3D point cloud.

We can see that the cyclist in the middle right of the image has a distinct difference in terms of the intensity values. For the point cloud, there are also some visible but sparse boundaries of it with respect to the depth, which partitions the object and the environment. The cyclist in purple means it is closer to the LiDAR sensor, while the walls behind are in blue, representing farther distance. In Fig 4.4, there is a significant edge of the cyclist, which verifies the assumption. In order to explore further on edge alignment, we will elaborate this approach in a mathematical way in the following section.

4.2.1 Image Processing

In order to extract the edge of an image, we often consider the image as a discrete function with two variables x and y, indicating the vertical and horizontal coordinates. Each pixel carries the intensity value, which is also the value of the function. The changes in intensity values in small intervals are considered as edges. The goal of finding the edge is then finding the difference between pixels. In this method, a 3×3 maximum difference kernel is used. It calculates the absolute differences of intensities between itself and its 8 neighbors, and the maximum value is preserved as the intensity of the edge. Fig 4.5 illustrates this process of the middle pixel in a 3×3 grid.

Figure 4.5: The process of calculation the edge value of a pixel using maximal difference kernel. The pixel in the middle is the target pixel. After comparing the difference between pixels, the bottom right pixel is chosen, and the maximum difference is achieved and saved in the edge map.

After applying this kernel to every pixel in the image, an edge map $E \in \mathbb{R}^{M_x \times M_y}$ is obtained.

In order to encourage the laser points that are projected close to the edge of the image, it is also necessary to enlarge the edges on the image. Therefore they can cover more area and gradually "lead" the points to where the intensity of the edge reaches the peak.

In the original method, the spill-off for the edge is achieved by applying the IDT (inverse distance transform).

$$D_{ij} = \alpha \cdot E_{ij} + (1 - \alpha) \cdot \max_{x,y} \left(E_{x,y} \cdot \gamma^{\max(|x-i| \cdot (y-j|)} \right) \tag{4.13}$$

The new calculated image D has the same size as edge map E. Each pixel value $D_{i,j}$ is calculated by (4.13). The closer to the edge, the higher value assigned. This goal can also be accomplished with a similar result by using a Gaussian kernel to smooth and spread the edges [21]. Similar to the previously mentioned kernel calculation, it is the edge image E convolve with the Gaussian distribution weight matrix. As for the Gaussian distribution weight matrix, it is the product of sampling and normalizing the density function (that is, the Gaussian function) of the two-dimensional normal distribution. Comparison of D and E is shown in Fig 4.6.

(a) The edge map E calculated by maximum difference kernel.

(b) The map D derived from E.

Figure 4.6: The edge processing before and after edge spill-off using 35×35 Gaussian kernel.

4.2.2 Point Cloud Processing

The essential part of the depth discontinuity method is that the algorithm is able to extract the edge of the point cloud in a short period of time. Therefore the automatic online update of the calibration parameters becomes possible. Before extracting the edge, we should define what is the edge of a sparse 3D point cloud. The edge is in fact, distinguished by the depth difference. The depth value of a 3D point in autonomous driving sense is normally defined as

$$d_i = \sqrt{{x_i^l}^2 + {y_i^l}^2}. \tag{4.14}$$

Generally, the upward direction z is not considered within the calculation of depth due to the reason that the cylindrical surface is easier to analyze than the spherical surface. This is the case, especially when taking the bird's-eye-view, since the cylindrical surface and the corresponding depth will be converted directly into the circle and radius, respectively.

The edge extraction algorithm compares the depth value of a point between its neighbors. If the depth difference is larger than a threshold, the point is defined as a point with depth discontinuity. As mentioned in section 3.2, multi-beam LiDAR possess multiple laser emitters. Each of them has a different pitch angle. The edge extraction algorithm runs within a single beam of laser realizations. So the neighbor of a point is simply its leftward and adjacent rightward points. To fully process the whole point cloud, it runs beam by beam. The expression of the algorithm is presented in (4.15). The visualization of the depth discontinuity process of a single laser beam is illustrated

in figure 4.7.

$$c_i = [\max(d_{i-1} - d_i, d_{i+1} - d_i, 0)]^\gamma \tag{4.15}$$

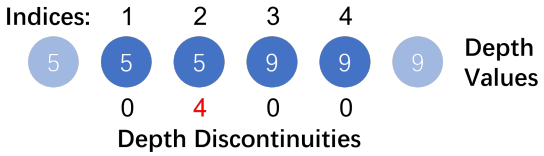


Figure 4.7: A simple illustration of depth discontinuity calculation of a set of laser realizations in a row. The first line labels the indices of points. The second line contains each point's depth values, and the last line shows the calculated depth discontinuity values. According to (4.15), point 2 has a positive depth discontinuity value. Intuitively, its neighbor point 3 is also an edge point, while according to the definition, it has a depth discontinuity value of $\max(-4, 0, 0)$, which is 0.

This can also be achieved by an one-dimensional 1×3 maximum difference kernel, which is similar to the 2D kernel for image processing in Fig 4.5. Note that the suffix $i-1$ and $i+1$ denotes the left and right neighbors of the i^{th} point. For each point, the corresponding depth difference is calculated. If we only select the points with values that are higher than the threshold, all the edge points are filtered out. The parameter γ is a factor to tune the curve of increasing depth values and affect the point selection. We stick to the value the same as the one in the original paper, which is $\gamma = 0.5$.

Figure 4.8: The results of edge point selection. The threshold is set as 2 meters.

The newly selected points are gathered as a new point cloud, denoted as X. It contains the edge points with their depth discontinuity value, which is only a small amount of points compared to the original point cloud. All the edge points are projected on the image edge map D, given the calibration parameter $\boldsymbol{\theta}$. Therefore the points now contain two features, the edge information of the image and their inherent depth discontinuity values. The form of the cost function is also direct. The cost function J for a single frame is defined as

$$J_\theta = \sum X_i \cdot D_i \tag{4.16}$$

which simply performs the multiplication between the two features. According to the assumption of edge alignment-based calibration, if the calibration parameter is correct, the cost function J should be maximized since, at that moment, most of the edge points

will also be located on the edges of the image, then the multiplication results should be larger than anywhere else.

The whole setting of the depth discontinuity formulation is for online calibration with some small drifts of parameters. Therefore the computational cost of the whole algorithm is minimized. This is the case even for optimizing the objective function. Since gradient-based optimization scheme is time-consuming, they use a 1-step grid search for the online alignment and only update when there is a higher value for cost function within the grid. Generally, for larger bias of the initial calibration parameter, there are also a lot of local maxima that grid search cannot avoid. This might be improved by also considering the horizontal edges since all the edge points are extracted by the algorithm that can only identify the vertical edges.

4.3 Hybrid: A Joint Calibration-Fusion Approach

A lot of existing methods focus on matching the extracted features, thus building up the connection between the camera measured images and the point clouds acquired from the LiDAR system. For target-based calibration techniques, estimating the extrinsic calibration parameters is usually considered an independent task from sensor fusion. For example, the target used for calibration, a checkerboard, does not contribute to the further fusion procedure. This idea also applies to the majority of current target-less calibration cases. Nevertheless, calibration can also be a part of sensor fusion, significantly when the fusion task contributes to the calibration results. In 2016, Castorena et al. provide a fresh idea of a calibration algorithm based on edge alignment, achieved by solving a joint calibration and fusion problem [5].

In this section, we will focus on the joint problem of both calibration and sensor fusion. To be specific, the fusion task in this work is to construct a depth channel of the intensity image, given the depth measurement from LiDAR. This fusion approach is also considered as depth super-resolution or depth re-construction since the depth measurement from the point cloud is sparse. All the raw LiDAR points are projected directly onto the image plane, and initialize the sparse depth map. This raw data projection is considered as data-level fusion or low-level fusion. While building up the dense depth map, the calibration parameters are also calculated iteratively by matching the edges between the intensity image and depth map. The critical insight of this method is that the fusion and calibration procedures are complementary processes. On the one hand, the depth super-resolution provides the depth discontinuity information of LiDAR point cloud, which is helpful for edge alignment-based calibration. On the other hand, the calibration provides a better match between the depth map and the intensity image, thus contributes to the correct construction of the resulting depth image. The end of this process will generate an RGBD image (RGB color image with depth channel) as the result of the fusion and a set of finely calibrated extrinsic parameters as the result of calibration. Some introduced parameters and the projection equations from chapter 3.3.2 will be used again in the following content in order to mathematically describe the method,

4.3.1 3D-2D Projection

Assume the desired $SE(3)$ group transform matrix is $\boldsymbol{T} \in \mathbb{R}^{4\times4}$, which contains the rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and translation vector $\boldsymbol{t} \in \mathbb{R}^{3\times1}$. As derived before, the rotation matrix can also be expressed by the rotation angles r_x, r_y and r_z, shown in (3.11). Similarly, the translation vector is composed by translation distances t_x, t_y and t_z. The set of extrinsic calibration parameters can be defined as a vector $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = (r_x, r_y, r_z, t_x, t_y, t_z) \tag{4.17}$$

Suppose the input point cloud from a single frame of LiDAR is $\boldsymbol{\psi} \in \mathbb{R}^{N\times3}$, with the Cartesian coordinates (x_i^l, y_i^l, z_i^l) for i_{th} point. The reflectivity values are excluded and not considered in this approach. The total number of points is N. Given an extrinsic calibration parameter $\boldsymbol{\theta}$, the points in LiDAR coordinate system $\mathcal{C}_l$ can be projected onto the image coordinate system, using the extrinsic model. The process is displaced in (4.18)

$$\mathcal{P}_{\boldsymbol{\theta}} = \left(\boldsymbol{K}\cdot\boldsymbol{T}\cdot\begin{bmatrix}\boldsymbol{\psi}^\top \\ \mathbf{1}^\top\end{bmatrix}\right)^\top = \left(\boldsymbol{K}\cdot\boldsymbol{T}\cdot\begin{bmatrix} x_1^l & x_2^d & \cdots & x_N^l \\ y_1^l & y_2^l & \cdots & y_N^l \\ z_1^l & z_2^l & \cdots & z_N^l \\ 1 & 1 & \cdots & 1\end{bmatrix}\right)^\top, \tag{4.18}$$

where $\boldsymbol{K} \in \mathbb{R}^{3\times4}$ is the inherit intrinsic matrix of the camera model. The resulting $\mathcal{P}_{\boldsymbol{\theta}}$ is the set of projected points, saved in a $\mathbb{R}^{N\times3}$ matrix, in which the first and second columns are the coordinates in the digital image, while the third column contains the homogeneous coordinate.

If given the size of the input image $\mathbf{u} \in \mathbb{R}^{M_x\times M_y}$, the projected points which are located outside of the image range can be filtered out. The remaining points are re-allocated as pixels into a new empty image with the same size of the input image $\mathbf{u}$, given their coordinates in the pixel coordinate system $\mathcal{C}_p$. The newly generated image is denoted as $\tilde{\mathcal{P}}_{\boldsymbol{\theta}} \in \mathbb{R}^{M_x\times M_y}$. The values filled for each allocated point is the depth information d_i, where the depth information is defined through the measurement of the forward and leftward distance shown in (4.14). Therefore, the $\tilde{\mathcal{P}}_{\boldsymbol{\theta}}$ is a sparse image containing the depth information from LiDAR measurements.

4.3.2 Depth Super-resolution

In this subsection, we will present the depth super-resolution method from [5]. Normally, pixels are densely placed within an image. Each one of them possesses the intensity information. The $\tilde{\mathcal{P}}_{\boldsymbol{\theta}}$ created above is not the case. Only some 3D points are lucky enough to be projected inside the range of the image. The rest of them are empty, carrying no information. The resulting $\tilde{\mathcal{P}}_{\boldsymbol{\theta}}$ is a sparse image of depth values, highly sub-sampled comparing to the dense image we want to recover.

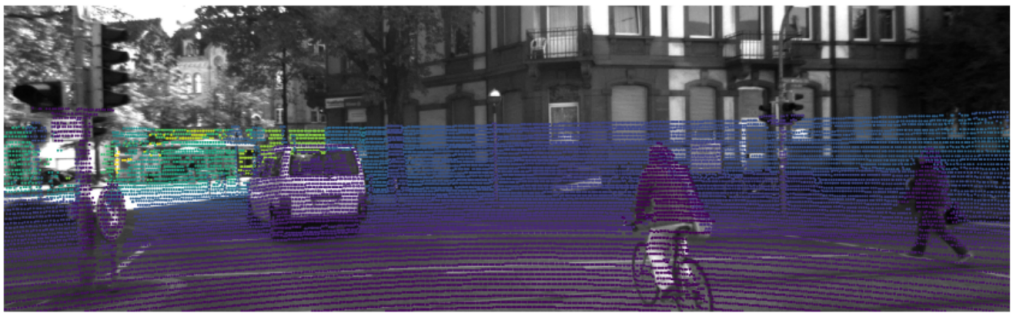
To validate the data fidelity, we set the constraint as the ℓ_2 norm of the difference between the projected data with measured depth and the desired depth map $\boldsymbol{\phi}_{\boldsymbol{\theta}}$ to be recovered. The cost function also includes the total variation norm of the depth map $\boldsymbol{\phi}$,

which is the squared sum of the gradient values at every pixel in the desired depth map. The cost function to minimize is shown in (4.19).

$$J_{\boldsymbol{\theta}}(\boldsymbol{\phi}) = \frac{1}{2}\left\|\tilde{\mathcal{P}}_{\boldsymbol{\theta}} - \mathbf{H}\odot\boldsymbol{\phi}\right\|_{\ell_2}^2 + \lambda\sum_{x=1}^{M_x}\sum_{y=1}^{M_y} w_{xy}\left\|[\boldsymbol{\nabla}\boldsymbol{\phi}]_{xy}\right\|_{\ell_2} \tag{4.19}$$

The operator $\odot(\cdot)$ denotes the element-wise matrix multiplication. $\mathbf{H}$ is a sparse binary masking matrix, with value 1 only at the location where points are projected, and the rest spots remain 0. Therefore the term $\left\|\tilde{\mathcal{P}}_{\boldsymbol{\theta}} - \mathbf{H}\odot\boldsymbol{\phi}\right\|_{\ell_2}^2$ indicates that the difference between masked dense depth map and the original sparse measurements. Minimizing this term promises the fidelity of the recovered dense map, which means these pixels at the designated locations of the new dense map should carry the same depth values compared with the original sparse depth map derived from the raw measurements. The weighted isotropic total variation promotes the sharpness at edges while preserving the smoothness elsewhere. λ is a positive regularization parameter, influencing the smoothness of the resulting depth map effectively. The dense depth map to be recovered from the LiDAR projections given calibration paramter $\boldsymbol{\theta}$ then can be estimated by minimizing the cost function J, denoted as shown in (4.20). This ℓ_2 least squares with total variation norm as regularization term can be optimized by algorithms like FISTA [3], or split-Bregman [18].

$$\hat{\boldsymbol{\phi}}_{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\phi}\in\boldsymbol{\Phi}} J_{\boldsymbol{\theta}}(\boldsymbol{\phi}) \tag{4.20}$$



(a) The projected points on the grayscale image.

(b) The depth completion by solving (4.20).

Figure 4.9: The results of depth completion.

In Fig 4.9 we present the depth completion results using FISTA algorithm, using the solver `PyUNLocBoX` [23]. We can observe that in the lower region with sparse depth pixels, the depth completion performs well, but for the upper area with no depth realization, there is a distinct boundary. Due to the occlusion, the right part of the cyclist is corrupted since the foreground points are mixed with background points. In general, the regularized optimization achieves the goal to create a dense depth map from sparse measurements.

4.3.3 Joint Calibration

In the joint calibration-fusion pipeline [5], after successfully reconstructing the dense depth map, the calibration problem of LiDAR and camera is reformulated as matching the edges of the depth map and intensity image. This is actually a variant of the image registration problem. An intuitive way to find the similarity between two images is to align the edges of the images. Another reason for edge alignment is the perceptual features possessed in the depth map are very few, except for the depth gaps, which are the edges of the regions with distinct depth differences.

In order to diminish the error raised during the depth completion step, only the points with LiDAR realizations will be considered for gradient calculation. The way to punish the edge misalignment is using the multiplication of the gradient of the depth map and the modified gradient of intensity image. Results at all selected points are summed and normalized in order to offset the impact of the numbers of selected points. The objective function is shown in (4.21).

$$\mathcal{F}(\boldsymbol{\theta}) = \sum_{k\in\{x,y\}} \frac{\sum_{n\in\boldsymbol{\Omega}_{\boldsymbol{\theta}}} w_{n,k} \cdot |[\boldsymbol{\nabla}_k \boldsymbol{\phi}_{\boldsymbol{\theta}}]_n|}{\left(\sum_{n\in\boldsymbol{\Omega}_{\boldsymbol{\theta}}} w_{n,k}\right) \cdot \left(\sum_{n\in\boldsymbol{\Omega}_{\boldsymbol{\theta}}} |[\boldsymbol{\nabla}_k \boldsymbol{\phi}_{\boldsymbol{\theta}}]_n|\right)} \tag{4.21}$$

In the equation above, $\mathcal{F}(\boldsymbol{\theta})$ is our target function to minimize. The point set $\Omega_{\boldsymbol{\theta}}$ represents all the points that are projected on the image plane and within the image range. The parameter k is a direction indicator for gradient $\boldsymbol{\nabla}_k$, which can be horizontal gradient $\boldsymbol{\nabla}_x$ or gradient of vertical direction $\boldsymbol{\nabla}_y$ for images. The parameter $w_{n,k} = \exp\left(-\gamma\,|[\boldsymbol{\nabla}_k \mathbf{u}]_n|\right)$ is the weight for the gradient of intensity image. It includes the same directional gradient at the same pixel but for intensity image. The terms in the denominator are the multiplication of summed weights and depth gradients, which is a necessary normalization factor that takes into account the difference in number of points in $\Omega_{\boldsymbol{\theta}}$ that are included in the evaluation given different parameter sets $\boldsymbol{\theta}$ [5].

A proper choice of γ is crucial for a reasonable weight. The weight w is to tune the effect of the intensity image. Assume it is an edge in the camera image. If γ is large, the weight is extremely close to 0. Then, it will not punish the gradient of the depth map since the difference between edge and non-edge is very small after multiplying with weight. Similarly, a small γ reduces the effect of the intensity image. Thus it will penalize both the edge or non-edge of the depth map similarly. Fig 4.10 shows the values of weight given the intensity values from 0 to 255.

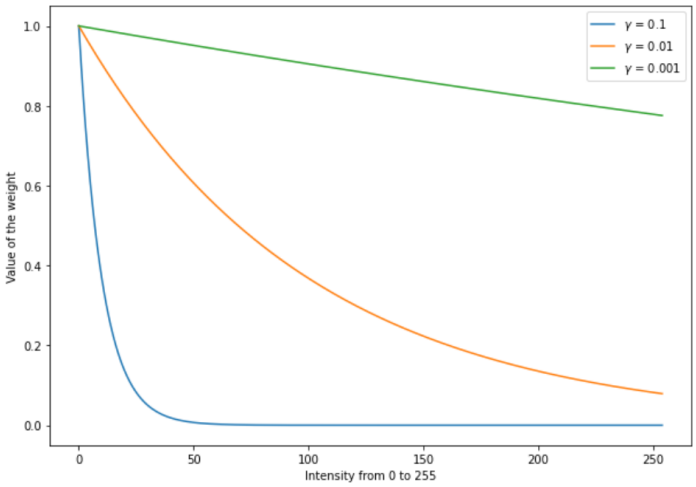


Figure 4.10: The influence of γ on the value of the weight.

The target function $\mathcal{F}(\theta)$ to be minimized is the weighted sum of the gradient of a small group of points in the depth map, where the group of points is actually the projection of LiDAR measurements given θ. The reason for constraining the projected points, rather than using all the points, is that only the projected depth is promised to be the real measurements, while other depth information is estimated by solving the TV norm [39]. $\mathcal{F}(\theta)$ can be expressed in vector form, if we stack the weight values and depth map gradients into a vector. Then the target function can be rewritten as

$$\mathcal{F}(\boldsymbol{\theta}) = \sum_{k \in \{x,y\}} \frac{\boldsymbol{w}_k^\top \boldsymbol{g}_k}{\|\boldsymbol{w}_k\|_{\ell_1} \|\boldsymbol{g}_k\|_{\ell_1}} \quad , \tag{4.22}$$

where each element in $\boldsymbol{g}_k$ is the absolute gradient value of depth map, expressed as $|[\boldsymbol{\nabla}_k \boldsymbol{\phi}_{\boldsymbol{\theta}}]_n|$. If we temporarily overlook the sum, the rest is similar to the expression of the cosine angle between the two vectors $\boldsymbol{w}_k$ and $\boldsymbol{g}_k$.

