Multi-Sensor Fusion for Robust Simultaneous Localization and Mapping

Cong Li

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Robotics

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Prof. George A. Kantor
Prof. Michael Kaess
Ming Hsiao

CMU-RI-TR-19-60

August 2019

Abstract

Simultaneous Localization and Mapping (SLAM) consists of the estimation of the state of the robot, and the reconstruction of the surrounding environment simultaneously. Over the last few decades, numerous state-of-the-art SLAM algorithms are proposed and frequently utilized in the robotics community. However, a SLAM algorithm might be fragile or even fail due to imperfect sensor data, uncontrived environments and hardware failures. Furthermore, many approaches sacrifice loop closure and reduce to odometry, which might suffer from accumulated drift. In addition, most of the SLAM systems only provide metric representations for mapping, which is difficult for high-level robot tasks. In this thesis, we first explore the integration of the monocular camera, a Light Detection and Ranging (Lidar) sensor, and an Inertial Measurement Unit (IMU) to achieve the accurate and robust state estimation. We then propose one loop closure module combining camera and Lidar's measurements to correct motion estimation drift. Finally, we propose one 3D semantic occupancy mapping framework integrating monocular vision, Lidar's measurements and our state estimation system. We first test our state estimation system on the publicly available datasets as well as challenging custom collected datasets. Then, a series of experiments are conducted to demonstrate the effectiveness of the loop closure module. Finally, the KITTI odometry dataset is used to demonstrate our 3D semantic occupancy mapping framework. The experimental results indicate that our proposed state estimation system works well in various challenging environments and an accurate large-scale semantic map can be constructed.

Acknowledgements

It would be impossible to complete this thesis without the support of many people. Firstly, I would like to thank my advisor Prof. George A. Kantor. In the last two years, you always gave me valuable suggestions and led me into the right research direction. It would be impossible to finish this master thesis without your valuable advice, insights and patience.

I am also grateful for the support of Prof. Michael Kaess and Ming Hsiao. Thank you for taking time to be my committee members and the useful feedback.

I would also like to thank my labmates and classmates. I really appreciate the time working with Weizhao Shao and Srinivasan Vijayarangan on the Autel mapping project. You always gave me valuable advice thus I could find the efficient ways to solve the problem quickly. It's my honor to work with you and I could not finish this thesis without your help. I am really impressed by your intelligence and diligence, which I will carry along with in the future. I would like to thank Fan yang and Mengqing Jiang for the help of ground extraction part of this thesis.

Finally, I would like to thank my families and friends. Thank you for my parents for their boundless love. Without your support, I could not study here to pursue my life goals. Thank you for my friends to understand me and support me when I have a hard time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Recently, a lot of progress has been made in the robotics field but the full robot autonomy is still challenging. A fully autonomous robot has the ability to perceive its surrounding environments, then make decisions based on what it perceives and finally manipulate the environment or actuate a movement. To achieve the full robot autonomy, a lot of engineers and scientists have been engaged in this field. Simultaneous Localization and Mapping (SLAM) is one crucial skill for robots to perceive the surrounding world and navigate in the unknown environment. SLAM is regarded as an elementary problem for robots to achieve fully autonomy. In recent years, SLAM has gradually become a research hotspot in the robotics community and successfully applied into a wide range of fields including indoor mobile robots, autonomous vehicles, and augmented reality (Figure 1.1). For indoor mobile robots in the GPS-denied environments, they could only rely on the on-board sensors such as cameras, Inertial Measurement Unit (IMU) and Light Detection and Ranging (Lidar) to perform high-level tasks including path planning and obstacles avoidance. For autonomous vehicles, SLAM helps them know where they are and navigate through the unknown

(a) (b) (c)

Figure 1.1: SLAM applications (a) Bossa Nova's retail service Robotics (b) Nuro's robot delivery car (c) Pokémon Go. Image source: (a) *Bossa Nova Robotics Inc.* (b) *Nuro.ai Inc.* (c) *Niantic Inc.*

environments. Furthermore, with the help of the SLAM, different kinds of maps including geometric map and semantic map can be constructed to serve for vehicle planning and perception modules.

Is SLAM solved? Many people may argue SLAM is solved, but most of the current state-of-the-art SLAM algorithms might be fragile, or even fail due to imperfect sensor data, uncontrived environments and hardware failures [44]. For visual SLAM, it is hard to deal with texture-less environments. Sometimes, the captured image may contain large noise thus deteriorating the accuracy of the algorithm. For Lidar-based SLAM algorithms, they are difficult to yield satisfactory results in the environment lacking geometric structures. The accuracy of the Lidar measurements is also affected by weather phenomena such as snow, fog, and rain. IMU could work reliably most of the time, but the state estimation results would drift quickly because of the noisy measurements and incorrect estimation of biases. Some robots carry GPS to get global measurements, but sometimes the GPS signal is bad thus the measurement accuracy is affected. Furthermore, some tasks such as fast closed-loop control require high-frequency state estimation input, but many SLAM algorithms are unable to meet this requirement [4]. Therefore, it is still challenging to design one accurate, robust and high rate SLAM system that can deal with all difficult environments.

SLAM algorithms can also output the map simultaneously which is crucial for robots. The map information can be used by high-level tasks such as path planning and obstacle avoidance. Moreover, the accuracy of state estimation can be improved by utilizing the useful information of the prior map. With the wider application of robotics technologies, the requirements for the map is higher. First, the accuracy requirement of the map is unprecedented in some fields [4]. For example, the autonomous vehicles need maps have a precision requirements of centimeters or millimeters in some special situations. To let robots have a better ability to understand the surrounding environments, the map should provide more information. Semantic 3D mapping which integrates geometric 3D mapping and semantic segmentation is becoming increasingly important for the high-level tasks of the robots. For the autonomous vehicles, useful objects such as lanes, roads, traffic lights and crosswalks are segmented in the map. In this way, it can help vehicles stay in its lane and adhere to laws and polite culture. However, it is still challenging to construct the large-scale and accurate 3D semantic map.

To address these problems, this thesis first concentrates on improving the robustness of the SLAM algorithm by fusing the measurements of multiple sensors including monocualr camera, IMU and Lidar. In this thesis, we will use the multi-sensor fusion for estimating the pose of the robot. The proposed systems consist of a monocular Visual Inertial Odometry(VIO), Lidar odometry, odometry source selection module and Lidar scan to map optimization system. In addition, it's necessary to build the loop closure module because the SLAM algorithm might have the large drift after long-time running. In this thesis, we first use the image features to generate Bag of Words(BoW) [11] to detect the loop closure candidates. Then, the feature matching process is conducted for the current frame and loop candidate frame. The Grid-based Motion Statistics (GMS) feature matcher [2] is utilized to eliminate the matching outliers. After confirming a loop closure candidate, the point to plane ICP algorithm

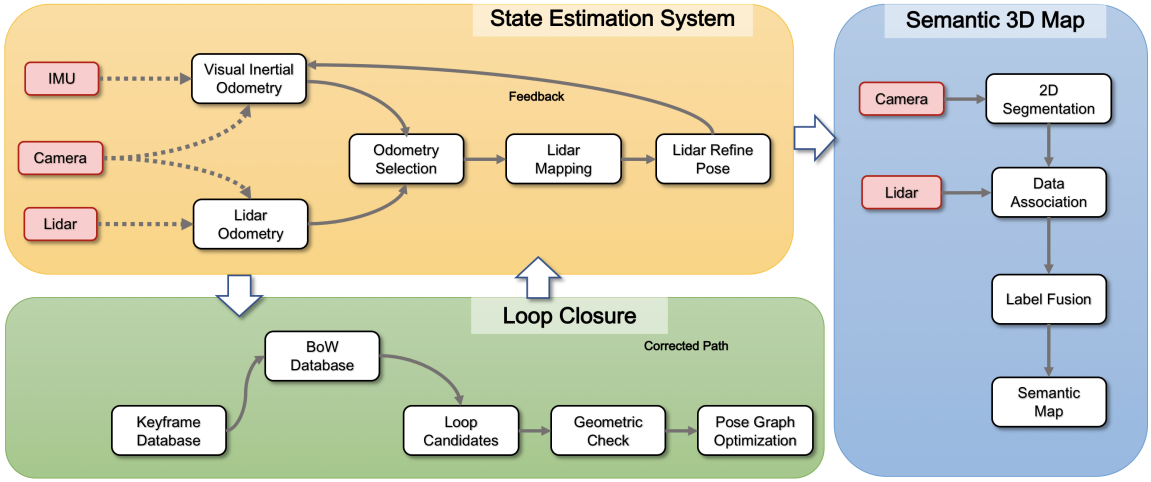

Figure 1.2: Framework of the proposed SLAM system

is employed between sparse feature points of the current frame and loop candidate frame. In this way, the accurate relative pose transformation between these two frames is obtained and added into the pose graph outputting the corrected pose. The loop closure module could also support robot localization and refine the map online. Finally, a large-scale 3D semantic map is constructed by combining the measurements of the Lidar and a camera. To build a 3D semantic map, the previous proposed state estimation pipeline is used to get the pose of the robot, and the Convolutional Neural Network (CNN) is used to segment images. Then, the coordinate transformation is exploited to get the semantic label of the Lidar point and the Bayes' update scheme is used to update the label of the point cloud. The framework of overall system is shown in Figure 1.2.

1.2 Contribution

In this thesis, we concentrate on improving the robustness of the SLAM system and achieve satisfactory accuracy for challenging environments, or even sensor failure case. In addition, the loop closure module is implemented to reduce the accumulated drift.

Finally, the 3D semantic occupancy map is constructed serving for the high-level robotic tasks. In particular, this thesis makes the following contributions:

We propose one multi-sensor fusion state estimation pipeline to improve the robustness and accuracy of the localization system.

We implement monocular VIO and Lidar odometry. For the monocular VIO, the Lidar refined poses are added into the optimization framework to obtain better estimation results. For the Lidar odometry, visual features are utilized to get a better estimation of the orientation.

We design the loop closure module utilizing visual and Lidar points information and support robots for relocalization and map online refinement.

We propose one Lidar semantic 3D occupancy mapping pipeline to construct accurate and large-scale semantic 3D map.

We illustrate the performance of the proposed algorithm on the publicly available datasets as well as custom collected datasets.

1.3 Outline

This thesis is organized as follows. In Chapter 2, state-of-the-art algorithms related to odometry, loop closure and mapping are discussed. In Chapter 3, the proposed state estimation system using multi-sensor fusion is proposed. In Chapter 4, the loop closure module fusing Lidar and monocular camera's information is presented. In Chapter 5, the proposed 3D occupancy semantic pipeline fusing camera and Lidar's information is presented. Chapter 6 first demonstrates the robustness and accuracy of the proposed SLAM system, then the loop closure module is tested with several datasets and the 3D occupancy semantic mapping results are presented. Finally, we conclude the work of this thesis in Chapter 7 and present possible future research directions.

Chapter 2

Related Work

In this chapter, we discuss some state-of-the-art SLAM algorithms. In particular, we will concentrate on the odometry, loop closure and map representations.