Recall the simple expression of the angle between two vectors:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} \tag{4.23}$$

Note that the only difference in our target function and this expression is the norm. In the cost function, the norm is ℓ_1 while here it is the square root of ℓ_2.

The angle between 2 vectors can also be the geometric interpretation of the Pearson correlation coefficient. If we look at the sample reflective correlation (or cosine similarity), which is a variant of Pearson correlation, our target function $\mathcal{F}(\boldsymbol{\theta})$ can be roughly considered as the correlation between the weight $\boldsymbol{w}$ (**negatively** related to gradient of camera image) and vector $\boldsymbol{g}$ (depth map gradient). Therefore, in order to maximize the correlation between the gradient of intensity image and depth image, the target function $\mathcal{F}(\boldsymbol{\theta})$ should be minimized.

4.3.4 Simulated Annealing Method

The procedure of solving the target function $\mathcal{F}(\boldsymbol{\theta})$ is a straightforward heuristic optimization algorithm. Considering the difficulty of calculating the numerical gradient of $\mathcal{F}(\boldsymbol{\theta})$, using annealing simulation is more random since it is not always going forward to the local minimum. But this is also a benefit in another way since it preserves the possibility of jumping out of the local valley and keeping searching.

The procedure of the simulated annealing algorithm is simple. It starts with an initial point, randomly selects a neighbor, and calculates the value of the target function. If the value at the neighbor is smaller than the value at the previous point, proceed update of the current position to the neighbor. The key idea of this algorithm is to set a possibility of jumping out of the local optima. This is realized by adding randomness into the algorithm. If a certain condition is reached, the algorithm performs the update even if the value of the target function is higher than before. The detailed algorithm for the mutual information formulation is presented below.

Algorithm 2: The joint calibration-fusion algorithm [5]

Input: Image $\mathbf{u}$, point cloud $\boldsymbol{\psi}$, initial guess $\boldsymbol{\theta}_0$ and starting temperature T_0.
Output: The depth map $\hat{\boldsymbol{\phi}}$ and the estimated calibration parameter $\hat{\boldsymbol{\theta}}$.

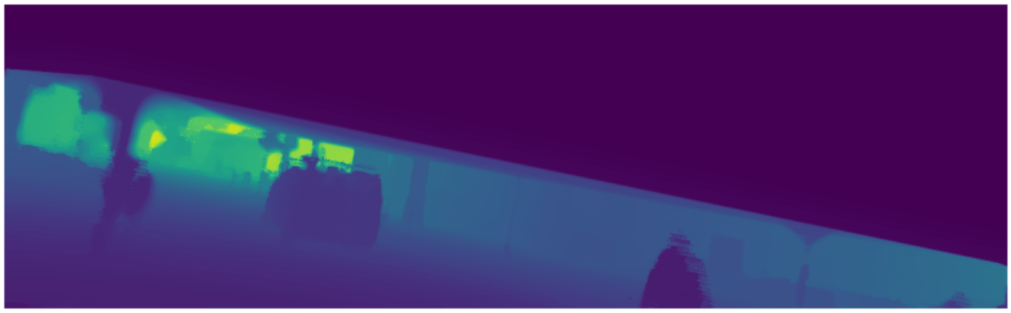
while $T > T_{threshold}$ **do**
  Calculate $\mathcal{P}_\theta\{\boldsymbol{\psi}\}$ through point cloud projection given $\boldsymbol{\theta}^t$ based on (4.18).
  Estimate $\boldsymbol{\phi}_\theta^t$ by depth completion (4.20).
  Calculate $\mathcal{F}(\boldsymbol{\theta}^t)$ according to (4.21).
  $\Delta T = \mathcal{F}(\boldsymbol{\theta}^t) - \mathcal{F}(\boldsymbol{\theta}^{t-1})$
  if $\Delta T < 0$ **then**
    Update:
      $\hat{\boldsymbol{\phi}} = \boldsymbol{\phi}_\theta^t$. $\quad \hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^t$. $\quad T = \alpha T$, where constant parameter $\alpha \in (0, 1)$.
  else
    if $\exp -\frac{\Delta T}{T} > rand\ (0, 1)$ **then**
      Update:
        $\hat{\boldsymbol{\phi}} = \boldsymbol{\phi}_\theta^t$. $\quad \hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^t$
    else
      continue.
    end
  end
end

The important term $\exp -\frac{\Delta T}{T} > \text{rand}\ (0, 1)$ illustrates that the algorithm has the possibility to update even if the current value of function is not better. When T gradually decreases, the possibility of jumping out is reduced either. After a long iteration, it will gradually stop at local optimal. The results are therefore the estimated calibration parameter $\hat{\boldsymbol{\theta}}$ and the depth map $\hat{\boldsymbol{\phi}}$.

By examining the whole pipeline, we can see that in the joint calibration-fusion formulation, the purpose of fusing the dense depth map is to extract the gradient information of the point cloud because the edge of the point cloud in 3D is converted to the gradient of the depth map in 2D. Therefore, in some sense, it improves the feature selection of

the depth discontinuity method since the gradient of the image covers both vertical and horizontal edges.

Although the idea of jointly solving both calibration and fusion problems is novel, the drawback of the pipeline also limits its performance. Generally, the whole optimization procedure is slow. One of the most important factors of its inefficiency is that the data processing steps are inside the loop of the simulated annealing-based optimization. Therefore, the FISTA or split-Bregman optimization algorithms have to run repetitively, while most of their calculation is redundant. This can be verified by showing the depth map given different calibration parameters.



(a) The depth map given incorrect calibration parameters.

(b) The depth map given correct calibration parameters.

Figure 4.11: The comparison of dense depth maps between different calibration parameters.

In Fig 4.11, there are overlapping regions in both maps, such as the cyclist and the vehicle in the middle. More importantly, their topology is static and will not be affected even if the calibration parameter changes. For example, the distant pillar on the right of the car presents almost the same relative position to the vehicle, although the above image is twisted. This can also be recognized since the input point cloud is static, which will also lead to a static depth map, although due to the change of calibration parameters, some information will lose. Therefore, in hundreds of optimization steps, the majority of the calculation time and power is wasted on performing the depth completion on an almost static depth image with respect to the content. The problem of the optimization scheme is also influencing the speed. A lot of attempts that will not lead to the decreasing of the temperature are superfluous, while they also perform the whole calculation in the loop.

4.4 Conclusion

In this chapter, we have closely examined three influential methods (Mutual Information formulation [32], depth discontinuity method [25] and joint calibration-fusion pipeline [5]) for target-less extrinsic calibration problem. Each of them uses different types of features to formulate the problem. To summarize, we sketch a table to show the features used in those methods.

Methods	**Depth**	**Reflectivity**	**3D Points**	**Multi-frame**	**Type**
Mutual information [32]	✗	✓	All points	✓	Statistical correlation
Depth discontinuity [25]	✓(horizontally)	✗	Edge points only	✓	Edge alignment
Joint calibration-fusion [5]	✓	✗	All points (weighted)	✗	Edge alignment

Table 4.1: Comparison of three methods.

The mutual information formulation builds up the statistical correlation between the reflectivity values and the camera intensity images. The assumption of this method is based on the natural correlation between color and reflectivity. Therefore, a mutual information metric is developed between laser reflectivity and corresponding camera intensity to measure the calibration process. In order to achieve the optimal calibration parameters, the mutual information should be maximized. Statistically, it holds, but there are some drawbacks to this method. Firstly, it is heavily influenced by the shadows and shadings that exist in camera images. Therefore the assumption no longer holds. Secondly, by incorporating more than 20 to 40 frames, the shape of the solution space becomes smooth. It is an exhaustive calculation process and takes a lot of time, especially since all the calculations are inside the optimization loop. But when only several frames are available, the objective function is not smooth. Therefore the gradient ascent method is often stuck at local optima, resulting in heavily biased outputs.

The depth discontinuity method exploits the 3D information provided as a feature. It extracts the edge by examining the left and right neighbor points within a single beam. On the image side, a simple edge extraction kernel is used to quickly select all the edges. All the laser edge points can then be projected onto the edge image for edge alignment. The objective function is the sum of multiplications of the intensity of depth discontinuity of the points and the corresponding edge intensity of the image edge map. As an online method, it aims at fixing slight drift in a relatively small interval. Therefore local optima still exist. Also, it only takes horizontal edges into account. There are possibilities for improvements.

The joint calibration-fusion method addresses the calibration problem by actively performing the depth map completion as a sensor fusion step. All the LiDAR points are projected onto the image plane as a sparse image with depth values. Then given the

sparse image, a depth completion is performed to acquire a dense map. Then, the edge of the depth map and the edge of the intensity map are matched through a designed cost function based on a variant of cosine similarity. In each iteration, the depth completion is firstly performed as the sensor fusion step, given all the projected points. Then, the result will contribute to the calibration optimization. The updated calibration parameters will, later on, guide another round of points projection and fusion. This two-stage process is therefore called the joint calibration-fusion method. The novel idea is that it converts the 3D edge information into 2D as an image. But, the depth completion is re-calculated during each iteration, even though the major area to be recovered is static. The massive calculation repetition drastically reduces the speed of the algorithm. Its cost function is not smooth, therefore the gradient-free heuristic simulated annealing method is adopted. It preserves the ability to jump out of local optima while taking more calculation time to keep trying possible directions in a 6D solution space. Also, when few points are projected onto the image, the FISTA algorithm may provide non-reliable results, leading to exceptional errors to the whole pipeline.

Among all three methods, several drawbacks exist that influence the overall performance. We can also observe that no one uses the edge of reflectivity values as a descriptor to find edges. Two of them bring massive computational burdens to the optimization loops. Further ideas and improvements on these methods lead to our new MulFEA pipeline that is able to address the drawbacks mentioned above, which will be explained thoroughly in the next chapter.

The Proposed Multi-Feature Edge Alignment method: MulFEA

5

5.1 Overview of the MulFEA Pipeline

5.1.1 MulFEA Workflow

In previous chapter, we introduced several state-of-the-art methods on the targetless extrinsic calibration problem. Although their innovations and assumption are sophisticated, some drawbacks can also be observed during our own implementations and the analysis. In this section, we propose a new workflow called MulFEA (Multi-Feature Edge Alignment) that addresses some problems of previous works. The flow chart of the pipeline is shown in Fig 5.1.

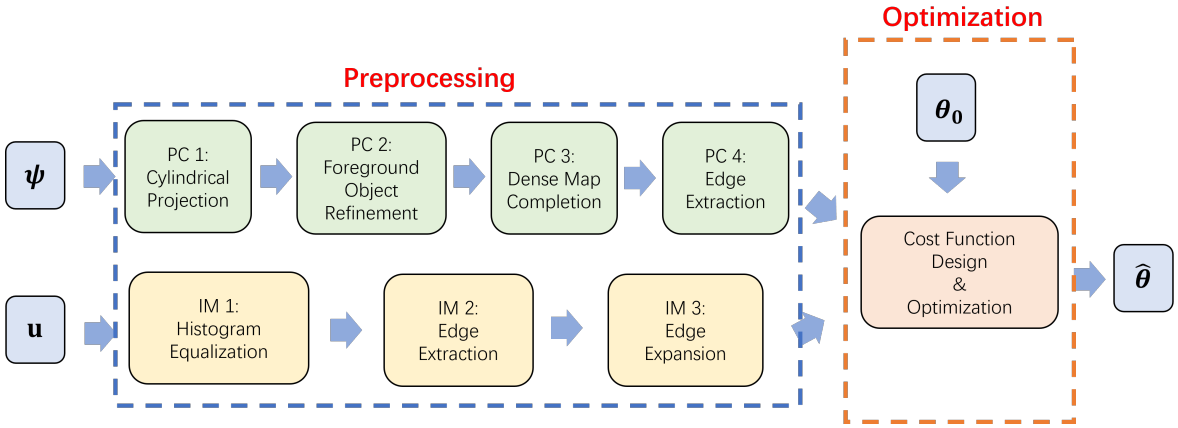


Figure 5.1: The proposed workflow. The blocks in blue at the left of the graph are the input image $\mathbf{u}$ and the point cloud $\boldsymbol{\psi}$ to be processed. The green boxes are the preprocessing and feature extraction steps for the point cloud. The yellow blocks are the image processing steps. The orange block is the optimization procedure, which runs after the data processing. The final blue block on the right is the output of the optimization process, which is the estimated extrinsic calibration parameters.

The input of MulFEA is the image $\mathbf{u}$ and the point cloud $\boldsymbol{\psi}$. To initialize the optimization, an initial guess of the calibration parameter $\boldsymbol{\theta}$ is also required. The pipeline outputs the estimated calibration parameter $\hat{\boldsymbol{\theta}}$. We use the blue dashed box to highlight the preprocessing step of input data and the yellow box for the optimization step. More specifically, the green-colored blocks are for point cloud processing, and the yellow ones are image processing steps. There are three inputs at the beginning to initialize the pipeline, which is the point cloud $\boldsymbol{\psi}$, the input image $\mathbf{u}$ and the initial guess of the calibration parameter $\boldsymbol{\theta}$. Only camera images and the LiDAR point cloud are processed through the preprocessing step. The initial guess of $\boldsymbol{\theta}$ will go directly to initialize the

gradient-based optimization step. The final output is the estimated calibration parameter $\hat{\boldsymbol{\theta}}$.

We concentrate on the point cloud processing in preprocessing steps since the 3D sparse points processing is non-trivial. Our proposed workflow for point cloud processing, shown in green on the upper side of the processing pipeline, includes cylindrical projection, foreground object refinement, dense map completion, and edge extraction. In the image processing branch, three steps are shown in yellow: histogram equalization, edge extraction, and edge expansion. Finding the edge of the 3D point cloud is more challenging than finding the edge of the camera image. Therefore, more complicated steps are required, and some techniques for robust features are applied. After extracting the reliable features, the upcoming procedure is to construct a cost function that incorporates all the features from both camera image and LiDAR point cloud. It should provide the best response when the extrinsic calibration parameters are correct. Similar to other edge-alignment methods, our cost function should also exploit the geometric similarities between the edge of the camera image and the edge of the point cloud. After that, some gradient-based optimization approaches will be used and provide the estimation results of the calibration parameter.

5.1.2 Workflow-level Improvements

One of the essential advantages of MulFEA pipeline is that all data preprocessing steps are performed before doing optimization in the iterations. In other words, all features that participate in the optimization are calculated beforehand rather than being calculated within iterations. This separation scheme, therefore, saves enormous computational power.

Although this repeated calculation problem is not difficult to think of, it is still overlooked in a lot of existing works, such as [5] and [48]. For example, in [5], each time when the calibration parameters are updated, the point cloud will be projected onto the image again. However, most of the points are the same compared to the previous iteration. The depth super-resolution is also performed similarly, although most areas are calculated and recovered by the FISTA algorithm in the earlier iteration. Repetitively running the depth super-resolution algorithm is the primary time-consuming process in his pipeline.

There are also some other improvements and innovations in this pipeline. For example, the cylindrical projection combined with dense map completion enables the feature extraction of the point cloud without any interaction with camera images. The foreground object refinement steps can strengthen the features of the fore. To enrich our edge map, we adopt multiple features, which are depth discontinuity, reflectivity, and object values. The details will be explained in the following sub-sections.

5.1.3 Outline

In the following sections, we will firstly introduce the point cloud processing in section 5.2, according to the sequence from PC 1 to 4, which are PC 1: cylindrical projection (chapter 5.2.1), PC 2: foreground object refinement (chapter 5.2.2), PC 3: dense map completion (chapter 5.2.3), and PC 4: edge extraction (chapter 5.2.4). After that,

we will explain our pipeline on image processing in yellow in section 5.3, including IM 1: histogram equalization (chapter 5.3.1), IM 2: edge extraction and IM 3: edge expansion (chapter 5.3.2). At the end, we will present our designed cost function and the optimization process (chapter 5.4).

5.2 Point Cloud Processing

The point cloud, as compared to the digital image, is hard to process. The distribution of pixels is discrete and tightly placed as a rectangle with known size, while the points are scattered in 3D space with continuous values. To track the pixel on an image, a single set of 2D coordinates will be enough, which illustrates the location of the pixel. The point cloud is in a completely different form. Usually, indices are used for the point selection, while no information can be derived through the index of a point. The cloud itself is shapeless, which means the points can be anywhere inside the 3D space. Some tasks, such as 3D reconstruction and modeling, will use voxels to construct a 3D version pixel-image, while it takes much computational power to build a 3D voxel space densely. Using a voxel grid will also down-sample the data, which is mainly used for in-door scenarios with high-resolution point cloud. In outdoor cases, points are already separated due to the more extensive range. If performing further the down-sampling scheme, some of the features will be lost after the processing.

In our case, we apply the pass filter, and the FOV (Field Of View) filter to approximately select the region of interest and filter out the rest of the points. The remaining points will move on to the next pre-processing steps.

As introduced in section 3.2, the LiDAR system scans the environment by spinning. All the points are collected with 3D coordinates (x_l, y_l, z_l). The pass filter is to select all the points within the pre-defined thresholds or 3D space. Imagine adding a huge b-box in the 3D space. All the points within the size of the b-box are the filtered points by a pass filter.

$$\text{All the points } (x, y, z) \text{ where } \begin{cases} x, & x \in [-x_t, x_t] \\ y, & y \in [-y_t, y_t] \\ z, & z \in [z_1, z_2] \end{cases} . \tag{5.1}$$

To set the proper threshold, we look up the whole dataset and make a cumulative plot for all the points in terms of coordinate x. Since the scanning is circularly performed, only examining the coordinate x will imply the same information for coordinate y. The plot is presented in Fig 5.2.

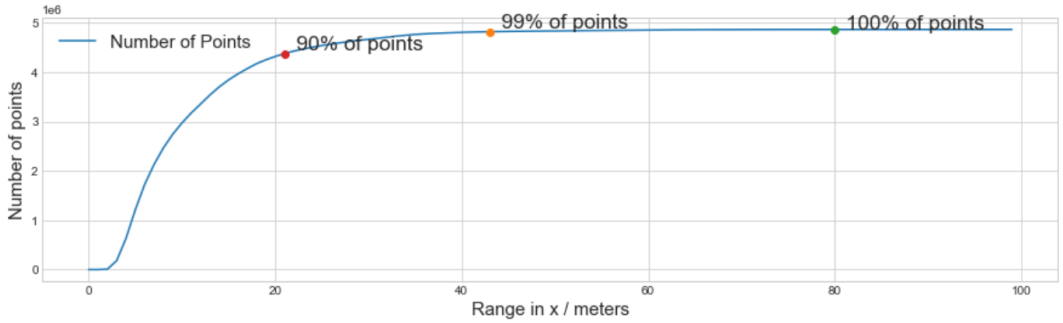


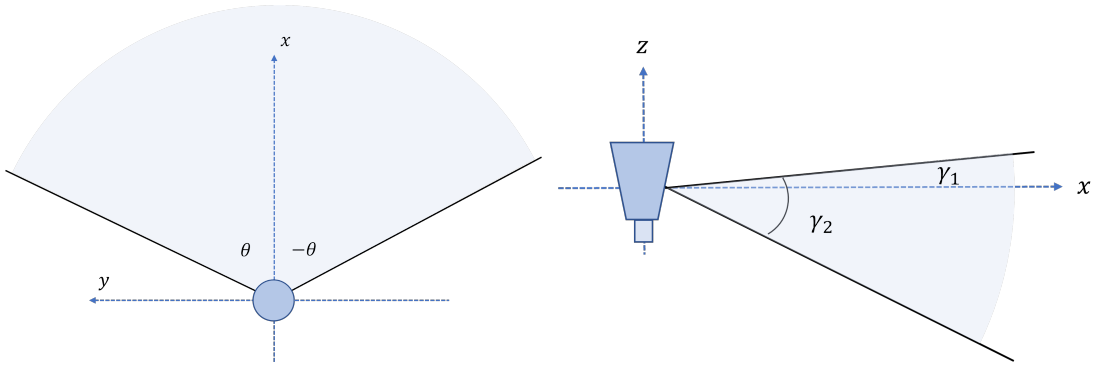
Figure 5.2: The cumulative distribution on range x in meters. The 90%, 99%, and 100% critical points are labeled on the plot. It gives the reliable reference on range selection for the pass filters.