2.1 Odometry

When the SLAM algorithm sacrifices the loop closure module, it reduces to the odometry [4]. In the last few decades, there are a lot of odometry algorithms including Visual Odometry (VO) [23, 26, 8], Visual Inertial Odometry (VIO) [12, 28], Lidar-based odometry [40, 7] and multi-sensor odomety algorithms [41, 43]. The VO can be classified into feature-based methods [26, 23], direct methods [8, 9] and semi-direct methods [13]. The feature-based methods first extract the feature of the images, then conduct the feature matching in successive frames and finally recover the pose of the camera by minimizing the reprojection error. However, feature-based methods may fail in featureless environments. Different from feature-based methods, the direct methods take the intensity values of the images and minimize the photometric error to estimate the motion. However, the direct methods are very sensitive to unmodeled artifacts including camera gain control and rolling shutter effect [38]. Semi-direct methods are considered as the hybrid of the feature-based and direct methods and are

proven to be accurate and robust [38]. Semi-direct methods such as SVO [13] combine the advantages of feature-based methods and direct methods, which extracts corner features and recover the camera pose by minimizing the the photometric errors of features. SVO is robust in scenes of little, repetitive, and high frequency-texture due to high frame-rate motion estimation and the outlier resistant probabilistic mapping method [13].

In the last few years, however, more and more odometry algorithms combine visual and inertial measurements, and have very high accuracy. Forster et al. [12] proposed the IMU preintegration theory which addresses the manifold structure of the rotation group and presents uncertainty propagation. In addition, the IMU preintegration measurements are integrated with visual measurements and the whole visual inertial pipeline is implemented by GTSAM [6]. The algorithm is fast and more accurate compared with state-of-the-art filtering and optimization methods. Qin et al. [28] proposed one similar visual inertial odometry system including estimator initialization and failure recovery. In addition to the improvement of accuracy, VIO could also deal with featureless area, illumination change and motion blur. Therefore, in this thesis, we also implement the monocular VIO as one odometry source.

Lidar odometry also plays an important role in the state estimation system. On the KITTI's odometry benchmark [14], Lidar-based algorithms outperform visual methods. Zhang and Singh [40] proposed one real-time Lidar Odometry and Mapping(LOAM) algorithm and ranked among the best methods on the KITTI odometry benchmark. LOAM first extracts feature points including edge and surface Lidar points based on the curvature of each point, then the scan to scan transformation is computed utilizing matched feature points, and finally scan to map matching is conducted to output refined state estimation results. On the basis of high accuracy of the Lidar odometry, this thesis also implements the lidar odometry and mapping

algorithm based on the LOAM. However, pure Lidar-based odometry is hard to deal with some environments lacking geometric features like the tunnel.

In recent years, many algorithms fuse the information of multiple sensors to improve the accuracy and robustness of the odometry. In the work of Zhang and Singh [43], they presented the SLAM algorithm leveraging data from a Lidar, a camera and the IMU. Their algorithm is capable of dealing with sensor degeneration. In our previous work [32], we design a state estimation system leveraging data from a Lidar, stereo cameras and an IMU. Different from the algorithm in [43] which loosely couples visual and inertial measurements, the stereo VIO is implemented and the estimation of Lidar scan to scan transformation is directly obtained from the VIO's high-frequency output instead of Lidar odometry. Similarly, the Lidar scan to map optimization module is implemented to correct the drift of the stereo VIO. Through a series of experiments, the algorithm demonstrated better accuracy and robustness compared with state-of-the-art Lidar odometry, especially in degenerate environments. However, the whole system could not be resilient to failure or wrong optimization results of the VIO. For example, On the KITTI odometry benchmark, some of the sequences miss the IMU measurements in a short period of time thus the whole pipeline stops working. Furthermore, the stereo configuration requires more efforts including precise calibration and time synchronization. If the stereo baseline is much smaller than the distance to the scene from the camera, the stereo case could be degraded to the monocular case [1]. In this way, we choose to implement monocular VIO in this thesis.

2.2 Loop Closure

Loop closure is also one of the ad hoc research topics in the SLAM community. After detecting the loop closure, the robot could eliminate the accumulated long-term

drift. For loop closure at the front-end of the visual SLAM, the Bag-Of-Words (BoW) models [11] are one of the state-of-the-art methods to quickly retrieve the loop candidates. After extracting features of the image, BoW uses hierarchical vocabulary trees to convert features to the words representing the image which is efficient to retrieve the loop images in large datasets. However, BoW is sensitive to the illumination and large viewpoint change. With the development of deep learning, some of loop detection algorithms based on Convolutional Neural Network (CNN) are also proposed [19], but they are hard to satisfy the real-time requirement. For the loop candidate detection in 3D points, state-of-the-art approaches are based on 3D local key-point descriptors [3, 36] or global scene descriptors [17]. However, compared with BoW, Lidar-based loop detection is time-consuming.

After detecting several loop candidates of current frames, loop closure validation should be conducted to reject the false positive candidates. For the visual-based methods, RANSAC is mainly used to eliminate the outlier. Recently, Grid-based Motion Statistics (GMS) [2], a statistical formulation for rejecting outlier based on the number of neighboring matches is proposed. Experimental results show GMS is faster and accurate than RANSAC. In Lidar-based methods, the loop closure is often validated through checking how well the current lidar scan matches the constructed map [4].

2.3 Map Representations

There are numerous map representations in SLAM. In the 2D case, landmark-based maps and occupancy grid maps are two main paradigms. The landmark-based maps represent the surrounding world as the collection of sparse salient landmarks, and the occupancy grid maps divide the whole map into cells which are assigned with the value indicating the probability to be occupied. In the 3D mapping, landmark-

based sparse representations are widely utilized. Many SLAM algorithms represent the environment as the collection of 3D landmarks including feature points, corners and lines [26]. However, the sparse representation would not be useful for some high-level tasks such as robot navigation and human-robot interaction. contrary to the sparse representations, dense representations have the ability to generate the high-resolution of the environment. The simplest representations are to use low-level representations including unstructured point clouds [8], polygons [33] and surfel maps [20]. These map representations, however, occupy a lot of storing space and present a very low description of the objects in the map. Boundary and spatial-partitioning dense representations are higher level representations which attempt to explicitly represent volumes and surfaces [4]. Plane-based representation is one of widely used boundary representations [22].

Recently, with the progress of deep learning, semantic mapping attracts increasing attention [4]. Semantic mapping assigns semantic labels to the objects perceived by the robots, overcoming the limitations of the traditional geometric representations to let robots have the better ability to accomplish complicated tasks. Kundu et al. [24] presented an approach for jointly inference both 3D structure and semantic segmentation. Their method shows that semantic information could improve the 3D reconstruction quality and 3D reconstruction could also help get more consistent semantic segmentation labels. However, it cannot run incrementally in real time without GPU acceleration. Yang et al. [37] utilize the stereo cameras to construct semantic 3D occupancy mapping incrementally in real time. However, the stereo configuration requires more efforts including precise calibration and time synchronization. They also adopt ELAS [15] for the estimation of stereo disparity, but this approach is not highly accurate especially when the baseline is small or the resolution of the image is not high. To overcome the shortcomings of the stereo methods, we fuse the information of Lidar and the monocular camera to reconstruct 3D semantic occupancy map.

Chapter 3

Robust State Estimation Using Multi-Sensor Fusion

In this chapter, we first introduce the proposed multi-sensor fusion algorithm for robust state estimation as shown in 3.1. The whole pipeline consists of monocular VIO, lidar odometry, odometry selection module and Lidar mapping refinement module.

3.1 Monocular VIO

In this section, the implementation details of the monocular VIO including visual frontend and backend optimization is introduced.

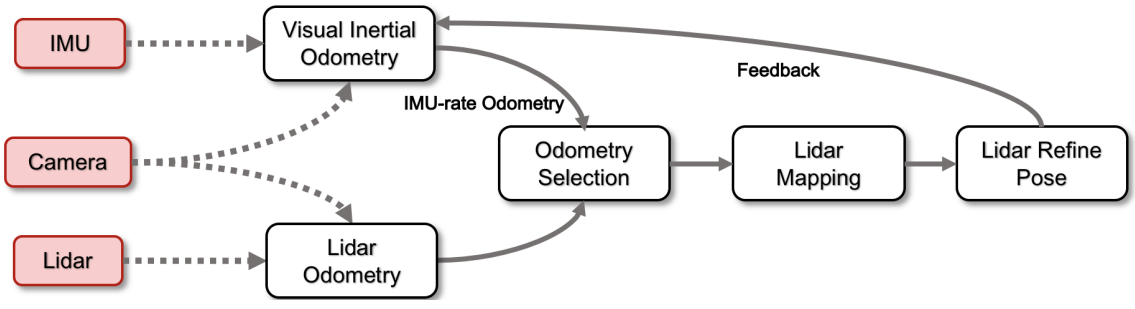

Figure 3.1: State estimation system

3.1.1 Visual Frontend

The visual frontend is crucial for the VIO. In order to get stable and salient visual features to be tracked from frame to frame, Shi-Tomasi corner detector [34] is utilized for each image. In order to extract visual features evenly, the whole image is divided into small cells and the number of visual features in each cell is no more than predefined thresholds. Furthermore, the Features From Accelerated Segment Test (FAST) corners [29] of each frame are detected and the orientation and Oriented FAST and Rotated BRIEF (ORB) [30] descriptors are then computed. These FAST corners and corresponded ORB descriptors would be utilized for the loop closure and Lidar odometry module. In this thesis, Kanade-Lucas-Tomasi (KLT) tracking method [34] is utilized to track corners from frame to frame. If the tracked feature number is smaller than the threshold, we will re-extract new corner features for this frame.

3.1.2 Backend Optimization

In this thesis, we utilize the GTSAM for the implementation of backend optimization [6]. As shown in Figure 3.2, the monocular VIO is a tightly-coupled system integrating the measurements of IMU and camera. For the VIO, the states to be estimated are the IMU position, rotation, velocity and bias: $\boldsymbol{S}_i = [\boldsymbol{R}_i, \boldsymbol{p}_i, \boldsymbol{v}_i, \boldsymbol{b}_i]$. In this thesis, we use $\boldsymbol{\xi}_i = [\boldsymbol{R}_i, \boldsymbol{p}_i] \in SE(3)$ to represent the pose of the IMU at time t, and the bias term could be expressed as $\boldsymbol{b}_i = [\boldsymbol{b}_i^g, \boldsymbol{b}_i^a] \in R^6$, where $\boldsymbol{b}_i^g, \boldsymbol{b}_i^a \in R^3$ are the gyroscope and accelerator bias, respectively.

To utilize high-frequency IMU measurements, the IMU preintegration on Manifold [12] is utilized. First, we introduce the 3-axis accelerate and 3-axis gyroscope measurement models. In this thesis, we define an inertial frame W and IMU frame

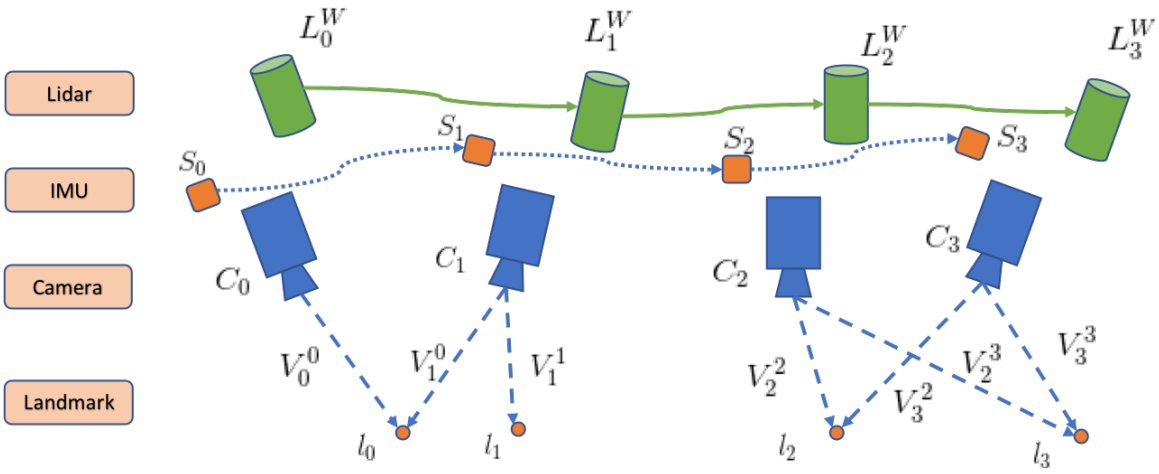

Figure 3.2: Monocular VIO framework

B which coincides with the body frame of the IMU.