Therefore, x is set from 0 to 50, while the threshold of y is from -40 to 40. Therefore, we preserved more than 99% of the major points and filtered out some outliers too far from the sensor. It is unnecessary to set thresholds for z, since it is not the key feature. We discard all points where x is less than 0, because the points located behind the LiDAR cannot have a common visible area with the camera. If there are no such areas, calibration cannot be achieved. Therefore, a crucial prior information is that the camera and LiDAR are also roughly facing the same direction without calibration.

The FOV filter can also realize similar ideas. Since all the lasers are considered emitting from a single point, the range of the angles can also be used as selection criteria. The [5.2] shows the selecting requirement of the FOV filter. If a point is located in the blue ROI (Region Of Interests) in Fig 5.3, both of the criteria must be fulfilled.

$$\text{All the points } (x,y,z) \text{ where } \begin{cases} \arctan\dfrac{y}{x} < \theta \text{ and } \arctan\dfrac{y}{x} > -\theta \\ \arctan\dfrac{z}{\sqrt{x^2+y^2}} < \gamma_1 \text{ and } \arctan\dfrac{z}{\sqrt{x^2+y^2}} > -\gamma_2 \end{cases}. \tag{5.2}$$

Therefore, by setting all the angles, the region of interests can be selected by the FOV filter.

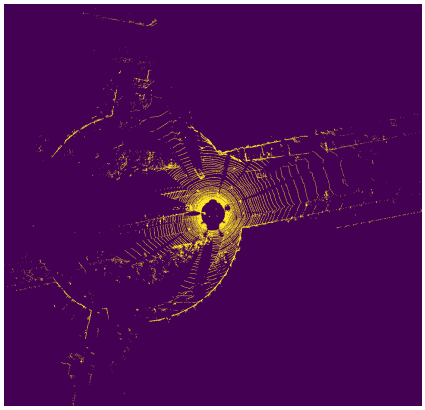


(a) Top view illustration for the scanning range. (b) Left view illustration of the scanning range.

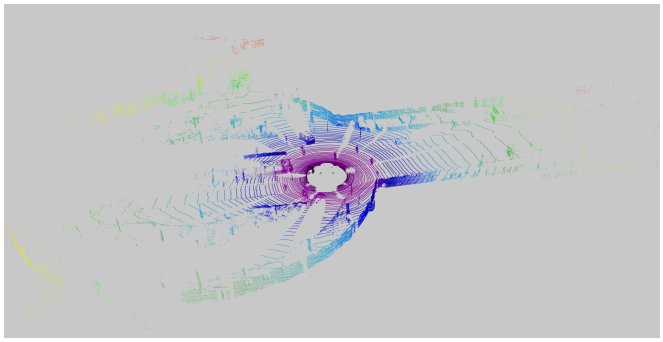
Figure 5.3: The illustration of FOV filter.

In Fig 5.4, an example of pass filter and FOV filter is presented. The BEV (Bird's

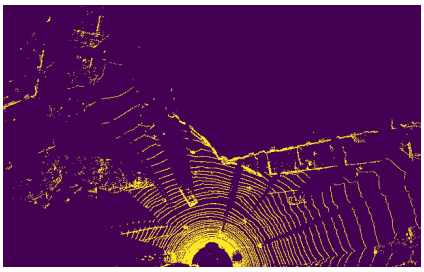
Eye View) image and the 3D realization for the original point cloud are shown above, where we can see that all points around the data collection vehicle are covered. After the pass filter and the FOV filter, half of the points behind the sensor are sliced off. The remaining points comprise the region of interest.



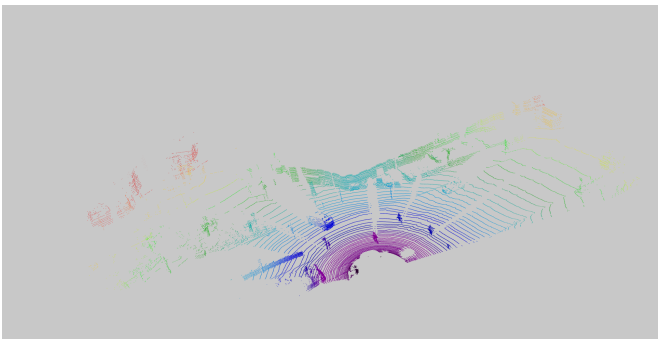
(a) The original point cloud without pass filter in bird's eye view.



(b) The original point cloud without pass filter.



(c) The point cloud after the pass filter in bird's eye view.



(d) The point cloud after the pass filter.

Figure 5.4: The illustration of pass filter.

5.2.1 Cylindrical Projection

One of the challenges for point cloud processing is edge extraction. The depth discontinuity method exploits the gaps of depth values between points in each beam, therefore performing the edge detection, but only in the vertical direction. To further exploit the edge of the sparse point cloud, the previously introduced joint calibration-fusion method uses the idea of sensor fusion to generate the dense depth map after projecting all the inlier points onto the image plane. The idea of performing depth completion is due to the need to calculate the edge of the point cloud. However, as described earlier, the optimization pipeline gets retarded due to the repetition of dense map completion. It is because when projecting points onto the image plane given the extrinsic parameters, only limited number of the points will eventually included, since the camera covers smaller field of view compared with LiDAR. That is the reason for re-projection and re-generate the dense map in the current pipeline.

Therefore, we propose a cylindrical projection for point clouds, which is able to address the previous problem. It only depends on the LiDAR point clouds, without any association with camera images. In Fig 5.5, we present the input and output of this block in the pipeline.

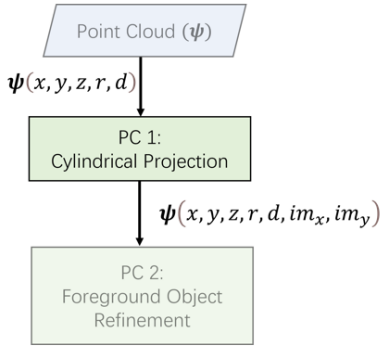


Figure 5.5: The block of PC 1: Cylindrical Projection. From the block, we can see that the input only contains the abovementioned filtered point cloud. This shows the successful separation from images.

The input point cloud ψ only contains 3D coordinates, the depth values, and the measured reflectivity values as features. The cylindrical projection is based on the location of the points in 3D. The output of this process appends the point cloud with the corresponding location of each point on the panoramic image. Fig 5.6 shows a sketch of the cylindrical projection.

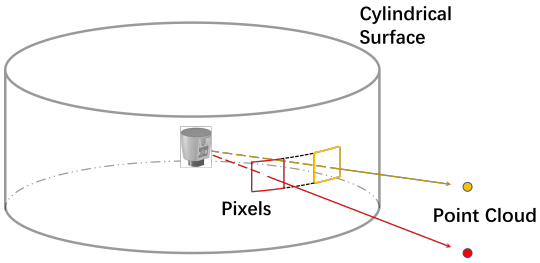


Figure 5.6: A draft of cylindrical projection of LiDAR point cloud.

According to the scanning beam of the LiDAR, we can arrange the laser point cloud by the azimuth angle and elevation angle. In order for a complex point cloud to form a picture neatly, we need to calculate the working frequency of the LiDAR at which it emits laser light and the maximum and minimum angle interval. By calculating the angle, each point will be a pixel on the panoramic image. Fortunately, the scanning beams of the 64-line LiDAR are parallel, so the image height is 64 or an integer multiple. In addition, we still need to know how many laser points can be generated by each beam in the field of view because this will be the minimum width of the image. The angular velocity of each emission (frequency) of the LiDAR system will be the horizontal resolution, and the interval between the elevation angles will be the vertical resolution. Similar to [5.2],

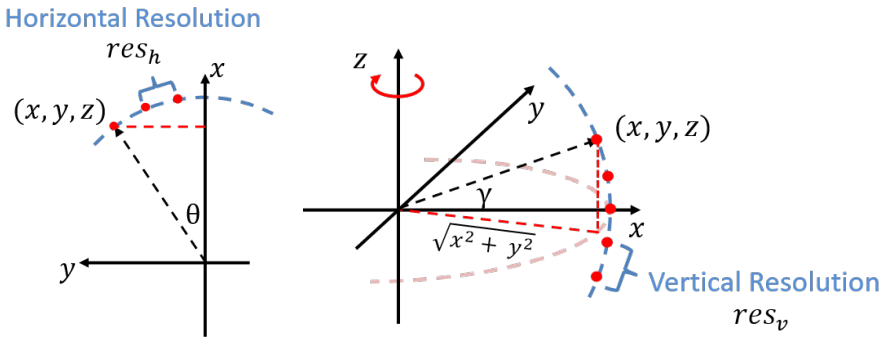
the elevation angle γ and the azimuth angle θ for a 3D point (x, y, z) will be

$$\begin{cases} \theta = \arctan \dfrac{y}{x} \\ \gamma = \arctan \dfrac{z}{\sqrt{x^2 + y^2}} \end{cases} . \tag{5.3}$$

Given the horizontal and vertical resolution, the 3D point will be converted into 2D on the panoramic image plane. The coordinate (i, j) of pixel on the panoramic image projected from point (x, y, z) in 3D space is defined in equation 5.4.

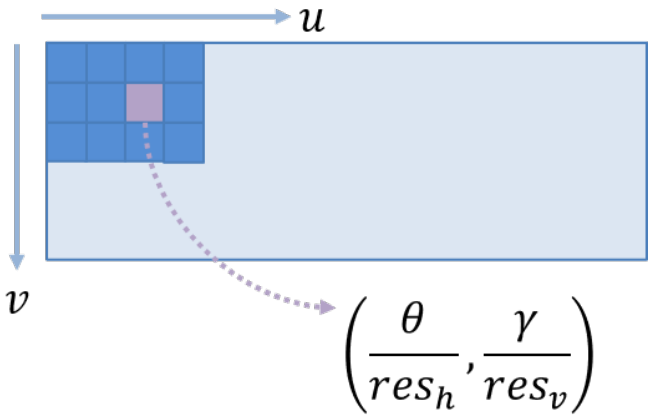
$$\begin{cases} i = \frac{\theta}{res_h} \\ j = \frac{\gamma}{res_v} \end{cases} , \tag{5.4}$$

where θ and γ are previously defined in 5.3. Given the base resolution, the maximum width of the panoramic image is defined by the horizontal FOV while the vertical FOV defines the maximum height. The illustration of this conversion is shown in Fig 5.7.



(a) Top view illustration for the selected range.

(b) Left view illustration of the scanning range.



(c) Left view illustration of the scanning range.

Figure 5.7: The illustration of FOV filter.

After the cylindrical projection, all the laser points within the FOV will be included. Therefore, all the 3D edge features are converted into a 2D map and extracted directly.

In other words, all the edge features of the point cloud are possessed in a single panoramic image. Therefore it is not necessary to project points and perform feature extraction inside the optimization loop. According to the corresponding coordinates, for each pixel, we append its position on a 2D panoramic image so that the association can be easily tracked. All the other features extracted from the 2D map will automatically correspond to the 3D point. It is the crucial idea of the cylindrical projection.

There is another benefit of the cylindrical projection, especially when there are objects in the foreground while some others are in the background. An image from [36] clearly shows the problem.

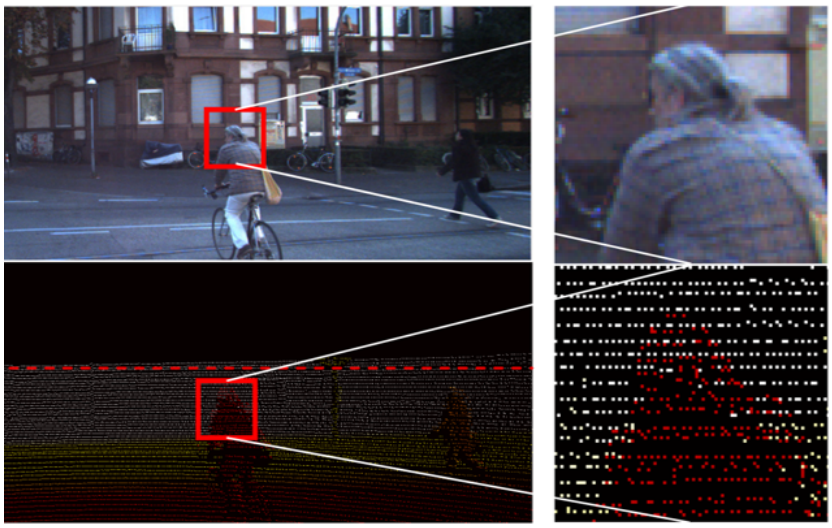


Figure 5.8: An example of the occlusion problem from [36].

If we look closer at the cyclist in the image, the background points somehow interfere with the main objects in the foreground if the extrinsic projection is directly used. Due to the occlusion problem, the foreground points are in front of the background from the perspective of extrinsic projection. So in some particular regions, the points from both foreground and background will be blended.

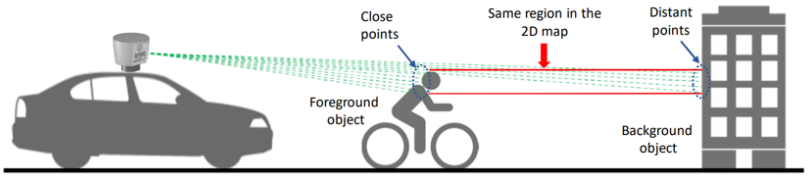


Figure 5.9: The reason of the occlusion from [36]. The foreground and background are scanned by laser in different beams. However, due to the projection perspective, they are mixed in some regions.

The cylindrical projection, in some sense, avoids this problem by separating beams according to their elevation angles. Therefore, the points from different beams can no longer be located in the same row of the image. Although the panoramic image is slightly

distorted due to the scheme of the cylindrical projection, the results are still acceptable and adequate for further process. We present the results of the panoramic image in Fig 5.10.

(a) The panoramic image with base resolution.

(b) The panoramic image with 2× base resolution.

Figure 5.10: The panoramic image with different resolution. Although presented in the same size, the pano-image in the bottom is actually 2 times larger than the top one, both in width and in height.

We also show the difference between our own cylindrical projection and the projection given extrinsic parameters. In the figure 5.11, we visualize the results of these two different approaches. We can observe that although the resolution and the main body of the cyclist and the pedestrian are better in the left image, the occlusion effect drastically degrades the edge, even after the dense map completion. There are some empty pixels in black on the right image due to the sparsity of the point cloud. The edges are precise and with no occlusion effect. Therefore, the sparse pano-image has an advantage in the sections afterward when dealing with the dense map completion algorithm.

(a) The cyclist in joint calibration-fusion method after the depth super-resolution.

(b) The cylindrical projection with base resolution on the cyclist, without the dense map completion.

Figure 5.11: The left image is the projection in joint calibration-fusion method introduced in chapter 4.3. The image on the right is the panoramic result.

The underlying fact of using cylindrical projection on the LiDAR points is that the calibration problem is converted from 3D-2D data registration to 2D-2D edge alignment. Later on, when performing the data processing, we extract the edge features from the 2D panoramic image as the edges of the 3D point cloud. Therefore, the complicated 3D point cloud edge extraction is simplified as 2D image edge detection, which is already a

mature topic within the field of image processing. All the features are collected inside the panoramic image and are accessible out of the optimization loop. This data separation technique is another vital advantage of using cylindrical projection in our pipeline. In the following sections, we focus on further processing on the panoramic image from the 2D image processing perspective rather than point cloud processing in 3D.

5.2.2 Foreground Object Refinement

At low resolution, the cylindrical projection in our process can effectively solve the occlusion problem. Although low-resolution maps are faster to perform the dense map completion, they also compress information, resulting in relatively larger errors in calculating features. On the other hand, the occlusion problem still exists to some extent when the resolution is increased to more than three times the base resolution. Therefore, we decide to use DBSCAN (Density-Based Spatial Clustering of Applications with Noise) for rough target recognition to separate foreground from background. We extract the main objectives within the LiDAR field of view and obtain a target map as the third feature. In addition to separating the foreground from the background, one advantage of the target map is the ability to extract the bottom boundary.

Figure 5.12: A typical example of the missing bottom edge. This is due to the low depth discontinuity on the bottom, since they are closer to the ground points, therefore less distinct.

As shown in Figure 5.12, the depth of the bottom edge of an object is usually almost equal to the depth of the ground region near it, resulting in the bottom boundary being ignored when distinguishing targets. The object map can effectively identify almost all points belonging to objects by clustering and provide more reliable object boundaries. Since DBSCAN clustering is based on the spatial location of point clouds, it is similar to the depth feature used for depth maps.

To successfully perform the object clustering, it is necessary to perform another filter for the whole point cloud to modify the connectivity. Therefore, our foreground object refinement includes 2 stages: Plane segmentation and object clustering.

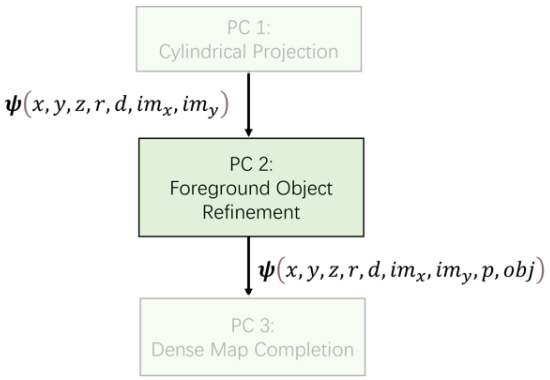


Figure 5.13: The position of the current block (PC: 2) in the whole pipeline.

After these stages, the point cloud will append two new middle-stage variables: binary p to illustrate if the point is a plane point and binary obj that shows if the point belongs to the foreground object. We will explain the process in detail in the following subsections.

5.2.2.1 RANSAC Plane Segmentation

Ground plane points in the point cloud usually provide less adequate information for the calibration process. Conversely, due to the huge number of points, ground points can cause noise in the objective function and mismatches in the results. Moreover, in the process of DBSCAN, because ground points are generally adjacent to each other, the clustering results will be seriously affected. Therefore, in order to improve the accuracy of the identification of the main objects and eliminate the interference from ground points, we use a fast and convenient method called RANSAC (RANdom SAmple Consensus) to coarsely remove ground points in preparation for the upcoming clustering step [12].

The simplest way of ground point selection is based on the height, which is represented by z coordinate. If given the prior knowledge of the height of the LiDAR and knowing the LiDAR system is set parallel to the ground, all the points on the ground should be approximately the same as height as the LiDAR, but with minus sign in the z coordinate. The idea of this method is similar to a pass filter, which is efficient to perform.

However, for the calibration problem, we try to avoid knowing the prior information of the pose. Therefore, the assumption of parallel is not available. On the other hand, the RANSAC algorithm does not require such an assumption. The idea of the algorithm is to identify the outliers from inliers that do not fulfill the requirement of a pre-determined mathematical model. In our case, 3 points are taken in a single trial since 3 points can uniquely determine a plane. The whole algorithm can be described as selecting the best plane model that fits the most significant number of inlier points.

The input for the algorithm is the point cloud $\boldsymbol{\psi}$, the threshold value for the distance, the total number of iterations, and the initial inlier number $k = 0$. In each iteration, 3 points are randomly selected, in order to construct a unique plane function. The plane function can be generalized as $Ax_i + By_i + Cz_i + D = 0$, with 4 parameters A, B, C and D. With the plane model function, the distance of each point $P(x_i, y_i, z_i)$ to the plane

can be calculated by (5.5).

$$d = \frac{|Ax_i + By_i + Cz_i + D|}{\sqrt{A^2 + B^2 + C^2}} \tag{5.5}$$

We append another feature as p to the point cloud, denoting if a point is plane or not. If the distance is larger than the threshold, it will be defined as outliers for each point. In this case, the point is classified as non-plane, therefore p is set as 0. The rest of the points are inliers. Moreover, the total number of inliers of the current model is accumulated as k_{iter}. If the plane model we find in this trial has larger k, it means that this model fits the data better than the previous best model. Therefore, the new model parameters will be updated, and the current maximum number of inliers k will be saved for further comparisons. This process statistically approaches the optimal plane, especially when the total iteration is large since it updates with a better model step by step. When the pre-defined iteration is achieved, the algorithm ends up with a list containing all the labels of inliers and outliers. The algorithm of RANSAC is described in detail in Algorithm 3.