$$\tilde{\mathbf{w}}_{WB}^{B}(t) = \mathbf{w}_{WB}^{B}(t) + \mathbf{b}^{g}(t) + \boldsymbol{\eta}^{g}(t) \tag{3.1}$$

$$\tilde{\boldsymbol{a}}^{B}(t) = \boldsymbol{R}_{WB}^{T}(t)(\boldsymbol{a}_{w}(t) - \boldsymbol{g}_{w}) + \boldsymbol{b}^{a}(t) + \boldsymbol{\eta}^{a}(t) \tag{3.2}$$

where the superscript B denotes that the corresponding quantity is expressed w.r.t. frame B. $\boldsymbol{b}$ and $\boldsymbol{\eta}$ denote time-varying sensor bias and noise respectively. $\boldsymbol{g}$ is the gravitational acceleration. $\mathbf{w}_{WB}^{B}(t)$ denotes the instantaneous angular velocity of B relative to W expressed in coordinate frame B. The integration of IMU measurements are as follows:

$$\boldsymbol{R}_{WB}(t+\Delta t) = \boldsymbol{R}_{WB}(t)\mathrm{Exp}\left(\int_{t}^{t+\Delta t} \boldsymbol{w}_{WB}^{B}(\tau)d\tau\right) \tag{3.3}$$

$$\boldsymbol{v}_{w}(t+\Delta t) = \boldsymbol{v}_{w}(t) + \int_{t}^{t+\Delta t} \boldsymbol{a}_{w}(\tau)d\tau \tag{3.4}$$

$$\boldsymbol{p}_{w}(t+\Delta t) = \boldsymbol{p}_{w}(t) + \int_{t}^{t+\Delta t} \boldsymbol{v}_{w}(\tau)d\tau + \iint_{t}^{t+\Delta t} \boldsymbol{a}_{w}(\tau)d\tau^{2} \tag{3.5}$$

We optimize the residual errors between the preintegrated factors and to-be-estimated states. The optimization objectives with preintegrated IMU factors,$\tilde{r}, \tilde{v}, \tilde{p}$

are shown below.

$$r_{\Delta \boldsymbol{R}_{ij}} = \log\left(\tilde{r}_{ij}\right)^T \boldsymbol{R}_i^T \boldsymbol{R}_j \tag{3.6}$$

$$r_{\Delta \boldsymbol{v}_{ij}} = \boldsymbol{R}_i^T(\boldsymbol{v}_j - \boldsymbol{v}_i - \boldsymbol{g}\Delta t_{ij}) - \tilde{\boldsymbol{v}}_{ij} \tag{3.7}$$

$$r_{\Delta \boldsymbol{p}_{ij}} = \boldsymbol{R}_i^T\left(\boldsymbol{p}_j - \boldsymbol{p}_i - \boldsymbol{v}_i\Delta t_{ij} - \frac{1}{2}\boldsymbol{g}\Delta t_{ij}^2\right) - \tilde{\boldsymbol{p}}_{ij} \tag{3.8}$$

where i, j denote two keyframes. These three equations encode the constraints on rotation, velocity and position using the measurement from IMU preintegration. In the process of integration, the bias is assumed to be constant between two keyframes. However, the estimation of the bias changes slowly during optimization, and [12] uses the first-order expansion to update the delta measurements.

For the vision factors, the residual error could be expressed as:

$$\boldsymbol{r}_v = \sum_{l=1}^{L} \sum_{i \in \boldsymbol{\chi}(l)} \left\| \boldsymbol{z}_{il} - \pi(\boldsymbol{\xi_i}, \boldsymbol{\rho}_l) \right\|_{\sum_c}^2 \tag{3.9}$$

where L is the total number of visual features into the optimizer, $\boldsymbol{\chi}(l)$ is the set of cameras could observe this landmark, π represents the projection model of the camera and $\boldsymbol{\rho}_l \in R^3$ denotes the 3D position of the lth landmark. In our implementation, the structureless vision factors are utilized for the linear elimination of landmarks to reduce the computational burden but still obtain the optimal MAP estimation [12]. After using the retraction to lift the cost function [25], the linearization results of the residue in the optimization process could be obtained.

$$\sum_{l=1}^{L} \left\| \mathbf{F}_l \delta \boldsymbol{\xi}_{\boldsymbol{\chi}(\boldsymbol{l})} + \mathbf{E}_l \delta \boldsymbol{\rho}_l - b_l \right\|_{\sum_c}^2 \tag{3.10}$$

where $\mathbf{F}_l$ and $\mathbf{E}_l$ are the Jacobian matrices and b_l is the error term. $\delta\boldsymbol{\rho}_l$ could be eliminated by projecting the residue into the null space of $\mathbf{E}_l$:

$$\sum_{l=1}^{L} \left\| (\mathbf{I} - \mathbf{E}_l(\mathbf{E}_l^T\mathbf{E}_l)^{-1}\mathbf{E}_l^T)(\mathbf{F}_l\delta\boldsymbol{\xi}_{\boldsymbol{\chi(l)}} - b_l) \right\|_{\sum_c}^2 \tag{3.11}$$

in this way, a lot of landmarks are eliminated during the optimization process and only the poses are optimized.