Algorithm 3: RANSAC for plane segmentation [12]

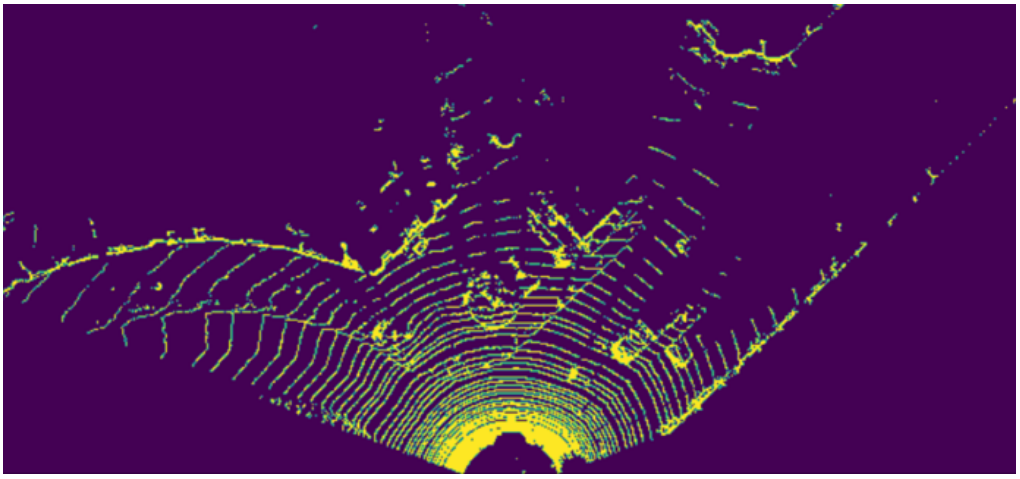
Input: point cloud $\boldsymbol{\psi}$ with n entries, number of iterations num, initial number of inliers k and the threshold value $threshold$.

Output: The label p, the plane model parameters A, B, C and D.

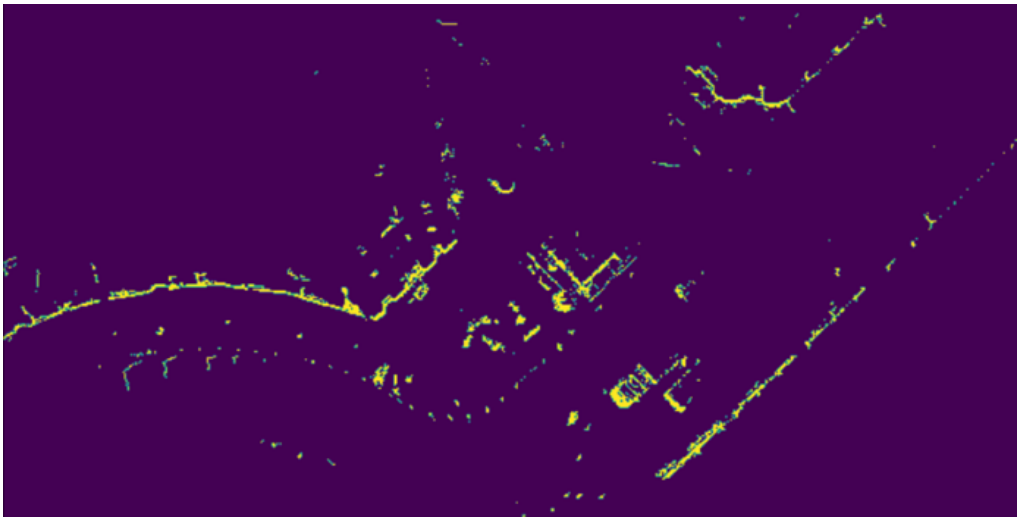
```
iter = 0.
while iter < num do
    i = 0.
    Randomly select 3 points from ψ.
    Fit the plane model Ax + By + Cz + D = 0.
    while i < n do
        Calculate the distance from ψ_i to the plane dist_i according to (5.5).
        if dist_i < threshold then
            p_i = 1.
        else
            p_i = 0.
        end
        i = i + 1.
    end
    k_iter = Σ_i p_i.
    if k_iter > k then
        update: A, B, C, D.
        update: k = k_iter.
    end
    iter = iter + 1.
end
```

Therefore, given the optimal plane model, all the points labeled as inliers are considered as the plane points. They can be easily filtered out by constraining $p = 0$, which selects only the non-plane points. All the remaining points are the actual LiDAR mea-

surements from objects. Figure 5.14 illustrate the performance of the RANSAC plane segmentation algorithm from the perspective of bird's eyes.



(a) The BEV map before plane segmentation.



(b) The BEV map after plane segmentation.

Figure 5.14: The BEV (Bird's Eye View) maps of the point cloud before and after RANSAC plane segmentation. The ground points in circles and arcs are roughly removed while the object points are preserved.

5.2.2.2 DBSCAN Clustering

In order to perform the clustering algorithm to separate the foreground and background targets, RANSAC is used in advance to filter out the ground points roughly. The remaining points are the object points with particular shapes. One feature that can be exploited to identify different objects is the density of the points. It is because, excluding the plane, objects are spatially far from each other. The density or scattering patterns of object points are different. For example in Fig 5.14, density of points within one object is much higher than density of points between different objects. Therefore, the

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an efficient solution to perform the clustering for point cloud with ground points excluded [10].

DBSCAN is a density-based clustering algorithm that can intelligently identify densely connected regions. It can divide regions with high density into clusters and find clusters of arbitrary shape in noisy point cloud. In short, it is an algorithm based on a process that expands continuously according to the density of objects. The algorithm inputs include the ground-free point cloud, the 3D detection ball with R radius, and the defined minimum points m within the R-range spherical space. To mathematically define the number of points within R-range sphere, we use the concept of cardinality, which is usually applied to express the number of element in a set. We define the total number of point within within R-range sphere given the center point p as $\text{card}(S_R^p)$.

In the beginning, all the points in the point cloud are labeled as 'unvisited'. The algorithm randomly picks up an unvisited point p as initial point and marks it as visited. Then, a deep search around p is performed if there are more than m points within the R-range space and point p will be put into a new cluster $\mathcal{C}$. All the other points within R-range will be saved temporarily in the set S. Next, for every point in S, a similar search will be performed. Points will be recursively classified to $\mathcal{C}$ if the requirements are fulfilled. Therefore, the region of cluster $\mathcal{C}$ is growing gradually. Some points that do not fulfill the requirement of having m neighbors within R-range will not be assigned to any of the existing clusters. Those points are classified as outliers and removed from the object point cloud. The full detail of the algorithm is presented in Algorithm 4.

After DBSCAN, we receive each points with their cluster index. It is necessary to calculate the centroid of each cluster to identify the foreground and background clusters. The calculation of centroid can be performed easily in (5.6) since only the depth information is the key for such separation.

$$d_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \sqrt{x_i^2 + y_i^2} \tag{5.6}$$

The k indicates the index of the cluster, N_k is the total number of points in cluster k.

We filter out all the clusters with the depth centroids that locate out of 15 meters range according to the experience. Therefore we can have a new point cloud that only contains the meaningful foreground points. The object is considered as a new feature of each point. For foreground objects, the mark is 2. For the background objects, all the points are labeled as 1.

$$obj_k = \begin{cases} 2, \text{if } d_k < 15 \\ 1, \text{otherwise} \end{cases} \tag{5.7}$$

According to (5.7), all the points are assigned as foreground as long as the centroid of the cluster it belongs is smaller than the threshold, even if the point itself may have depth value that is higher than the threshold. All the outlier points that fail to form a cluster are assigned as background.

Algorithm 4: DBSCAN Algorithm [10]

Input: Object point cloud $\boldsymbol{\psi}'$ with N entries, minimum number of points m to form a cluster, and the radius value R

Output: The set of clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$.

for $i < N$ **do**
  $v_i = 0$ // Mark all the points as unvisited
end
while $\sum_{i=1}^{N} v_i < N$ **do**
  Arbitrarily choose an unvisited point p.
  $v_p = 1$
  if $\text{card}(S_R^p) > m$ **then**
    Create a cluster $\mathcal{C}$
    $\mathcal{C} = \mathcal{C} \cup p$
    $S = S_R^p$
    for $q \in S$ **do** // Iterate every point q in set S
      if $v_q = 0$ **then**
        $v_q = 1$
        if $\text{card}(S_R^q) > m$ **then**
          $S = S \cup S_R^q$
          if $q \notin \mathcal{C}$ **then** // If q does not belong to any cluster
            $\mathcal{C} = \mathcal{C} \cup q$
          end
        end
      end
    end
  end
end

Figure 5.15: The sparse foreground object map.

However, the total number of points is too small to perform the dense map completion. The smooth transitions will fill all the empty spaces in order to minimize the total variation norm. If including the background points for the depth completion, the same problem remains: The background points are mixed within the region of foreground points.

As the solution, the morphological operators are used to cover the areas that tend to have the problem of occlusion. We firstly apply the dilation to fill up the empty holes, then the erosion to reduce the growing size of objects. These morphological processing tricks are often used in biological or medical image processing to remove particular noises from the microscopes' measurements or partition the tightly neighboring cells. Similarly, the idea can also be applied in our case to construct a dense object mask for the foreground objects.

For gray-scale images, the dilation is to apply a statistics filter that returns the maximum intensity value within the range of its neighbors defined by a moving kernel. The process can also be expressed mathematically in (5.8):

$$(f \oplus b)(x) = \sup_{y \in E}[f(y) + b(x - y)] \tag{5.8}$$

where $f(x)$ is considered as an image and operator $\oplus$ denote the dilation. The sup denotes the supremum, while in this case, it can be understood as maximum. The E is the region inside the kernel. Function $b(x)$ is the value of each element of the kernel. In our case,

$$b(x) = \begin{cases} 0, & x \in B \\ -\infty, & \text{otherwise} \end{cases}, \tag{5.9}$$

where $B \subseteq E$. The constraint on $b(x)$ indicates the calculation is proceeded locally. The supremum is the same as maximum in this bounded case.

The size of the kernel can be derived through the scaling parameter of the resolution of cylindrical projection. If the scaling parameter is 1, we assume that the points are tightly placed close to each other without empty holes. This can be confirmed by the top figure from Fig 5.10. Ideally, if the scaling parameter is 2, there are 1-pixel gaps between each LiDAR point on the panoramic image. Therefore, suppose the scaling parameter is an integer k, the minimum radius for dilation should be $k - 1$. Through the dilation process, the empty region close to LiDAR realizations will be filled by the foreground-object value two given a particular $(2k - 1) \times (2k - 1)$ size of the kernel iterating every pixel in the original sparse foreground-object map.

The dilation successfully fills the holes on the sparse map. However, it also enlarges the original area of the background object since the kernel enlargers pixels in every direction. A morphological erosion operator can be used to shrink the area of the foreground objects. On the contrary to dilation, erosion for the gray-scale image is to assign the minimum intensity value within the range of the kernel. It can recover the growing edges caused by dilation while preserving the regions within the objects that are already filled by dilation (denoted with $\ominus$).

$$(f \ominus b)(x) = \inf_{y \in B}[f(x + y) - b(y)] \tag{5.10}$$

The equation of erosion is similar to dilation. The infimum implies the local minimum value of the neighbors. The function $b(x)$ is still the flat structuring element with 0 value inside the window, while positive infinite elsewhere. An example of erosion after dilation is shown in Fig 5.16.

(a) The dilation operator on the sparse object map.

(b) The erosion operator on the result of dilation.

Figure 5.16: The process of morphological closing, which includes dilation (top) and erosion (bottom). The sparsity inside the object range is fixed by dilation, while the dilated shapes are eroded by erosion afterward. Therefore, the object mask is constructed, which can filter out the background points that are located on the foreground mask after cylindrical projection.

The combination of erosion after dilation is known as morphological closing. By firstly performing the dilation, all the points with values will grow to their adjacent pixels. Therefore, small holes are filled up with values. Nevertheless, it will also cause the problem that the areas are over-covered by the object pixels. We follow up with erosion to "grow" the background pixels, recovering the area that is taken up during the dilation. Meanwhile, since the empty holes inside the object are mainly filled, the original holes are no longer background pixels. In the end, the tiny holes will not show up after the erosion.

The process can be observed from Fig 5.17. After performing the dilation, the empty holes in the head and the back area are eliminated for the cyclist. On the other hand, the arm of the cyclist almost merges with the body. The space between them re-appears after the erosion, while the empty holes are filled up permanently. We use this object map as a masking image. All the background points locate on the mask after the cylindrical projection are discarded directly. Therefore the points that could cause the occlusion effect no longer exists. After these steps of morphological operation, the results of the dense depth map are improved on the edges. In Fig 5.17, we show the depth map before and after the morphological operation.

(a) The occlusion on 3 × base resolution.

(b) The occlusion disappears after the foreground-background separation.

Figure 5.17: The comparison of the occlusion effect. On high-resolution map, cylindrical projection cannot fully eliminate the occlusion. The result of the cylindrical projection is shown above. After the object clustering and morphological operation, the occlusion is completely removed from the dense map.

This sub-section introduced our efficient method for foreground-background objects separation by using plane segmentation, density-based clustering, and morphological closing. We also created the object map to enhance the depth map, providing the reliable shape of the foreground objects. This object map will improve the feature selection in further processes. The binary variables p and obj are appended in the feature vectors in the point cloud feature data. They represent the binary status of a point, if it is a plane point or a foreground object point. These two parameters do not directly contribute to the final objective function but are still important during the current stage since the desired points can be easily selected, considering these binary values as criteria.

5.2.3 Dense Map Completion

In the previous section, we introduced the sparse image created by performing the cylindrical projection. The feature vectors with plane and foreground object information are also provided but in a sparse form. In this section, we will introduce how to create dense maps given a sparse input image. It is the third stage of point cloud processing, which locates in the middle of the pipeline shown in image 5.18. Note that we use $\boldsymbol{\psi}$ to denote the sparse panoramic image and $\boldsymbol{\phi}$ to denote the dense panoramic image.

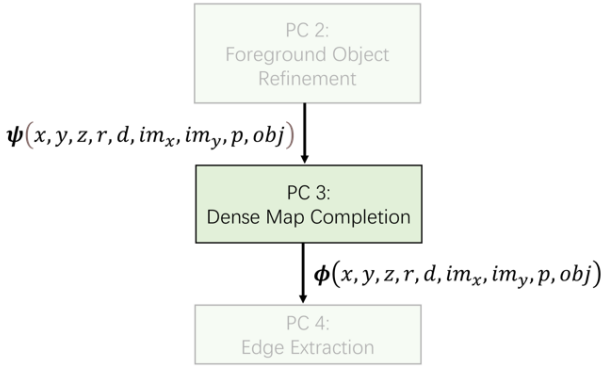


Figure 5.18: The block of PC 3: dense map completion stage with input and output features in the whole pipeline.

The best solution that is able to exploit the edge features carried by the 2D map is to perform the dense map completion, which can up-sampling the sparse map as a dense map. Similar to the form of the function in the joint calibration-fusion method in (4.19), we use the ℓ_2 norm to assure the data fidelity while using the anisotropic total variation norm to preserve the sharpness of the edge while smoothing the rest of the image.

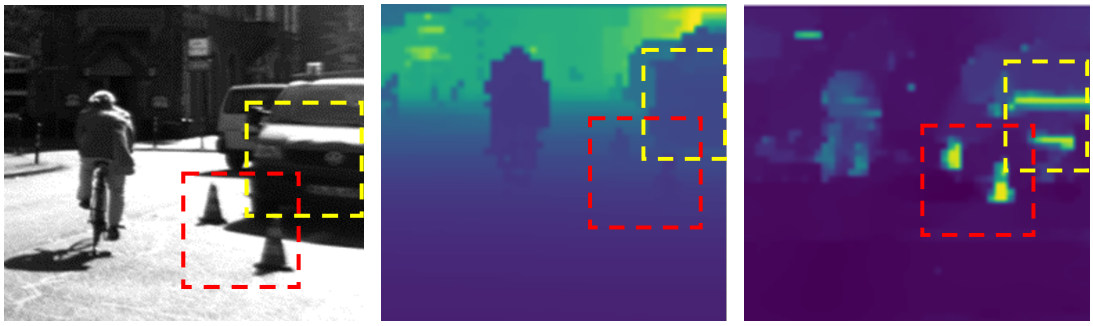
Suppose the $\boldsymbol{\phi} \in \mathbb{R}^{N_x \times N_y}$ is the desired dense panoramic image. The capital $\tilde{\mathcal{P}}$ represents the sparse panoramic image. Similarly, the $\mathbf{H}$ is the binary masking matrix, performing the selection on the locations, which should be the same as the sampled sparse panoramic image. We choose the anisotropic total variation norm in order to simplify the calculation while leaving little impact on the final result.

$$\hat{\boldsymbol{\phi}} = \text{argmax}_{\phi \in \Phi} \left\| \tilde{\mathcal{P}} - \mathbf{H} \odot \boldsymbol{\phi} \right\|_2^2 + \lambda \sum_{n=1}^{N} \| [\boldsymbol{\nabla}_x \boldsymbol{\phi}]_n \|_2 + \lambda \sum_{n=1}^{N} \| [\boldsymbol{\nabla}_y \boldsymbol{\phi}]_n \|_2 \tag{5.11}$$

The first term constrained by ℓ_2 norm is the measure of the data fidelity. It limits the values of the image in the given positions by the masking matrix. The latter two terms are the anisotropic total variation norms. They limits the total amount of changes, therefore ensure the smoothness. On the other hand, they are similar to ℓ_1 norms, which provide sharp edges on the locations with rapid changes. By minimizing the data fidelity and the total variation norms, the dense panoramic map can be recovered.

The features that are used to perform the dense map completion are both depth information and reflectivity information. As introduced in chapter 4.2, objects in the FOV tend to be isolated in urban road scenarios. Therefore, the edge of the depth map tends to be the edge of the object, thus being used to match the edges on the grayscale image. However, in traditional methods, as stated in table 4.1, no one has included the reflectivity values as a metric for edge detection. Generally, because the sparse reflectivity values are not easy to use and due to the measurement noises from the mechanism of LiDAR, sometimes the reflectivity values are not always reliable. The reason for also including the reflectivity is based on the observation that although the majority of the reflectivity values are lower than 0.4 (majorly the road surface and the building in the background), on the road scenario, there are a lot of traffic cones, signs

and other traffic-related objects that use illumining materials providing high reflectivity values. Also, white areas generally provide higher reflectivity values. Therefore the reflectivity values can be used to add some details to the pattern of the surface. For example, in the reflectivity map in Fig 5.19, the traffic cone is distinct. While in the depth map, it is buried in the ground pixels. Because it is very short, bringing little depth difference, therefore it is not very obvious in the depth map.



(a) The original image. (b) The depth map. (c) The reflectivity map.

Figure 5.19: The original image of the traffic cone on the ground, compared with the depth map and the intensity map, given base resolution. The red boxes point out the area with traffic cones while the yellow boxes indicates the front of a vehicle, with car plate and white body.

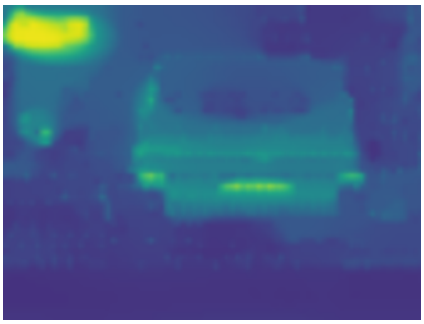
The yellow box at the middle right of the reflectivity map are the license plate and the white front cover of the vehicle. It provide high contrast in the camera image, while absolutely invisible in the depth map. Therefore, it can be assured that the reflectivity can provide some patterns and details (especially for white color and high-reflection materials) within the object surface while the depth map cannot.

Another example on the vehicle gives an even stronger argument on the necessity of combining reflectivity values as another feature. In the figure 5.20, we present a vehicle in intensity image, depth panoramic image, the reflectivity panoramic image and the half-half mixture of both depth and reflectivity features. On the depth map, the edge of the object is very distinct. There is a clear shape of the contour of the vehicle. On the contrary, the white pillar on the left of the vehicle is emphasized. The car license plate, the lights are also visible, while these are not presented on the depth map. More importantly, those patterns also exist in the camera image. However, the left rearview mirror of the reflectivity map disappears completely. Therefore, the combination of the coarsely mixed image (50% transparency of depth map + 50% transparency of reflectivity map) on the bottom right provides both the shape and the pattern of the vehicle.

To further utilize the edge of the foreground object, the binary object information is also considered. The background objects are known as zero on this feature, but the actual sparsity comes from the empty area with no information. Therefore, to distinguish the background binary feature 0 and the sparse areas (filled with 0), we set the foreground information as 10, the background areas as one, and the rest regions with no information as 0. Therefore, we can also construct a dense map of foreground objects. In Fig 5.21, we give examples of the dense maps of these three features.

(a) The original image.

(b) The depth map.



(c) The reflectivity map.

(d) The mixed map on both depth and reflectivity.

Figure 5.20: The original image of the white vehicle on the ground, compared with the depth map, the intensity map and a mixture by given half transparency on both depth and refletivity maps. The resolution is 3× base resolution, in order to recover more details.

(a) The dense depth image.

(b) The dense reflectivity image.

(c) The dense object image.

Figure 5.21: The dense maps of depth, reflectivity, and object features, respectively. The depth map is smooth and continuous in major areas, while the reflectivity map is noisy. However, the high reflectivity areas are distinct and easy to extract. The object map loads the artificial features, providing the foreground features given the clustering results. It provides the clear bottom edges of different targets, while the depth map does not.

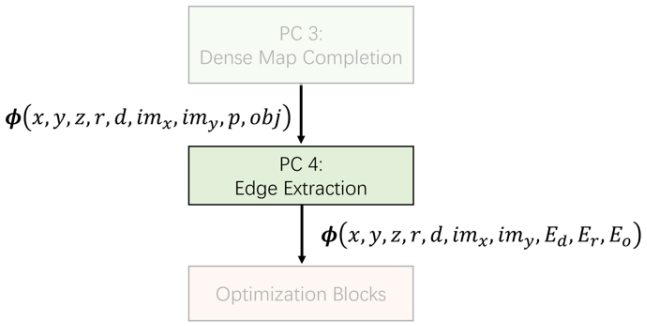


Figure 5.22: The final stage of point cloud processing: Edge extraction.

In this block, the input features are the points with object information. We keep those features untouched while fills the empty holes in the sparse image. To distinguish the difference between the input and the output, we apply different notations. The sparse input panoramic set $\boldsymbol{\psi}$ becomes a dense image $\boldsymbol{\phi}$, shown in figure 5.18. The output dense feature maps (depth, reflectivity, and object information) are moved on for the edge extractions in the next section.

5.2.4 Point Cloud Edge Extraction

The previous sections introduced how to use cylindrical projection to convert the 3D laser point cloud into a 2D sparse panoramic image and how to perform the dense map completion of the image and solve the occlusion problem. After those steps, we get three dense 2D panoramic images with depth, reflectivity, and object features. Therefore, the problem of extracting the edge of the 3D laser point cloud is transformed into extracting the edges of the 2D panoramic image. In this subsection, we will focus on how to extract the image edge information. The details of block PC: 4 is shown in Fig 5.22 From the I/O perspective, the temporary parameter obj and p is discarded since it no longer participates in any calculation. The newly calculated edge features are extended in the final point cloud. In this section, we will explain how to extract these edge features from dense maps.

At present, the commonly used edge alignment step of extrinsic calibration is still to project the boundary points of the laser points onto the grayscale image, and then carry out further edge matching. If we can effectively determine which points are boundary points for the sparse laser point cloud, it will be much easier in the next projection and matching. If too many points are identified as boundaries, it will increase the error during matching and affect the final matching effect. Relatively speaking, non-boundary points are less influential in our edge matching workflow. Therefore, how to extract the boundary of the laser point cloud as accurately as possible is transformed into extracting the boundary of the panoramic image as accurately as possible. In essence, this is still an image processing problem.

The most direct embodiment of the image boundary is the position where the pixel value changes. The change of pixel grayscale value can be tracked by derivative. A digital image $\mathbf{u}$ can be understood as a two-dimensional discrete function (gray value

from 0 to 255). Then, its change can be calculated by difference. The acquisition of edges can be made by taking the derivative of this discrete function, shown in equation set 5.12.

$$\begin{cases} \dfrac{\partial \mathbf{u}(x,y)}{\partial x} = \mathbf{u}(x,y) - \mathbf{u}(x-1,y) \\ \dfrac{\partial \mathbf{u}(x,y)}{\partial y} = \mathbf{u}(x,y) - \mathbf{u}(x,y-1) \end{cases} \tag{5.12}$$

The derivatives of a pixel in image $\mathbf{u}$ on both x and y directions are calculated using the difference of intensity values of its adjacent pixels. To effectively perform the calculation on the image scale, convolutional kernels are often used. In the current traditional methods, the commonly used edge extraction methods are, for example, Sobel kernel based on the first-order derivative, Laplacian kernel based on the second derivative and their variants. However, the best effective one is still the algorithm called Canny edge detection proposed by Canny in 1986 [4]. The following is a brief list of some convolutional edge extraction operators and their characteristics.

Name	Description	Characteristics
Roberts Kernel	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	This is better for images with steep changes and low noise level. Two directional kernels required. Sensitive to noise.
Sobel Kernel	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	This is better for images with gradual gray-scale change.Two directional kernels required. The results of noisy image are acceptable.
Laplacian Kernel	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	This is accurate to locate the step-like edge pixels. This is efficient since only one kernel is used. Not able to fetch the directional information. Extremely sensitive to noise.

Table 5.1: Comparison of Popular Edge Detection Kernels. The example kernels inside the description are rotational in order to get edges from other directions.

For the Canny edge detection, the process is beyond a single stage of convolution. The whole Canny edge detection algorithm contains four main steps: Gaussian filtering, gradient and orientation calculation, non-maximum suppression, and hysteresis thresholding.

Using the Gaussian filtering technique aims to filter some potential noise on the digital image. Usually, a 5×5 kernel is used. The Sobel kernel is selected to calculate the gradient on both x and y directions in the second step. The Sobel kernel is a weighted filter. It reacts to the perpendicular edges and edges with orientations since the corners of the kernel also have values. We denote the horizontal gradient of a single pixel as g_x, the vertical gradient as g_y. Given the 2 dimensional gradient values, the strength (amplitude) and the angle (or orientation) of the gradient can be calculated as

$$G(x,y) = \sqrt{g_x^2 + g_y^2} \tag{5.13}$$

$$A(x,y) = \arctan \frac{g_y}{g_x} \tag{5.14}$$

The angle is rounded in 4 directions, which are vertical, horizontal, and 2 diagonal directions. After getting gradient magnitude and direction, a full scan of the image removes any unwanted pixels that may not constitute the edge. For this, at every 3×3 neighborhood, the center pixel is checked if it is a local maximum in the neighborhood in the direction of gradient [45]. Note that the direction of the edge should be perpendicular to the direction of its gradient. An illustration of the idea of non-maximum suppression is presented in Fig 5.23.

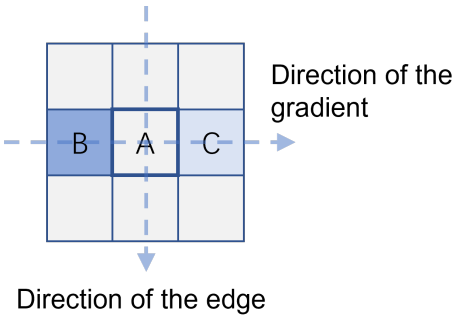


Figure 5.23: The illustration of non-maximum suppression. Point A is on edge (in the vertical direction). The gradient direction is normal to the edge. Points B and C are along the gradient direction. So point A is checked with points B and C to see if it forms a local maximum (which means, G_A is bigger than both G_B and G_C). If so, it is considered for the next stage. Otherwise, it is suppressed (put to zero).This process is also expressed in (5.15).

It is a widely-used edge thinning trick in order to suppress the thick edges of the Sobel kernel. Within a bold edge, only a one-pixel-width edge will represent the actual edge since other pixels that do not fulfill the requirement are suppressed. A thinner edge gives a more precise information when we map back the edge pixels to the edge points in 3D point cloud. Note that the selection of pixels B and C is based on the angle of the gradient of point A.

$$\text{NMS}(G_A) = \begin{cases} 0, & \text{if } G_A < \max\{G_B, G_C\} \\ G_A, & \text{otherwise} \end{cases}, \tag{5.15}$$

The final step of Canny's detection pipeline is the hysteresis thresholding. There are two values needed as upper bound and lower bound thresholds. Any pixel with a gradient value higher than the upper threshold is considered a definite edge. Similarly, any pixel value after the NMS (Non-Maximum Suppression) below the lower bound is assigned as a definite non-edge pixel. If the pixel in the middle of bounds closely connected with a definite edge is also considered a definite edge. Otherwise, the pixel is discarded. The

complete Canny edge detection algorithm is presented in Algorithm 5.

Algorithm 5: Canny Edge Detection

Input: Image $\mathbf{u}$, the upper threshold T_u, the lower threshold T_l
Output: The binary edge map I.

Calculate vertical edge map: $\mathbf{u}_v = \mathbf{u} * S_v$
Calculate horizontal edge map: $\mathbf{u}_h = \mathbf{u} * S_h$
// S_v and S_h are Sobel kernels on different directions
for (i, j) *in* $\mathbf{u}$ **do** // iterate every pixels
  Calculate $G(u_v(i,j), u_h(i,j))$ (5.13).
  Calculate $A(u_v(i,j), u_h(i,j))$ (5.14).
end
for (i, j) *in* $\mathbf{u}$ **do**
  Calculate NMS$(G(i,j))$ (5.15).
end
for (i, j) *in* $\mathbf{u}$ **do**
  if $G(i,j) > T_u$ **then**
    $I(i,j) = 1$ // (i,j) is an edge
  end
  if $G(i,j) < T_l$ **then**
    $I(i,j) = 0$ // (i,j) is not an edge
  end
  if $T_l < G(i,j) < T_u$ **then**
    if $\sum_{i-1}^{i+1}\sum_{j-1}^{j+1} I(i,j) > 0$ **then**
      $I(i,j) = 1$ // (i,j) is connected to an definite edge
    else
      $I(i,j) = 0$ // (i,j) is not an edge
    end
  end
end

In Fig 5.24, we show some examples of the results of the Canny edge detector on the pano-images. The binary edge maps are extracted by Canny edge detector from the panoramic depth map, relfectivity map and object map, respectively. The forth image at the bottom is the multi-feature edge map. It shows that edges from different feature maps, like the shape of the major objects, the reflection of the car plate, and other scenery in the FOV are also included in the mixed multi-feature map.

(a) The edge map of the depth map.

(b) The edge map of the intensity map.

(c) The edge map of the object map.

(d) The mixed-feature edge map comprises the depth edges, the intensity edges, and the object edges.

Figure 5.24: The mixture of 3 different types of edges. The depth edges provide the depth discontinuity of the environment. The reflectivity edges add some details on patterns of the objects. The object edges provide full contour of the foreground objects. The mixture of them contains the high fidelity of the edges solely from LiDAR points.

The mixed edge map is defined as shown in 5.16. We consider the same weight on each of the feature. If an edge is valued as 3, it is highly likely to be an edge. Lower values are assigned to pixels with less features. Therefore, the final multi-feature mixed map is created, with more edge information than any of the single feature map we generated above.

$$E = E_d + E_r + E_o \tag{5.16}$$

Compared to other existing edge-alignment-based pipelines that focus on extracting edges only from depth information, ours provides the depth discontinuity and the entire contour of the objects and the boundaries of the high-resolution areas. The mixed map provides more details than any other feature map, bringing rich edge information for further alignment.

5.3 Image Processing

In this section, we will focus on how to extract edge features from camera images. In the whole workflow in 5.1, the blocks are highlighted with yellow color. As with the

panoramic image we have processed before, the primary way the edges that have been extracted is still by using the pixel-wise gradient of the whole image. The difference is that due to the unique nature of the camera image, some degree of preprocessing of the camera image to continue extracting is needed. Ideally, we want the edges of the camera image to contain only the main boundaries of the object and the boundaries of high reflectivity areas, as the LiDAR point cloud also provides these after preprocessing. However, the reality is that thanks to the resolution of the camera, which is much higher than the laser point cloud, and the characteristics of cameras photosensitive devices, the camera can capture much more detail, so there will always be boundaries in matching that are not contained by the laser point cloud synthetic boundary. Similarly, many of the boundaries that can be represented in the laser point cloud are not visible in the camera image because of the low grayscale difference or the effect of shadows. Under road conditions, complex roadside obstructions, such as street trees and billboards, can leave shadows on the ground or in the field of view under sunlight. These shadows are faithfully recorded by passive sensors such as cameras. When we want to extract edge information from an image, the shadow region leaves a fairly distinct boundary that is not present in LiDAR because the boundary is not caused by changes in depth or reflectance of the object. On the other hand, due to the hardware limitations of the camera itself, the details of the shadow will be ignored due to low contrast in high-exposure cases. Likewise, this part of the boundary information is not overlooked by LiDAR. The boundary difference between the two sensors will result in the final mismatch, affecting the calibration algorithm's performance. It is hard for conventional methods to solve this problem completely, but there are still ways to reduce the effect. This chapter uses simple and effective histogram equalization for image enhancement to reduce the error in these types of scenarios. It changes the grayscale histogram of the original image from its gray interval to a uniform distribution in all grayscale ranges. Because this algorithm is simple and does not need the parameter setting of external factors, it is a commonly used method to enhance the image effectively.

5.3.1 Image Enhancement by Histogram Equalization

In this section, we will introduce the preprocessing step IM 1: histogram equalization. The block is shown in figure 5.25. The input is the camera image $\mathbf{u} \in \mathbb{R}^{M_x \times M_y}$. The pixel values are integers from 0 to 255. The output of this block is denoted as $\mathbf{u}'$, where (x, y) represents the pixel's coordinate in the image.

Numerous methods on camera-LiDAR calibration focus on the laser processing side, while they have overlooked the possible improvements from image processing. We will introduce our idea in this section by discussing the underlying assumption of edge alignment. The input is the raw grayscale image $\mathbf{u}$. A grayscale image is composed of pixels with different gray values. The distribution of gray intensities in the image is an essential feature of the image. The grayscale histogram describes the distribution of grayscale values in the image and can intuitively show the proportion of each gray level in the image. This histogram of an image is a function of the gray level, which describes the number of pixels with the gray level in the image: the abscissa is the gray level, and the ordinate is the frequency of the gray level. Histogram equalization is usually used to increase the

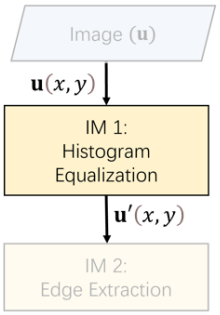


Figure 5.25: The detailed input and output of block IM 1: histogram equalization.

global contrast of many images, especially when the contrast of the valuable data of the image is quite close. By this method, the brightness can be better distributed on the histogram. In this way, it can be used to enhance the local contrast without affecting the overall contrast. Histogram equalization realizes this function by effectively expanding the commonly used grayscale values.

Assuming the image A is the original image that we want to modify. Its histogram is expressed as $H(D_A)$, where D represents the gray-scale value. The objective image that transformed from A is B, with histogram $H(D_B)$. A non-linear function $f : \mathbb{R} \to \mathbb{R}$ is used to complete this transform. The function f is pixel-wise transform. All the pixels with intensity D_A is converted to D_B, according to

$$\begin{cases} D_B = f(D_A) \\ D_B + \Delta D_B = f(D_A + \Delta D_A) \end{cases} . \tag{5.17}$$

This defines the gray-scale conversion. Also, this can be applied to

$$\sum_{D=D_A}^{D_A+\Delta D_A} H_A(D) = \sum_{D=D_B}^{D_B+\Delta D_B} H_B(D) \tag{5.18}$$

which can be interpreted as the number of pixels within intensity interval $[D_A, D_A+\Delta D_A]$ in image A is the same as the number of pixels in interval $[D_B, D_B + \Delta D_B]$ in image B. It can also be extended to

$$\sum_{D=0}^{D_A} H_A(D) = \sum_{D=0}^{D_B} H_B(D) \tag{5.19}$$

In order to achieve the equalization of the histogram, ideally, the target histogram should be a uniform distribution. It means $H(D_B) = \frac{N}{256}$, where N is the total number of pixels in the image and 256 is the total range of grayscale values from 0 to 255. Therefore, from (5.19) the function f can be derived only given the histogram of the original image.

$$\sum_{D=0}^{D_A} H_A(D) = \frac{N \cdot D_B}{256} = \frac{N \cdot f\left(D_A\right)}{256} \tag{5.20}$$

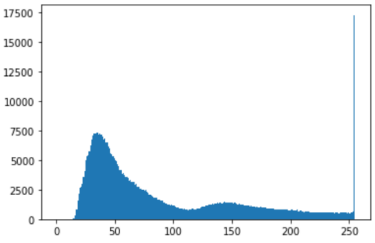
$$f\left(D_A\right) = \frac{256}{N} \sum_{D=0}^{D_A} H_A(D) \tag{5.21}$$

According to (5.21), each grayscale pixel of the original image can be explicitly converted into a new value based on the cumulative density function of the image. In the end, the resulting image will improve the local contrast, which is very useful when dealing with the shades in the camera image. Therefore, in our case, the equalized image $\mathbf{u}'$ is defined in equation 5.22.

$$\mathbf{u}'(D) = \frac{256}{N_x N_y} \sum_{0}^{D} H_{\mathbf{u}}(D) \tag{5.22}$$

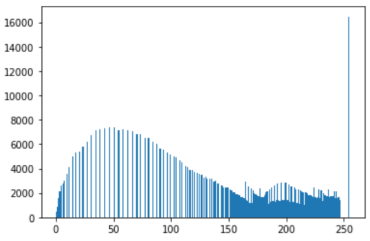
In figure 5.26, we present the comparison of an image before and after the histogram equalization. Meanwhile, we also present their histograms alongside.

(a) The original image.



(b) The histogram of the original image.

(c) The image after histogram equalization.



(d) The corresponding histogram of the processed image.

Figure 5.26: The illustration of histogram equalization. The images are on the left side, while their histograms are on the right side. We can observe that the original image is affected a lot by the shadings of the trees and the buildings. Some details are not visible in the shadow, while those details are mostly revealed since the histogram equalization improves the local contrast on those regions.

From Fig 5.26, it is evident that some details under the shadow of trees and buildings are much clearer in the processed image than the ones in the original image. For example, the left shoulder of the cyclist has almost the same grayscale values as the background building in dark gray, while after the histogram equalization, the edge is more distinct. It is because the histogram equalization improves the local contrast. Similarly, for the vehicle's license plate, which becomes brighter compared to the surrounding dark region. On the other hand, the shadow of the cyclist on the road is weaker, which means that this type of invalid edge will play a more negligible role in the objective function. In

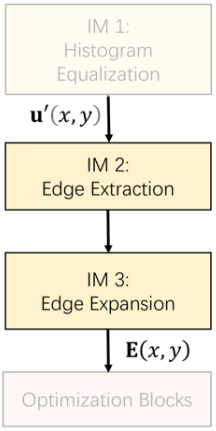


Figure 5.27: The two blocks of IM 2 edge extraction and IM 3 edge expansion.

general, the histogram equalization enhances the regions with low contrast, providing more apparent edges, while it reduces the importance of the firm edges, which are often caused by the sunlight, rather than the natural edges of the objects.

5.3.2 Edge Extraction and Edge Expansion

After the histogram equalization, the next two steps are edge extraction and edge expansion. We borrow the idea of edge expansion that is used in depth discontinuity method, which is introduced in chapter 4.2. The blocks in image 5.27 show the changes of the input equalized image. The output is the edge map of the camera image. In the following section, we will explain the detailed procedures.