For this multi-sensor state estimation system, Lidar refined poses are obtained by Lidar mapping optimization, which is the final output of the system and will be introduced in the following section. It's beneficial to utilize the feedback information of lidar refined poses to get more accurate and robust IMU states. The Lidar refined pose could still have the accumulated drift, but the relative Lidar pose is usually more accurate than the VIO's estimation. In this way, the *Lidar refined pose between factor* is added into the pose graph to model this relative motion. Suppose $\mathbf{L}_{t-1}^W$ and $\mathbf{L}_t^W$ are the refined poses of the Lidar relative to the world coordinate frame at time $t-1$ and t. Thus the relative Lidar motion $\mathbf{L}_t^{t-1}$ can be expressed as $(\mathbf{L}_{t-1}^W)^{-1}\mathbf{L}_t^W$. However, we should obtain the relative pose of the IMU instead of the Lidar, each relative Lidar motion $\mathbf{L}_t^{t-1}$ has to be transformed into the relative IMU motion $\boldsymbol{\zeta}_t^{t-1}$ before adding in the VIO's pose graph. The transformation is denoted as:

$$\boldsymbol{\zeta}_t^{t-1} = \mathbf{T}_L^I\mathbf{L}_t^{t-1}(\mathbf{T}_L^{I\,-1}) \tag{3.12}$$

where $\mathbf{T}_L^I$ is the extrinsic between IMU and Lidar, which transforms the point in the Lidar coordinate frame to the IMU coordinate frame. This can be calibrated offline previously. The residue of the *Lidar refined pose between factor* is denoted as:

$$r^L = (\boldsymbol{\zeta}_t^{t-1})^{-1}(\boldsymbol{\xi_{t-1}})^{-1}\boldsymbol{\xi_t} \tag{3.13}$$

for the optimization of the pose graph, the bounded-size sliding window is utilized to optimize recent states and marginalize out old states and measurements. The marginalization process is implemented based on the Schur complement [35] and the prior factor based on the marginalized measurements corresponded to the removed states is added into the graph.

The optimal states of the proposed monocular VIO could be expressed as:

$$\mathbf{S}_t{}^* = \underset{\mathbf{S}_t}{\operatorname{argmin}}(\|\mathbf{r}^p\|^2_{\sum_p} + \sum_{i\in w} \|\mathbf{r}^I_{i(i+1)}\|^2_{\sum_I} + \sum_{i\in w} \|\mathbf{r}^L_{i(i+1)}\|^2_{\sum_l} + \sum_l \|\mathbf{r}_v\|^2_{\sum_v}) \tag{3.14}$$

where $\mathbf{r}^p$, $\mathbf{r}^I_{i(i+1)}$, $\mathbf{r}^L_{i(i+1)}$ and $\mathbf{r}_v$ denote residues for the prior factor, IMU measurements, refined Lidar feedback information and the visual measurements. $\sum_p$, $\sum_I$, $\sum_l$ and $\sum_v$ signify the covariance of these measurements.

After the optimization, the camera-rate VIO is obtained. In order to generate the IMU-rate VIO, we propagate the latest optimized state with the most recent IMU measurements. Sometimes, the number of visual features in the optimizer is small or there are a lot of outliers which can cause sudden jumps. Therefore, besides publishing IMU-rate VIO, another odometry source is generated only by the IMU-preintegration. Before the preintegration, the bias and velocity of IMU are corrected using the results of latest optimized results. In this thesis, this odometry is called corrected IMU-rate smooth odometry.

3.2 Lidar Odometry

LOAM [40] has been proven to be accurate and efficient. In the LOAM, edge and planar feature points are first extracted, then the scan to scan and scan to map optimization is conducted to get the robot motion. But [31] points out that there may be sensor noise from the ground because Lidar in the vehicle is always close to the ground, and thus unreliable edge points may be extracted from the ground. Further-