Before introducing the edge detector used in this image processing step, it is necessary to review the edge detector we used for panoramic images created based on LiDAR data. The edges of the panoramic images, no matter depth, reflectivity, or object maps, provide the "actual" edges. These edges are generated by the depth discontinuities, the rapid changes in laser reflectivities, or the contour lines of the near field objects. For those edge alignment tasks, the more precise the edge detector used, the better results are presented. Therefore, the Canny edge detector is suitable, compared to the purely kernel-based methods, like the Sobel kernel.

In the previous chapter of panoramic image edge extraction, we used the Canny operator for further extraction to ensure accurate extraction of the significant edges of the object. In this section, because the object boundary is not directly related to the strength of the edge of the grayscale image, we want to retain all possible edges for subsequent matching, so the Sobel kernel is used in the edge extraction process. Although the traditional algorithm is not easy to accurately find the edge of the object in the camera image, its advantages such as high efficiency, no need for a large number of training sets, or complex deep neural network make it still very suitable for the task of rough boundary extraction for extrinsic calibration. The horizontal and vertical 3×3

Sobel kernels are defined in (5.23)

$$\begin{cases} S_v = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \text{ vertical edge detector} \\ S_h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \text{ horizontal edge detector} \end{cases} \tag{5.23}$$

The convolution with these kernels will provide negative values, indicating the direction of the edge. In the edge map that will be used for edge alignment, the direction of the edge is unnecessary. Therefore, another step is required to eliminate the directions and combine the vertical and horizontal edge information, shown in 5.24.

$$\mathbf{E} = \sqrt{(\mathbf{u} * S_v)^2 + (\mathbf{u} * S_h)^2} \tag{5.24}$$

The edge map E is the result of Sobel edge detector, combining 2 edge maps from perpendicular directions. Note that the square root and squares are element-wise calculation.

After the histogram equalization, we assume that all the edges from the intensity image are able to be extracted, including both the right and wrong edges. Some extreme cases, for example, the background and the foreground objects have exactly the same grayscale values, cannot be solved entirely. However, these types of edges take only a tiny portion of all edges. It is reasonable to ignore them. The results of the edge extraction of Sobel kernel before and after histogram equalization is presented in figure 5.28. The Sobel kernel successfully extracts the edges of the intensity image. The choice of the kernel is trivial. Other simple edge detectors can also be used to extract the edges roughly. The results of the edge map prove that the histogram equalization is effective on contrast improvement. The region inside the red dashed box clearly show the improvement we discussed before. The reflective band on the traffic cone, the cyclist's left shoulder, and the car license plate on the right are also revealed after the histogram equalization.

Similar to the idea used in depth discontinuity formulation in 4.2, to punish the points that locate far to the edges and reward the points close to the edge on the intensity image, it is necessary to expand the current edges wider to cover more areas. The closer to the edges, the higher the values. The highest values are still assigned to the location of the original edge. Therefore, using a Gaussian kernel to smooth the edge map is adequate for the edge expansion. The expanded image will be used in further alignment.

(a) The edge map of the original image.

(b) The edge map after the histogram equalization.

Figure 5.28: A comparison of edge maps extracted by Sobel kernel before and after histogram equalization on the original intensity image.

5.4 Objective Function and Optimization

In this section, we will concentrate on the design of the objective function. Since if we assure that the ground truth matches the global optimum perfectly with the smooth shape of the curve, solving the nearly concave objective function is trivial. Combining all the features we collected in previous sections, we present our orange blocks from the main pipeline here in Fig 5.29.

5.4.1 Design of the Objective Function

After obtaining the dense maps of depth maps and LiDAR reflectivity, we can extract edges to match the edge of the intensity image. However, it is worth noting that although the edge on the intensity image can represent the edge of the object to some extent, its gradient value (edge intensity) is caused by the difference of the grayscale image. In other words, the actual boundary of an object may not be very intensive on the edge of the intensity image. Similarly, the difference in depth on a depth map only provides information about the boundary, but it does not mean that the boundary is as strong on the edge of the intensity image. Overall, although we can collect the intensity images, the edges of depth maps, reflectivity maps, and the intensity of edges, it does not mean that the strong boundary on the depth map matches the firm boundary of the intensity image. A simple example in Fig 5.28 is the combination of the edge and background of a rider's upper left body in an intensity image, where the value of the edge is expected

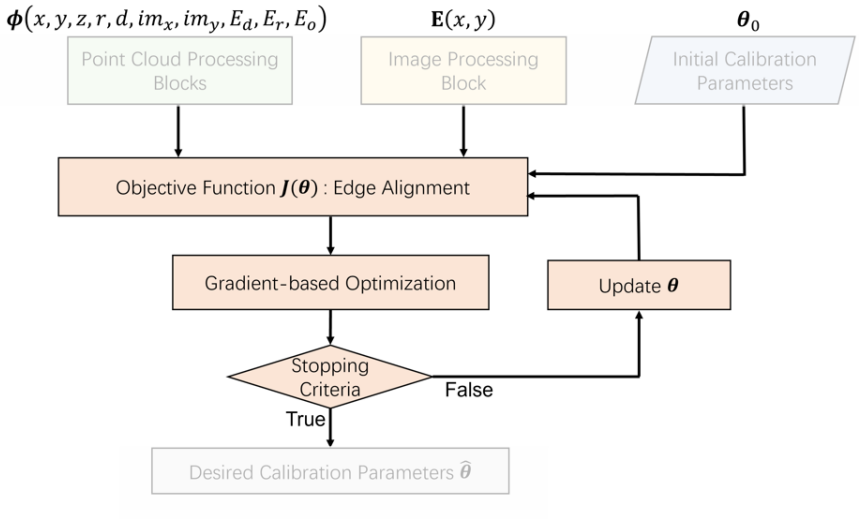


Figure 5.29: The sub-block about cost function and optimization. The input edge image, point cloud with multiple features and initial guess of the calibration parameters are included.

to be lower. However, this boundary is evident on the depth map. Direct edge matching of this type of boundary is complex in the target equation because their product is not necessarily large. If there is a more pronounced boundary on the intensity image at this point in the vicinity, the product of the depth map boundary and this firm boundary may be larger than the product of the depth map boundary and the correct weak boundary, resulting in a local optimum on the objective function.

In order to avoid this type of mismatch, we adopted histogram equalization to reduce the effect of solid edges. Furthermore, using Canny edge detection in a way avoids this as well by assigning binary edges. It weakens the unclear relationship between the strength of depth/reflectivity edges and the grayscale edges. Therefore, the multiplication of image edge values and mixed edge values is better than directly using intensity values to multiply the depth discontinuity values. Because if an edge point is classified as the edge in all three Canny maps, it is highly possible that it will correspondingly match to the edge of the intensity image.

To further ensure the true extrinsic calibration parameters providing global optimum, we also utilize a factor to indicate the percentage of "correctly matched" points. Assume there are in total N points in the point cloud. Among them, N_e points are edge points selected by the mixed Canny edge maps. If an edge point is projected onto the intensity pixel edge of the camera image, the counter N_m, which stands for the number of matched points, will add 1. This ratio is mathematically defined as $\frac{N_m}{N_e}$. We can also interpret this term as precision. Assume all the edges larger than a certain threshold in intensity image are labeled as *True* edges, and the rest are *False*. In this sense, our goal is to maximize the number of estimated laser edge points that collide with the edge pixels of the intensity image. According to Canny edge maps of three features, the edge point should at least carry the edge value larger or equal to 1. We can also label all the points with 0 edge value as non-edge points and vice versa. Considering edge detection as a classification problem, we can draw a confusion matrix to explain the indication of this

ratio.

Points \ Pixels	Edge	Non-edge
Edge	**TP**	**FP**
Non-edge	**FN**	**TN**

Table 5.2: A confusion matrix on edge classification.

In the table above, the T and F mean true and false, while P and N are positive and negative. The true positive indicates that our estimated edge points match with the edge pixels when they are projected on the image plane (which is N_m). Our tuning ratio $\frac{N_m}{N_e}$ in fact equals to the definition of precision shown in (5.25).

$$Precision = \frac{TP}{TP + FP} = \frac{N_m}{N_e} \tag{5.25}$$

If the assumption of edge alignment holds strong, ideally, the correct calibration parameters will result in the highest ratio of $\frac{N_m}{N_e}$. By also coarsely identifying edge/non-edge to intensity image, the influence of the actual value of the edge is further reduced. The form of the objective function is determined as

$$J(\theta) = \frac{N_m}{N_e} \sum_{i=1}^{N} P_i(\theta) \cdot E_i \tag{5.26}$$

The $P_i(\theta)$ is the corresponding edge value of the projected point p_i. The edge information of point i is E_i, where $E_i = 0, 1, 2, 3$ according to the mixed edge map. The strongest edge present on all three maps has the highest weight, while the non-edge points do not participate in the process.

In the whole pipeline, we aim at promoting the edge alignment of the camera image and LiDAR point cloud. In the beginning sections, we focus on exploiting features further on point cloud, other than only depth information; Later on, we discuss the problem of edge alignment function, especially the fact that the edges may be matched, but not their intensities. Therefore we apply Canny edge detection on the edge of the point cloud, while on the other hand design the percentage of the matched edge points. In the following section, we will explore how to solve the optimization problem in the 6 dimension solution space.

5.4.2 Optimization

To successfully optimize the target function (5.26), common choices include the gradient-based or Hessian-based optimizer. In our case, the choice is not particularly important since we already add features to improve the smoothness of the function. Then, a typical choice can be the Barzilai and Borwein gradient ascent method [2].

The non-trivial part of using gradient ascent method is the gradient is hard to be derived through an analytical manner, since the statistical counts and indirect projection features cannot be straightly related to the calibration parameters in the form of

functions. Therefore, the feasible approach is to use the numerical derivation of the gradient.

$$\mathbf{G} = \nabla J(\boldsymbol{\theta}) = \frac{J(\boldsymbol{\theta} + \Delta\mathbf{h}) - J(\boldsymbol{\theta} - \Delta\mathbf{h})}{2 \cdot \Delta\mathbf{h}} \tag{5.27}$$

The numerical derivatives can be calculated given a small parameter $\Delta\mathbf{h}$. Using Barzilai and Borwein's formulation that already introduced in section 4.1, the update equation of the gradient ascent method is defined as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \gamma_k \frac{\mathbf{G}_k}{\|\mathbf{G}_k\|} \tag{5.28}$$

where k indicates the k-th step in the iteration. The adaptive step size is also defined as

$$\gamma_k = \frac{\boldsymbol{s}_k^\top \boldsymbol{s}_k}{\boldsymbol{s}_k^\top \boldsymbol{g}_k} \tag{5.29}$$

where $\boldsymbol{s}_k = \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$ and $\boldsymbol{g}_k = \mathbf{G}_k - \mathbf{G}_{k-1}$.

Algorithm 6: The proposed MulFEA algorithm.

Input: Point cloud $\boldsymbol{\psi}$, image $\mathbf{u}$, initial extrinsic parameter $\boldsymbol{\theta}_0$
Output: The estimated $\hat{\boldsymbol{\theta}}$

$\boldsymbol{\psi}_1 \leftarrow$ Cylindrical projection of $\boldsymbol{\psi}$ (5.4).
$\boldsymbol{\psi}_2 \leftarrow$ RANSAC plane segmentation of $\boldsymbol{\psi}_1$ as Algorithm 3.
$\boldsymbol{\psi}_3 \leftarrow$ DBSCAN clustering as Algorithm 4.
Calculate centroids d_k of k clusters (5.6).
Create object features (5.7).
Object mask $H_{obj} \leftarrow$ morphological closing (5.8) (5.9) (5.10).
Dense maps $\boldsymbol{\psi}_d, \boldsymbol{\psi}_r, \boldsymbol{\psi}_o \leftarrow$ Dense map completion of 3 features (5.11);
$E_d, E_r, E_o \leftarrow$ Canny edge detection of $\boldsymbol{\psi}_d, \boldsymbol{\psi}_r, \boldsymbol{\psi}_o$ as Algorithm 5.
Multi-feature edge map $E_{pc} \leftarrow$ edge mixture 5.16.
$\boldsymbol{\psi}_4 \leftarrow$ edge feature E_{pc} extended to point cloud.
$\mathbf{u}' \leftarrow$ histogram equalization of $\mathbf{u}$ (5.22).
Edge map $E \leftarrow$ Sobel edge detector of $\mathbf{u}'$ (5.24).
while $\|\nabla\boldsymbol{\theta}_t\| > \epsilon$ **do**
    Calculate $\boldsymbol{J}(\theta)$ (5.26);
    Calculate $\boldsymbol{G}_t$ (5.27);
    Calculate γ_t (5.29);
    Update $\boldsymbol{\theta}_t$ (5.28);
end

5.5 Conclusion

In this chapter, the proposed multi-feature workflow from Fig 5.1 is covered. We have discussed the separate branches of point cloud processing (in green), image processing (in yellow), and function design and optimization (in orange). To illustrate the pipeline, we show the image again in Fig 5.30. The contributions of MulFEA are listed below:

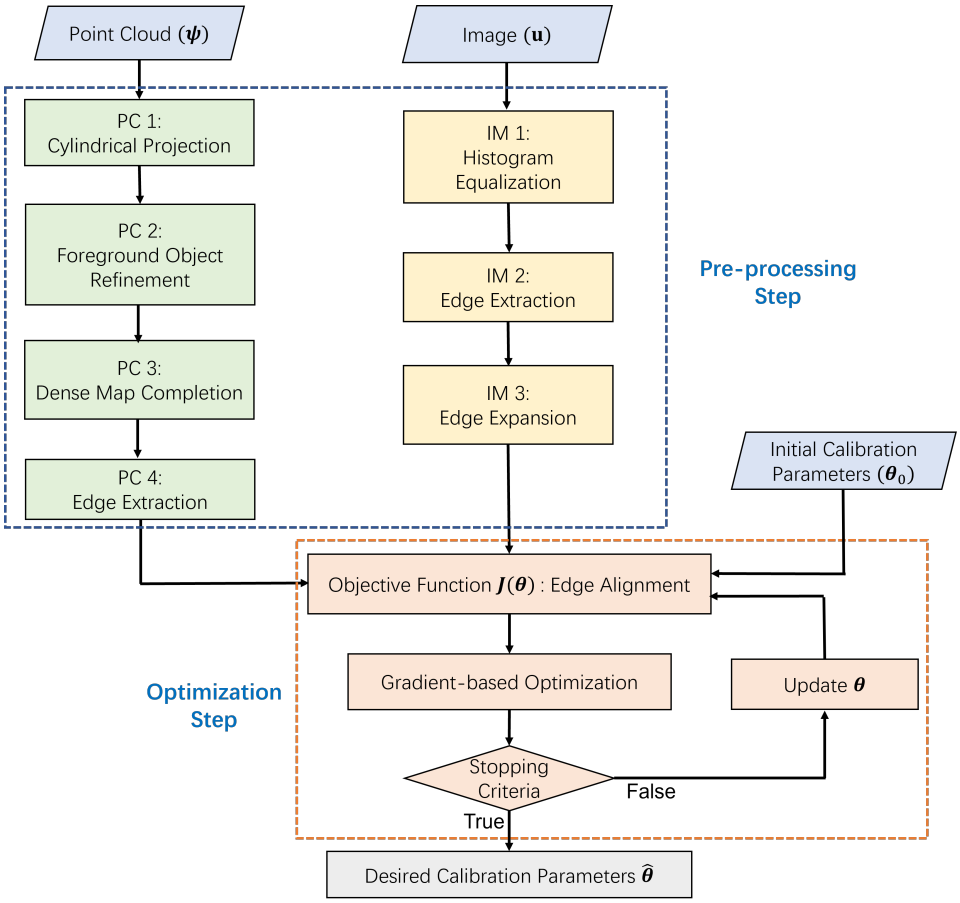


Figure 5.30: The full workflow of MulFEA. The blocks in blue at the top of the graph are the input data to be processed. The green and yellow boxes are the preprocessing and feature extraction steps for LiDAR and camera, respectively. The orange blocks are the optimization procedure, which runs after the data processing. The final gray block is the output of the optimization, which is the desired extrinsic calibration parameters.

- The cylindrical projection enables the isolated processing of laser points. It converts the 3D problem into 2D, where we have more mature techniques on 2D image processing rather than 3D point cloud. More importantly, it allows us to create a direct connection between 3D points and their corresponding 2D coordinates in the panoramic image. All the features extracted from the panoramic image, therefore, directly correspond to the 3D points.
- The foreground object refinement uses an effective method in order to achieve the occlusion-free projection since the occlusion degrades the shapes of the objects drastically. We firstly separate the plane points so that they will not influence the clustering. Then we apply the density-based clustering method to perform the separation of foreground and background points. To figure out precisely which points are occluded, we use the traditional methods from morphological image processing and successfully reduce the occlusion effect.
- We adopt the Canny edge detector to extract the edge features from the panoramic

image. The returning binary edge label reduces the weak correlation between the intensity of the features from the point cloud and the grayscale image difference from the camera.

- The histogram equalization achieves a similar goal by improving the local contrast, reducing the effect of solid edges, and enlarging the importance of weak edges.
- Aside from the direct multiplication for edge alignment, we also multiply another factor, defined as the precision metric from the concept of the confusion matrix. It ensures the smoothness of the cost function from the binary classification perspective.

6 Experiments and Results

In this chapter, our whole pipeline will be tested based on the LiDAR scans, and images from KITTI dataset [14]. Before presenting our results, we will briefly introduce the KITTI dataset and its data format. We will also include the raw data processing in a programmatic approach followed by the experiments. The ground truth of the calibration parameters is available from the dataset. We select several typical scenarios to compare the shape of the curve on different calibration parameters of the objective function. To illustrate the influence of the amount of data, we extend our method to a multi-frame variant that exploits results from multiple frames.

6.1 KITTI Dataset Explanation

KITTI dataset is one of the most popular benchmarks for autonomous driving. It includes several aspects, mainly focusing on different types of computer vision tasks in an autonomous driving environment. They are stereo vision, odometry, optical flows and object classification in 2012 [15] and 2015 [29], respectively. There are also road benchmark [13] and raw data [14]. In our experiment, we adopt only the raw data from the camera and LiDAR to perform the validation.

The system setup is presented in figure 6.1. It includes 4 optical cameras (00 and 01 are gray-scale cameras while 02 and 03 are color cameras), 1 LiDAR (Velodyne-HDL64 [20]), a GPS/IMU inertial navigation system and 4 varifocal lenses on board. In our experiment, we exploit images from camera 00 and laser scans from LiDAR. Both of these two sensors are placed on the middle top of the vehicle, which would make the assumption of edge alignment stronger. The time synchronization is achieved by a

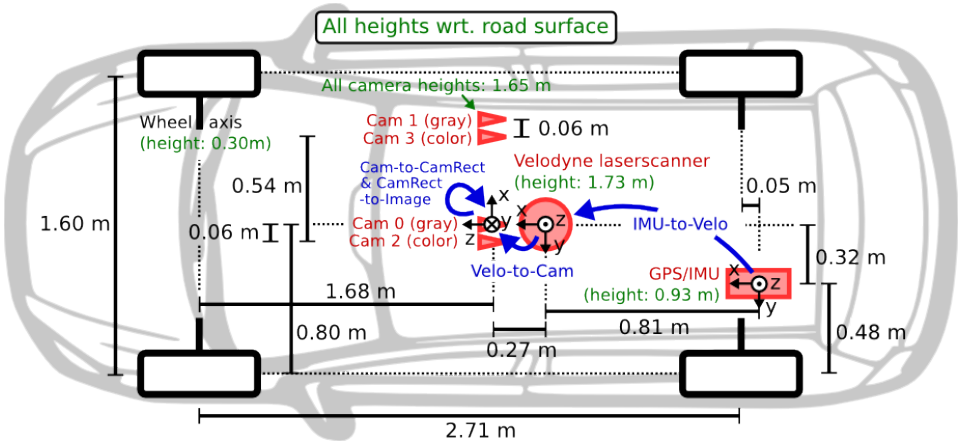


Figure 6.1: The layout of sensor placements [14].

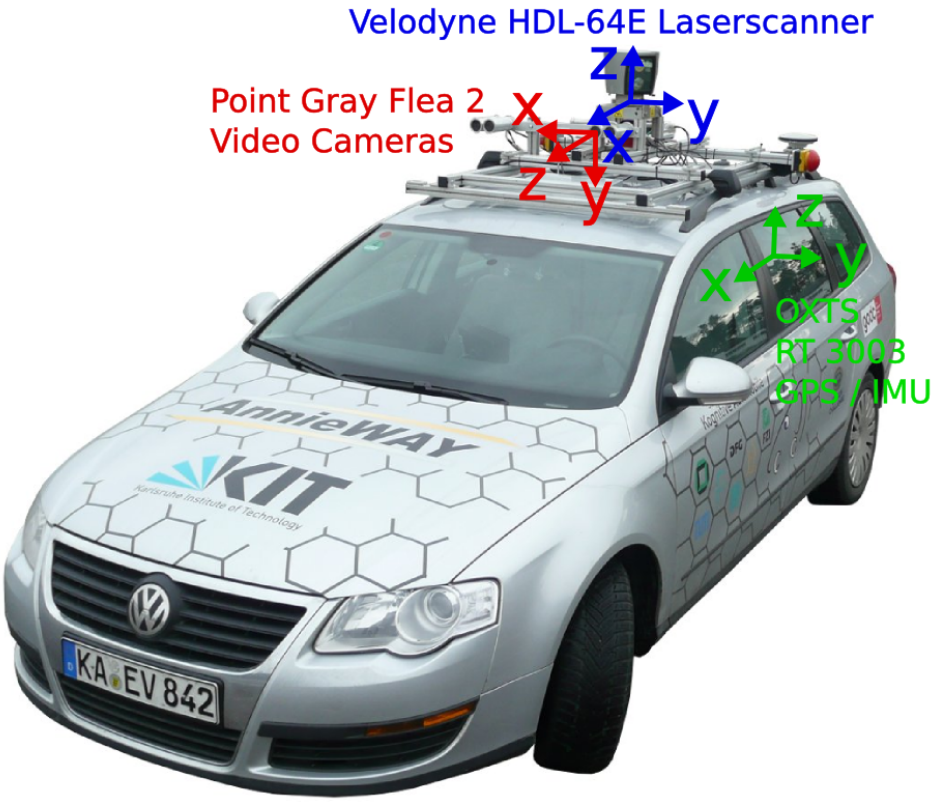


Figure 6.2: The fully equipped vehicle [14].

trigger-based method. Each second, the scanning module performs 10 spins. Every time it passes its forward direction, the shutter of the camera is triggered. The image of the geared-up vehicle is shown in Fig 6.2.

The raw images are all in the same size of 375×1242 pixels. The rectification is performed given the distortion parameters. The built-in GPS module of Velodyne-64HDL fixes the raw laser scans from LiDAR output with basic motion adjustment. This adjustment avoids the effect of ego-motion during each scan, which can cause errors in the final point cloud. Usually, the raw image and the raw point cloud from KITTI raw data are ready to use with no artificial errors.

According to our pipeline, to efficiently process the whole data, we append each point with all the needed information. Therefore when calculating the cost function, the time complexity will stick to $\mathcal{O}(n)$. This is the best example of point cloud processing separation and feature collection. Since the whole point cloud is considered, all the features are attached with every point after the preprocessing. The optimization program will simply read the features in the optimization stage rather than repeat the processing again due to small changes in the calibration parameters.

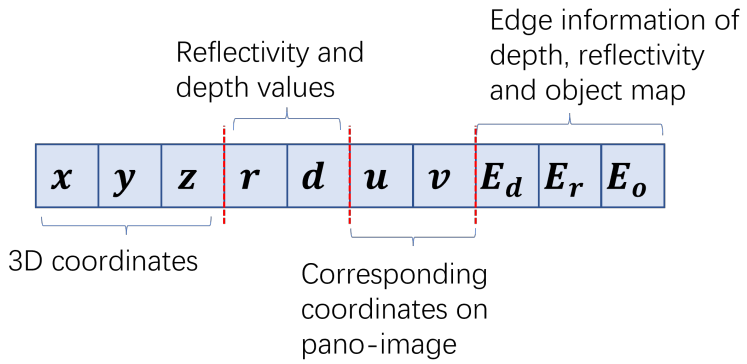


Figure 6.3: The appended row of a single laser point. In total an $n \times 11$ matrix will be created to store the whole point cloud with features.

6.2 Typical Scenario Trials

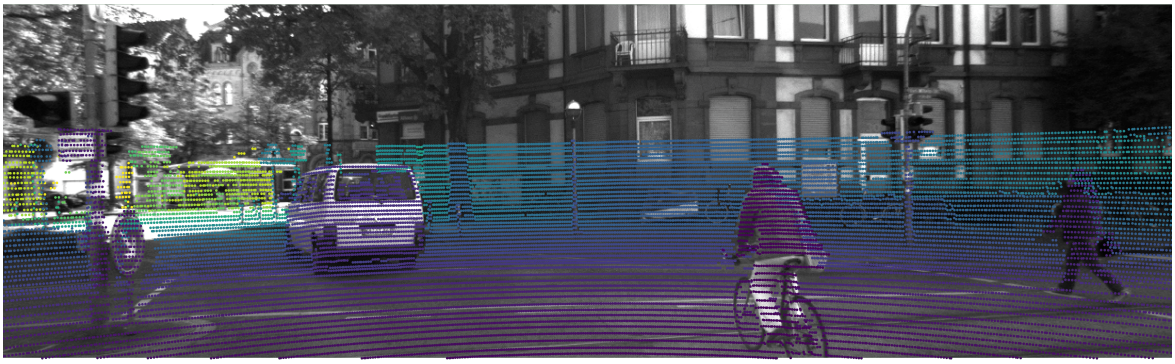
In order to verify the performance of our proposed method, we select several different scenarios with various environments on the road as test data. We investigate the objective functions of 3 state-of-the-art methods discussed in Chapter 4 compared with our own cost function. The exemplary shapes of objective functions on each parameter with ground truth in the middle are presented. For rotational parameters, the range is from -0.3 to 0.3 rads. The interval for translational parameters is from -0.3 to 0.3 meters. In each plot, the x axis shows the interval of different calibration parameters, while the y axis is the score on each objective function. We make a table of those cost functions in order to present the results afterward more conveniently, as seen in table 6.1.

Name	**Form**	**Objective**
Mutual Information Formulation [32]	$\mathcal{F}(\boldsymbol{\theta}) = \mathrm{MI}(X, Y; \boldsymbol{\theta})$ (4.9)	Maximize
Depth Discontinuity Method [25]	$J_{\boldsymbol{\theta}} = \sum X_i \cdot D_i$ (4.16)	Maximize
Joint Calibration-Fusion Method [5]	$\mathcal{F}(\boldsymbol{\theta}) = \sum \frac{\sum_n w_{n,k} \cdot \lvert[\nabla_k \phi_{\boldsymbol{\theta}}]_n\rvert}{(\sum_n w_{n,k}) \cdot (\sum_n \lvert[\nabla_k \phi_{\boldsymbol{\theta}}]_n\rvert)}$ (4.21)	Minimize
Proposed MulFEA	$J(\boldsymbol{\theta}) = \frac{N_m}{N_e} \sum_{i=1}^{N} P_i(\theta) \cdot E_i$ (5.26)	Maximize

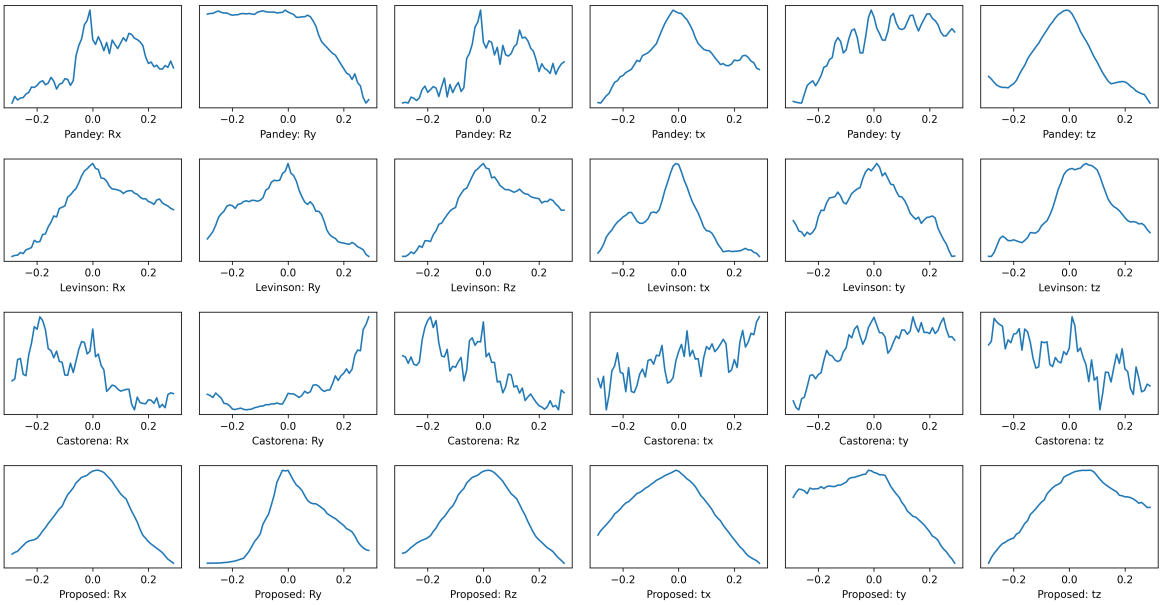
Table 6.1: Table of four objective functions.

6.2.1 Case 1: Urban Road

The first case is a typical city road scenario. There are pedestrians crossing the road, the vehicle in the middle of FOV, the cyclist on the road, the tram rail, the traffic lights and signs, and buildings in the background. It is ideal for clustering since all the major objects are fully separated from each other. The mutual information formulation provides local maxima around the ground truth values. Still, it is generally non-smooth for this type of single-frame trial. The depth discontinuity method is better, since the global optimum is more distinct. But there are still some tiny local gaps that gradient optimizer may get stuck. It is reasonable since this method is for online small shifts fixation rather than a global search of the optimal value. The joint calibration-fusion



(a) The original image with correctly projected LiDAR points.



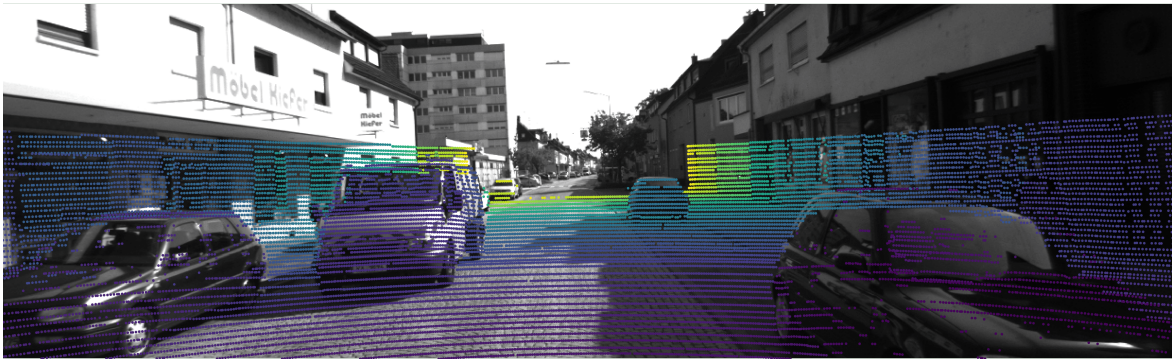
(b) The plots on 4 methods, in 6-DOF solution space.

Figure 6.4: Case 1 shows a typical example of urban road scenario. The experiment vehicle is crossing a junction of tramlines. There are pedestrian, vehicles and other traffic related objects within the field of view.

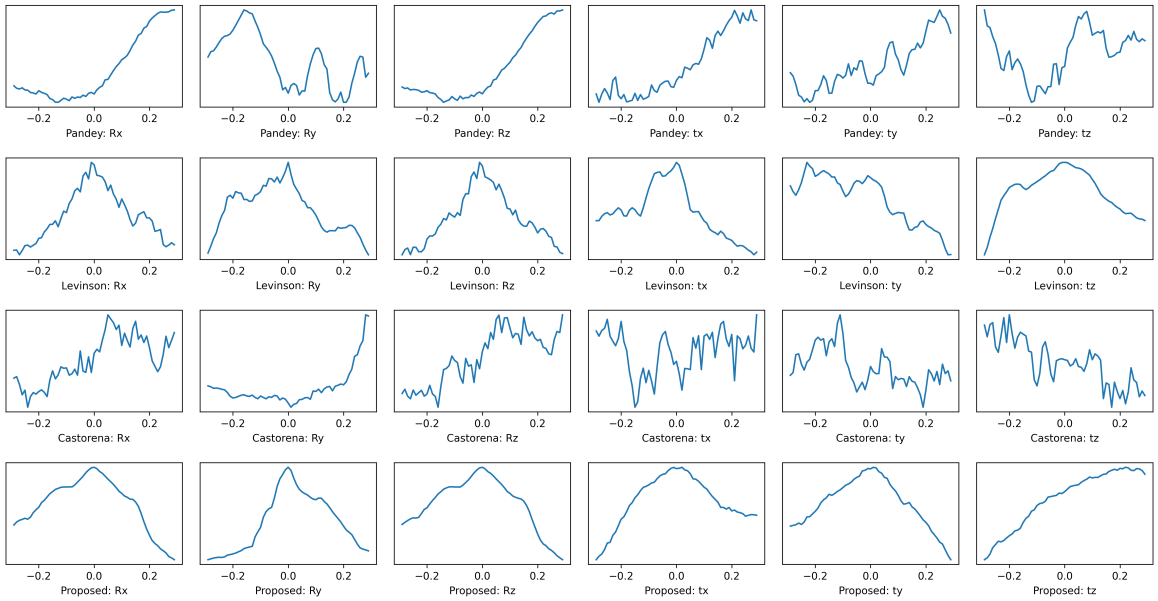
method has some limitations as well, since it overlooks the occlusion effect. The shape of the cost function is not smooth. In the original paper [5], the cost function is solved by simulated annealing-based method rather than gradient-based optimization. This might reduce the influence of local optima. However, the global optimum is not located in the middle of the curves, which will degrade the results. In the proposed MulFEA method, the vertical edges are considered, and the histogram equalization for image edge enhancement is also used. The precision factor also tunes the curves better. Therefore, it performs pretty well.

6.2.2 Case 2: Neighbourhood Alley

The second example is a narrow straight alley with parked cars along the road. In this case, no lanes on the road nor other traffic signs. The shadow of the building on the right leaves a hard edge in the middle of the image. Due to the electroplated surface of the vehicles, the mirror reflection happens on the near field cars. Therefore a large portion of points is missing, raising the difficulties on edge detection. The image and the results can be seen from Fig 6.5. The existence of shadow influences the results of



(a) The original image with correctly projected LiDAR points.



(b) The plots on 4 methods, in 6-DOF solution space.

Figure 6.5: Case 2 shows a small alleyway without any high reflective traffic objects. The direct sunlight brings troubles on calibration algorithms.

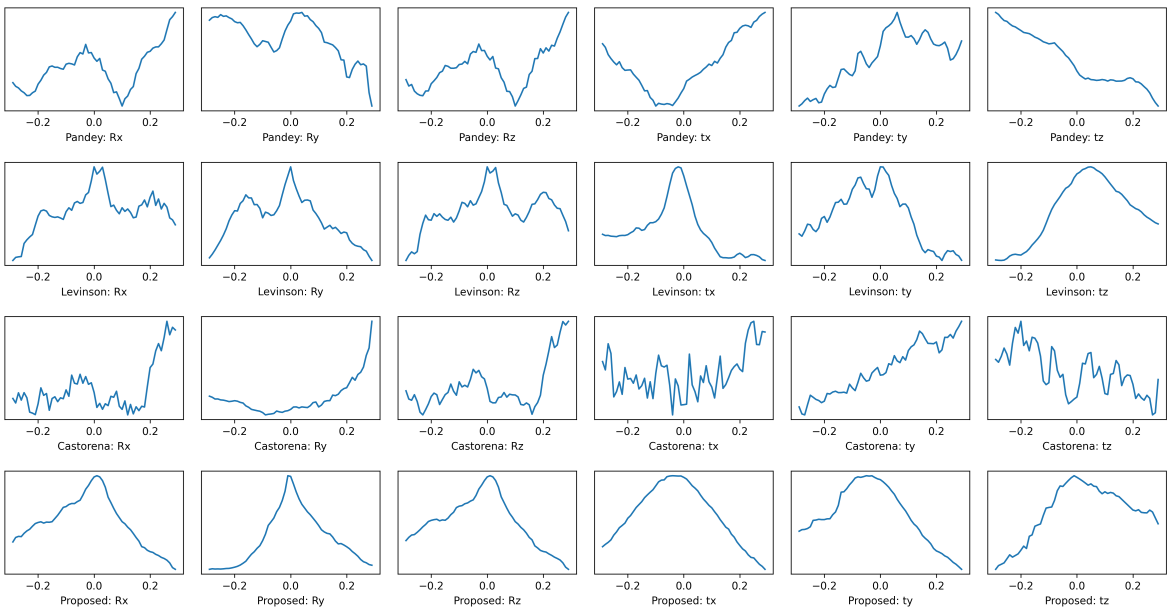
mutual information formulation heavily, since the intensity and reflectivity matching is no longer the same inside and outside of the shading area. The loss of object points due to the mirror reflection on the surface of near-field vehicles also degrades the results of all three edge alignment methods, but generally the proposed method is more stable

than the rest. Nevertheless, it is possible that the gradient-based method will lead to a biased estimate on the final results of the estimation of calibration parameters since the shape of curves, especially on t_z, is not smooth.

6.2.3 Case 3: Highway

The third example is from a main road (highway) with all kinds of vehicles around. There are clear lanes and traffic signs in the distance. Similarly, the vehicle on the left is too near to the LiDAR, many laser beams that should illuminate the vehicle's body are reflected elsewhere that LiDAR cannot receive. The densely placed vehicles make the clustering hard to perform successful partitioning. The shadow of the sun is another problem that often happens. In Fig 6.6, we show the image with laser points and the results. In these very complicated scenarios with shadow, the performance of the mutual

(a) The original image with correctly projected LiDAR points.



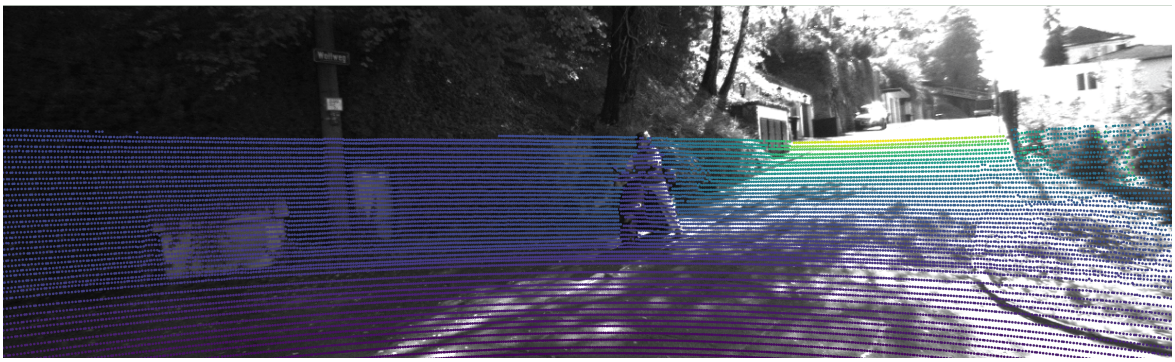
(b) The plots on 4 methods, in 6-DOF solution space.

Figure 6.6: Case 3 presents an example of highway scenario. LiDAR cannot see through the near-field vehicles around, therefore providing little information from farther objects.