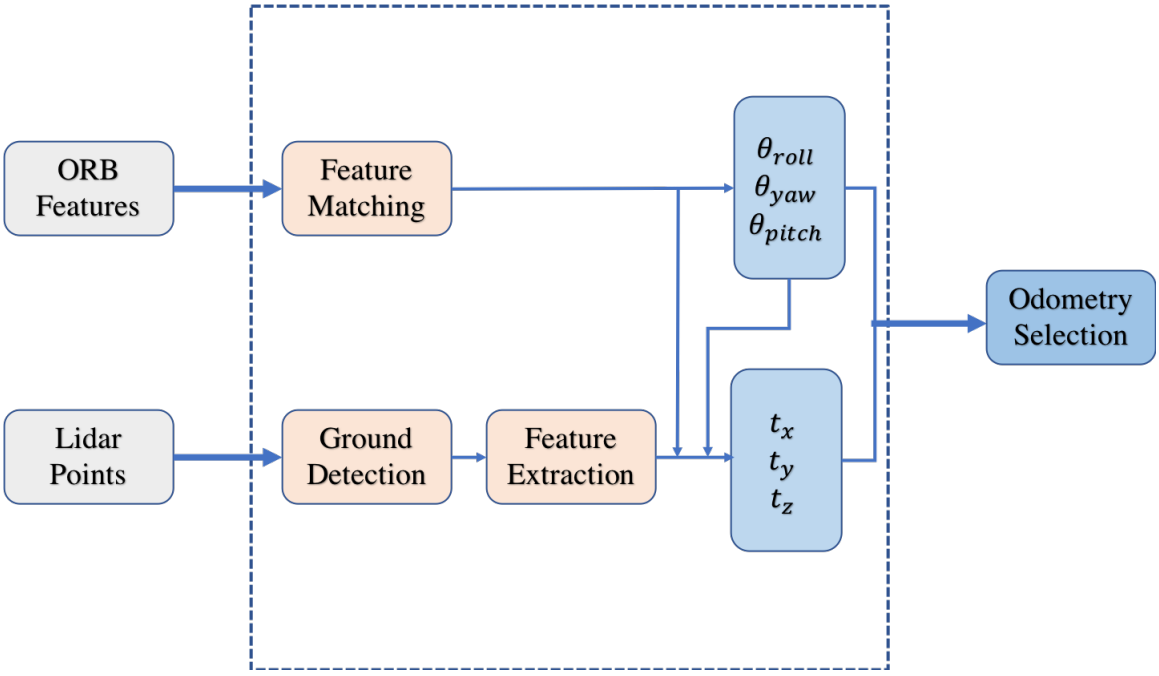

Figure 3.3: Lidar odometry framework

more, [21] indicates that visual measurements are slightly superior for estimating the rotation movement, whereas Lidar is better for estimating the translation.

To overcome the limitations of the LOAM, we implement one Lidar Oodmetry fusion visual measurements as shown in Figure 3.3. For the Lidar odometry, we first extract the ground points from the point cloud to avoid the noisy edge features are extracted from that region. Inspired by the ground extraction method in LeGO-LOAM [31], we cluster points that have relatively low angle between adjacent points.

Specifically, we use all points on the lowest half part of scan as initial seeds (i.e. 8 rows for VLP-16 or 32 rows for HDL-64E), and then cluster points from near to far according to neighborhood vertical angles, as visualized in Figure 3.4. The clusters whose number of points is over the threshold are labeled as ground. The pseudo code is given in Algorithm 1.

Then, the edge and planar feature points from the raw point cloud are extracted. To extract the edge and planar feature points, the curvature of each point is calculated

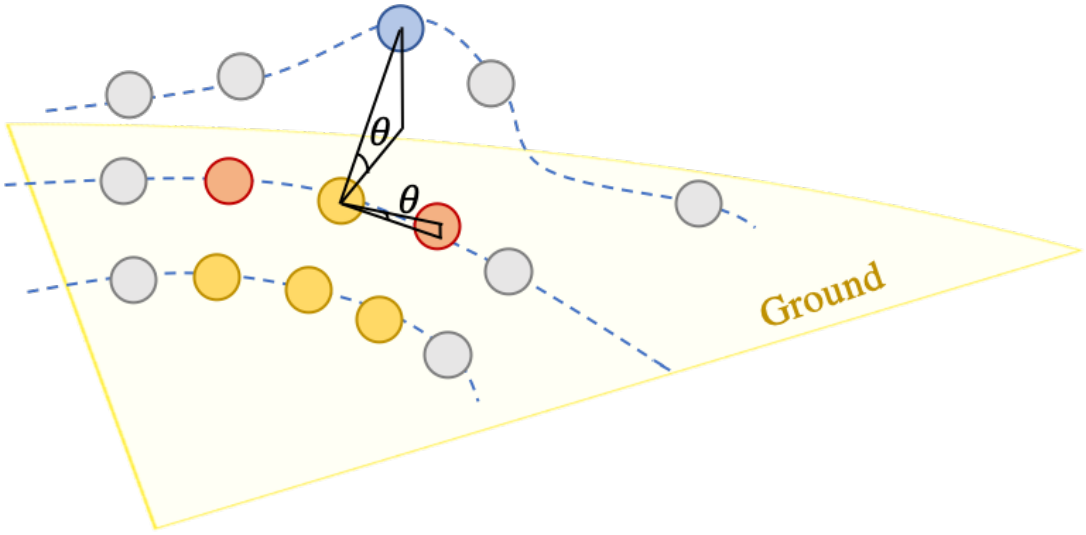

Figure 3.4: Ground Plane Fitting. Yellow dots are estimated ground points, blue dots are estimated non-ground points and red dots are lowest point representatives.

Algorithm 1 Our proposed ground extraction algorithm

```
1:  procedure GroundExtraction
2:      for j ← 0 to Horizon_SCAN do
3:          for i ← 0 to groundScanInd do
4:              GroundCluster(i, j)
5:  procedure GroundCluster(row, col)
6:      q ← New Queue
7:      cluster ← New Array
8:      push (row, col) to q
9:      while q is not empty do
10:         P ← top of q
11:         pop q
12:         for P_i ← neighboors of P do
13:             θ ← vertical angle between P and P_i
14:             if θ < angleThreshold then
15:                 push P_i to q
16:                 add P_i to cluster
17:         if size of cluster > sizeThreshold then
18:             label points in cluster as ground
```

as:

$$c = \frac{1}{|S| \cdot \|\boldsymbol{X}^L_{(k,i)}\|} \| \sum_{j \in S, j \neq i} (\boldsymbol{X}^L_{k,i} - \boldsymbol{X}^L_{k,i}) \| \tag{3.15}$$

where $\boldsymbol{X}^L_{k,i}$ denotes the ith 3D Lidar point expressed in the Lidar coordinate frame and S is the set of neighboring points of $\boldsymbol{X}^L_{k,i}$ in the same ring. The ring number of one Lidar point represents which laser beam detects this point. The large c value indicates this point is the edge feature point, whereas the small curvature represents the planar feature point. In our implementation, each ring is divided equally into six segments, and the curvature of each point is calculated then sorted. Finally, for each segment, two points with the largest curvature and not on the ground are selected as the edge features, five points with the smallest curvature are used for the planar features.