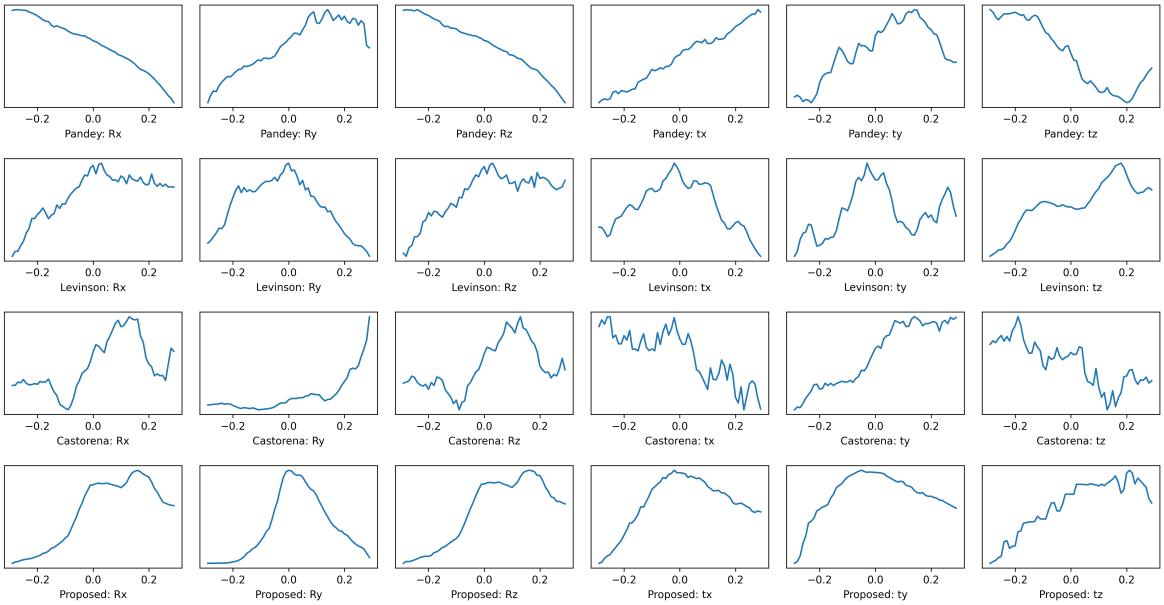
information method is no longer robust. For t_x and t_z, the assumption of "Maximum mutual information of intensity and reflectivity leads to correct calibration parameters" does not hold in this scenario. Small local optima are still the problem for the depth discontinuity method. The proposed method is smoother with fewer local gaps but still challenged.

6.2.4 Case 4: Suburb Area

The fourth experiment uses a very harsh environment in the suburban area. The experiment vehicle drove into a local neighborhood and came across a ramp. The plane segmentation is not able to extract both the flat plane and the ramp. Some plane points participate in the clustering, leading to awful object clustering results. The motorbike rider is completely hidden under the shade of trees. More than half of the image is corrupted by the shadow. The wire pole is close to the wall, with little depth discontinuity information that can be exploited. The clustering cannot separate them either. In the FOV, there is few valid objects for edge alignment. The results are illustrated in Fig 6.7. The mutual information based method completely fails in this scenario. The correlation between camera intensity and laser reflectivity is no longer convincing due to the shadows. The edge alignment method needs valid objects in the FOV, while little exists in this case. For the proposed method, both rotational and translational parameters are affected. Some local maxima existing in R_x and R_z will worsen the result of the gradient-based optimization.



(a) The original image with correctly projected LiDAR points.



(b) The plots on four methods, in 6-DOF solution space.

Figure 6.7: Case 4 provides a very harsh environment in suburb area for calibration algorithms. Scarce objects provide valid edges, while the shadow influences more than half of the camera image. On the wall on the left there are also vines with leaves, providing edges on the intensity image while nothing on the point cloud.

6.3 Multi-frame Enhancement

In this section, the experiment is carried out by incorporating multiple frames into our cost function. Since our method does not consider motion or information continuity, the frames are not necessarily to be placed as a sequence with any particular orders. Generally speaking, with more data provided, the assumption of edge alignment holds stronger. Although from our single frame results, the performance is outstanding. In more complicated scenarios, i.e., with shadows and reflections heavily involved, the performance is degraded.

A way to improve the results at the cost of time and storage spaces is by adding more frames into the cost function. Therefore, the objective function (5.26) can be slightly

deformed as shown in (6.1).

$$J'(\theta) = \sum_{j=1}^{M} J(\theta) = \sum_{j=1}^{M} \left(\frac{N_{jm}}{N_{je}} \sum_{i=1}^{N_j} P_i(\theta) \cdot E_i \right) \tag{6.1}$$

where the N_j denotes the total number at $j-th$ frame. The original $J(\theta)$ can be referred to the single-frame cost function (5.26).

In this experiment, we compare the single frame 6-DOF curves with a 20-frame result. We choose 20 continuous images from the KITTI dataset for multi-frame results, and the last trial is for the single-frame results. Fig 6.8 illustrates the normalized results in the same charts.

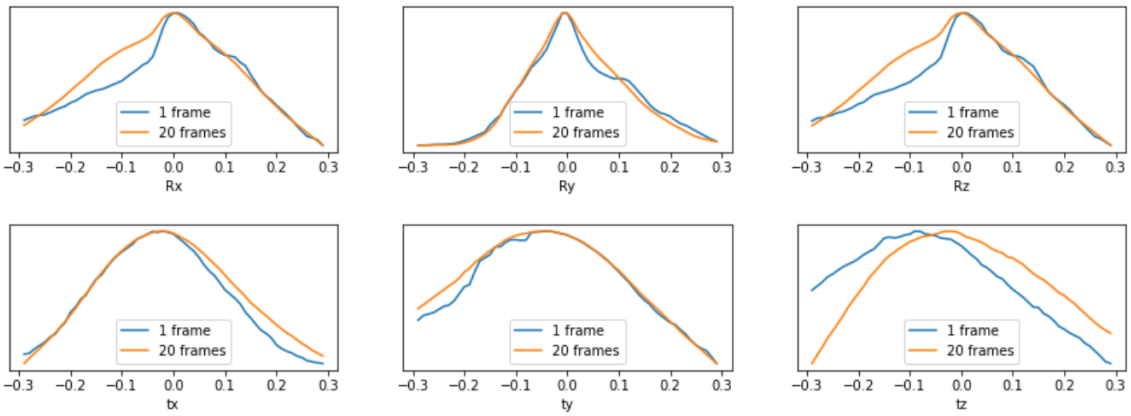


Figure 6.8: The results of a single frame trial and a 20-frame trial. The values are normalized to keep them both visible.

It can be easily observed that the multi-frame results provide smoother shapes on the solution space of the cost function. For translational parameter t_z, the bias in the single frame can be canceled since there are results from other frames that have less bias. Nevertheless, the trade-off between computational power and accuracy or smoothness of the objective function always exists.

7 Conclusion

In this thesis, we discussed the development and some state-of-the-art methods of camera-LiDAR calibration problem. Based on the drawbacks of those methods, we developed a novel pipeline named MulFEA to solve the problem via multi-feature edge alignment. By considering multiple levels of data pre-processing, the shape and smoothness of the objective function are further improved, which leads to more reliable estimates on extrinsic parameters. This chapter describes the main findings in the thesis, the limitations of our pipeline and provides suggestions for future work.

7.1 Summary and Contributions

This project aims at addressing the camera-LiDAR extrinsic calibration problem through edge alignment. Our MulFEA pipeline utilizes a cylindrical projection technique to convert the 3D LiDAR point cloud into a 2D panoramic image, which simplifies the point cloud processing. To avoid the occlusion effect on the edge of foreground objects caused by the background environment, we first use RANSAC and DBSCAN to remove plane points, select object points in 3D. Then we adopt a morphological closing step occluding points in the 2D panoramic image. After that, a dense map completion is performed based on depth, reflectivity, and foreground-object features. The Canny edge detector then extracts the edge information of the point cloud. For image processing, we use histogram equalization to reduce the effect of shades and shadows. We also employ a Gaussian-kernel-based edge expansion method to reward the points that locate close to the edge. In the cost function, we add a statistical parameter to adaptively ensure the correctness of calibration, given the ratio of successful edge-edge matching. In our experiments, the KITTI dataset is used to verify our results, compared to other state-of-the-art methods. The results show that in the general sense, our multi-feature edge alignment method tends to have smoother curves and less local optima.

The major contributions of this work are listed as follows:

- A pipeline of cylindrical projection combined with dense map completion is proposed for point cloud processing. It converts the 3D sparse point cloud into a 2D panoramic image, simplifying the problem's dimension and reducing the calculation redundancy.
- A mixed map extracted from the multi-feature panoramic image is developed. It gathers the advantages of depth, reflectivity, and object information, providing an edge map for edge alignment with richer information. It provides more similarities compared to the edge map of the camera image.
- An edge alignment technique called MulFEA is proposed. There are two key novelties in our proposed solution. The first one is the quarter-nary property of the

mixed edge. It indirectly reflects the validity of edges, omitting the weak correlation between depth/reflectivity edges and image intensity edges, promoting the cost function's smoothness. Secondly, by also considering a precision parameter, our cost function reduces the existence of local optima.

7.2 Limitations

Despite the contributions mentioned above, the MulFEA approach presented in this thesis still has some limitations. First, the limitation of edge alignment assumption itself. For example, when cameras and LiDAR are placed far away from each other (for example, one on the left edge of a vehicle and the other on the right edge), they might have different perspectives when looking at some near-field objects. This may lead to errors in the overall idea of edge matching, even if the fundamental assumption is no longer valid. Secondly, the depth completion algorithm included in this thesis is based on the TV norm in traditional image processing. When point clouds disappear on the road at a remote distance, the accuracy of depth completion in this area is much less. Compared with more advanced machine-learning-based algorithms, the traditional TV norm still has gaps in depth completion results. The third point is that our ground extraction algorithm can only recognize relatively flat terrain. Ground extraction is less effective when facing curvy ramps or bumpy terrain and will affect subsequent boundary extraction. In addition, if there are few objects in the field of view, clustering and edge matching may also face particular difficulties. Finally, this thesis involves some hyper-parameters, and we have not specifically analyzed their impact during the experiment.

7.3 Future Works

As mentioned in [54], edge alignment can be expanded with semantic information, which provides more reliable and specific details on edge. This can also omit the problem of a weak connection between intensity edges and depth or reflectivity edges. Although using neural networks specifically for extrinsic calibration is costliness, the results of high-level data processing and fusion can be exploited and applied in our MulFEA pipeline. For example, the edge extraction of the image can be replaced by the results of object classification from the image processing and understanding side. The core idea is that pre-processing is also necessary for further fusion and other perception tasks. Therefore, their calculation can also contribute to our calibration step. The point cloud can also be analyzed with advanced methods for semantic separation. For example, in [53] they proposed a neural-network-based object detection pipeline with middle-level sensor fusion. If their intermediate results are fed into our calibration workflow, our calibration algorithm should still work. If all the networks are pre-trained beforehand, the idea is theoretically viable. A simple sketch in figure 7.1 shows our vision on the edge-alignment-based extrinsic calibration methods.

From a long-term perspective, the calibration would become more accessible with a partially known 3D environment when the roads are also incorporated with some intelligent labels. However, without large-scale V2X (Vehicle to Everything) systems

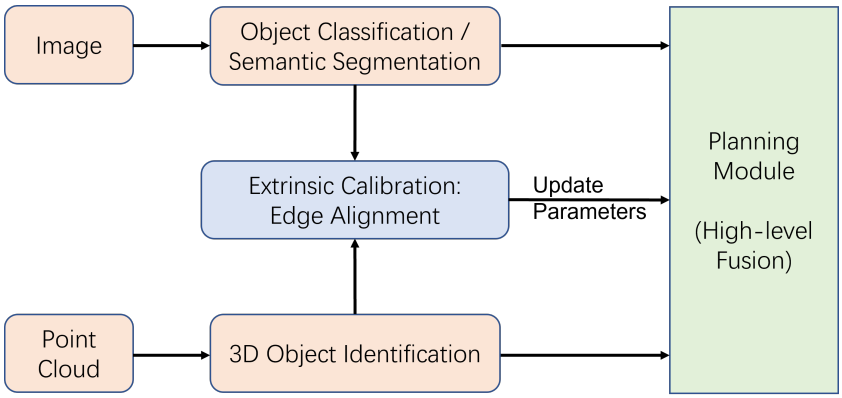


Figure 7.1: An ideal higher-level camera-LiDAR fusion. The middle-stage results can be used to perform the edge alignment calibration algorithms. This module can be activated regularly to update the calibration parameters in case of small drifts while consuming little extra computational power since the major pre-processing is also necessary for the planning and decision-making module.

distributed, this idea is still constrained in the laboratory and testing fields [11].

Bibliography

[1] Hatem Alismail, Douglas Baker, and Brett Browning. Automatic calibration of a range sensor and camera system. 10 2012.

[2] JONATHAN BARZILAI and JONATHAN M. BORWEIN. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 01 1988.

[3] A. Beck and M. Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18(11):2419–2434, 2009.

[4] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[5] J. Castorena, U. S. Kamilov, and P. T. Boufounos. Autocalibration of lidar and optical cameras via edge alignment. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2862–2866, 2016.

[6] TU Delft Circuit and System Group. Adacorsa: Airborne data collection on resilient system architectures. https://cas.tudelft.nl/Research/project.php?id=160.

[7] Demuni De Silva, Jamie Roche, and Ahmet Kondoz. Robust fusion of lidar and wide-angle camera data for autonomous mobile robots. *Sensors*, 18:2730, 08 2018.

[8] Opencv dev team. Camera calibration and 3d reconstruction. https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html.

[9] Wenbo Dong and Volkan Isler. A novel method for the extrinsic calibration of a 2d laser rangefinder and a camera. *IEEE Sensors Journal*, 18(10):4200–4211, May 2018.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[11] C. Fang, Shuai Ding, Zilong Dong, Honghua Li, Siyu Zhu, and P. Tan. Single-shot is enough: Panoramic infrastructure based calibration of multiple cameras and 3d lidars. *ArXiv*, abs/2103.12941, 2021.

[12] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[13] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[16] Andreas Geiger, Frank Moosmann, Ömer Car, and Bernhard Schuster. Automatic camera and range sensor calibration using a single shot. In *2012 IEEE International Conference on Robotics and Automation*, pages 3936–3943, 2012.

[17] Andreas Geiger, Frank Moosmann, Ömer Car, and Bernhard Schuster. Automatic camera and range sensor calibration using a single shot. In *2012 IEEE International Conference on Robotics and Automation*, pages 3936–3943, 2012.

[18] Tom Goldstein and Stanley Osher. The split bregman method for l1-regularized problems. *SIAM J. Imaging Sciences*, 2:323–343, 01 2009.

[19] Xiaojin Gong, Ying Lin, and Jilin Liu. 3d lidar-camera extrinsic calibration using an arbitrary trihedron. *Sensors (Basel, Switzerland)*, 13:1902–18, 02 2013.

[20] Velodyne Lidar Inc. Velodyne hdl-64 high definition real-time 3d lidar. urlhttps://velodynelidar.com/products/hdl-64e/.

[21] Kiyoshi Irie, Masashi Sugiyama, and Masahiro Tomono. Target-less camera-lidar extrinsic calibration using a bagged dependence estimator. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1340–1347, 2016.

[22] Ganesh Iyer, R. Ram, Krishna Jatavallabhula, and Madhava Krishna. Calibnet: Geometrically supervised extrinsic calibration using 3d spatial transformer networks. pages 1110–1117, 10 2018.

[23] EPFL LTS2 Laboratory. Pyunlocbox: Optimization by proximal splitting. https://pyunlocbox.readthedocs.io/en/stable/index.html#.

[24] Henrietta Lengyel and Zsolt Szalay. *The Significance and Effect of the Traffic System Signaling to the Environment, Present and Future Traffic*, pages 847–856. 01 2019.

[25] Jesse Levinson and Sebastian Thrun. Automatic online calibration of cameras and lasers. 06 2013.

[26] You Li and Javier Ibanez-Guzman. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*, 37(4):50–61, Jul 2020.

[27] Tao Ma, Zhizheng Liu, Guohang Yan, and Yikang Li. Crlf: Automatic calibration and refinement based on line feature for lidar and camera in road scenes, 2021.

[28] Frederik Maes, André Collignon, Dirk Vandermeulen, Guy Marchal, and Paul Suetens. Multi-modality image registration by maximization of mutualinformation. volume 16, pages 14–22, 07 1996.

[29] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[30] Omar Montoya, Octavio Icasio, and Joaquín Salas. *COUPLED: Calibration of a LiDAR and Camera Rig Using Automatic Plane Detection*, pages 209–218. 06 2020.

[31] Miguel Ángel Muñoz-Bañón, Francisco A. Candelas, and Fernando Torres. Targetless camera-lidar calibration in unstructured environments. *IEEE Access*, 8:143692–143705, 2020.

[32] Gaurav Pandey, James R. McBride, Silvio Savarese, and Ryan M. Eustice. Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information. In *AAAI*, 2012.

[33] Yoonsu Park, Seokmin Yun, Chee Sun Won, Kyungeun Cho, Kyhyun Um, and Sungdae Sim. Calibration between color camera and 3d lidar instruments with a polygonal planar board. *Sensors*, 14(3):5333–5353, 2014.

[34] Marcelo Pereira, David Silva, Vitor Santos, and Paulo Dias. Self calibration of multiple lidars and cameras on autonomous vehicles. *Robotics and Autonomous Systems*, 83, 05 2016.

[35] Juraj Peršić, Luka Petrović, Ivan Marković, and Ivan Petrović. Spatiotemporal multisensor calibration via gaussian processes moving target tracking. *IEEE Transactions on Robotics*, pages 1–15, 2021.

[36] Cristiano Premebida, Luís Garrote, Alireza Asvadi, A. Ribeiro, and Urbano Nunes. High-resolution lidar-based depth mapping using bilateral filter. pages 2469–2474, 11 2016.

[37] Zoltan Pusztai and Levente Hajder. Accurate calibration of lidar-camera systems using ordinary boxes. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 394–402, 2017.

[38] Sergio A. Rodriguez F., Vincent Fremont, and Philippe Bonnifait. Extrinsic calibration between a multi-layer lidar and a camera. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 214–219, 2008.

[39] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[40] Nick Schneider, Florian Piewak, Christoph Stiller, and Uwe Franke. Regnet: Multimodal sensor registration using deep neural networks. 07 2017.

[41] F. Sigernes. The norusca ii all-sky cameras. http://kho.unis.no/News/NORUSCAII.htm.

[42] Sungdae Sim, Juil Sock, and Kiho Kwak. Indirect correspondence-based robust extrinsic calibration of lidar and camera. *Sensors*, 16:933, 06 2016.

[43] Peter Sturm, Srikumar Ramalingam, Jean-Philippe Tardif, and Simone Gasparini. Camera models and fundamental concepts used in geometric computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6:1–183, 01 2011.

[44] Zachary Taylor, Juan Nieto, and David Johnson. Automatic calibration of multi-modal sensor systems using a gradient orientation measure. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1293–1300, 2013.

[45] OpenCV Development Team. Canny edge detector document. https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html.

[46] Ranjith Unnikrishnan and Martial Hebert. Fast extrinsic calibration of a laser rangefinder to a camera. 01 2005.

[47] Ulla Wandinger. *Introduction to Lidar*, pages 1–18. Springer New York, New York, NY, 2005.

[48] Li Wang, Zhipeng Xiao, Dawei Zhao, Tao Wu, and Bin Dai. Automatic extrinsic calibration of monocular camera and lidar in natural scenes. In *2018 IEEE International Conference on Information and Automation (ICIA)*, pages 997–1002, 2018.

[49] Zhangjing Wang, Yu Wu, and Qingqing Niu. Multi-sensor fusion in automated driving: A survey. *IEEE Access*, 8:2847–2868, 2020.

[50] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[51] Qilong Zhang and R. Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2301–2306 vol.3, 2004.

[52] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22:1330 – 1334, 12 2000.

[53] Xiangmo Zhao, Pengpeng Sun, Zhigang Xu, Haigen Min, and Hongkai Yu. Fusion of 3d lidar and camera data for object detection in autonomous vehicle applications. *IEEE Sensors Journal*, 20(9):4901–4913, 2020.

[54] Yufeng Zhu, Chenghui Li, and Yubo Zhang. Online camera-lidar calibration with sensor semantic information. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4970–4976, 2020.