To utilize the ORB features extracted from the visual frontend, we first use the brute-force method to match the ORB features of the previous frame and the current frame. After feature matching, GMS [2] is utilized to eliminate the outliers. If the feature number is higher than the predefined threshold, the angular components are estimated based on Nister's 5-point algorithm [27]. Finally, we integrate the information of visual measurements and Lidar feature points for estimating the motion translation. Let $\boldsymbol{E}_k$ and $\boldsymbol{H}_k$ denote the set of edge feature points and planar feature points for the current scan, respectively. Let $\boldsymbol{E}_{k-1}$ and $\boldsymbol{H}_{k-1}$ represent the set of edge feature points and planar feature points for the previous scan, respectively. Following the methods in [40], for the edge feature point $\tilde{\boldsymbol{X}}^L_{k,i}$ in $\boldsymbol{E}_k$, two nearest edge points $\tilde{\boldsymbol{X}}^L_{k-1,j}$ and $\tilde{\boldsymbol{X}}^L_{k-1,l}$ in $\boldsymbol{E}_{k-1}$ are retrieved using the k-d tree and the motion estimation of the previous scan to current scan. These three points set then construct the point to line distance residue:

$$r_e = \frac{|(\tilde{\boldsymbol{X}}^L_{k,i} - \tilde{\boldsymbol{X}}^L_{k-1,j}) \times (\tilde{\boldsymbol{X}}^L_{k,i} - \tilde{\boldsymbol{X}}^L_{k-1,l})|}{|\tilde{\boldsymbol{X}}^L_{k-1,j} - \tilde{\boldsymbol{X}}^L_{k-1,l}|} \tag{3.16}$$

similarly, for the planar feature point $\tilde{\boldsymbol{X}}^L_{k,i}$ in $\boldsymbol{H}_k$, three closest planar points $\tilde{\boldsymbol{X}}^L_{k-1,j}$, $\tilde{\boldsymbol{X}}^L_{k-1,l}$ and $\tilde{\boldsymbol{X}}^L_{k-1,m}$ in $\boldsymbol{H}_{k-1}$ are found to construct the point to plane distance residue:

$$r_p = \frac{\left| \begin{array}{c} \tilde{\boldsymbol{X}}^L_{k,i} - \tilde{\boldsymbol{X}}^L_{k-1,j} \\ (\tilde{\boldsymbol{X}}^L_{k-1,j} - \tilde{\boldsymbol{X}}^L_{k-1,l}) \times (\tilde{\boldsymbol{X}}^L_{k-1,j} - \tilde{\boldsymbol{X}}^L_{k-1,m}) \end{array} \right|}{|(\tilde{\boldsymbol{X}}^L_{k-1,j} - \tilde{\boldsymbol{X}}^L_{k-1,l}) \times (\tilde{\boldsymbol{X}}^L_{k-1,j} - \tilde{\boldsymbol{X}}^L_{k-1,m})|} \tag{3.17}$$

For the 2D-2D visual feature correspondences, the residue could be constructed based on the epipolar constraint:

$$r_v = \boldsymbol{s}^T \boldsymbol{E} \boldsymbol{m} \tag{3.18}$$

where $\boldsymbol{s} = [u_s, v_s, 1]^T$ and $\boldsymbol{m} = [u_m, v_m, 1]^T$ are the matching feature points of previous frame and current frame, respectively, $\boldsymbol{E}$ is the essential matrix, which is defined as:

$$\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R} \tag{3.19}$$

Stacking (3.16), (3.17) and (3.19) for each Lidar edge feature point, plannar feature point and matched visual feature, the nonliner optimization function is obtained,

$$\boldsymbol{f}(\boldsymbol{T}^L_k) = \boldsymbol{r} \tag{3.20}$$

where each row of $\boldsymbol{f}$ corresponds to a feature point, and each row of $\boldsymbol{r}$ denotes the residue generated by that feature. Then, the Jacobian matrix of $\boldsymbol{f}$ with respect to the current scan transformation $\boldsymbol{T}^L_k$ is calculated, which is expressed as $\boldsymbol{J}$. Based on the Levenberg–Marquardt algorithm, the $\boldsymbol{T}^L_k$ is updated following:

$$\boldsymbol{T}^L_k = \boldsymbol{T}^L_k - (\boldsymbol{J}^T \boldsymbol{J} + \lambda diag(\boldsymbol{J}^T \boldsymbol{J}))^{-1} \boldsymbol{J}^T \boldsymbol{d} \tag{3.21}$$

where λ is one parameter affecting the performance of the Levenberg–Marquardt algorithm.

3.3 Odometry Selection

There are three odometry sources including IMU-rate VIO, corrected IMU-rate smooth odometry and Lidar odometry so far. Through several experiments, we find IMU-rate VIO and corrected IMU-rate smooth odometry are more accurate than the Lidar odometry integrated from scan to scan transformation. Furthermore, The IMU-rate odometry sources can provide high-frequency poses, which are beneficial for the Lidar point cloud dewarping. Therefore, we design the odometry selection framework which prefers the IMU-rate VIO and corrected IMU-rate smooth odometry. But the sensor degradation may happen in some specific situations. For the VIO, sometimes, the insufficient number of visual features or the outliers may cause the odometry jumps. Furthermore, it's challenging for the Lidar odometry to deal with structureless environments such as pure planes and tunnels. Moreover, the system should be robust enough to handle aggressive motion and sensor failures. Therefore, it's important to select one appropriate odometry source used by the Lidar mapping module. To deal with the sensor degradation, [44] proposed one solution to deal with the degeneracy cases. Let $\boldsymbol{J}$ be the Jacobian matrix of Lidar odometry, and the eigenvalues of $\boldsymbol{J}^T\boldsymbol{J}$ are computed. Theses eigenvalues represent how well the optimization result is conditioned in the direction of eigenvectors [44]. In this way, small eigenvalue indicates the degenerate case of the Lidar odometry.

If the number of visual features is larger than Thr_{med}, we select the odometry source between IMU-rate VIO and corrected IMU-rate smooth odometry. For the VIO, the velocity estimation can be obtained from the IMU-rate VIO and corrected IMU-rate smooth odometry, respectively. If the VIO jumping happens, the optimized

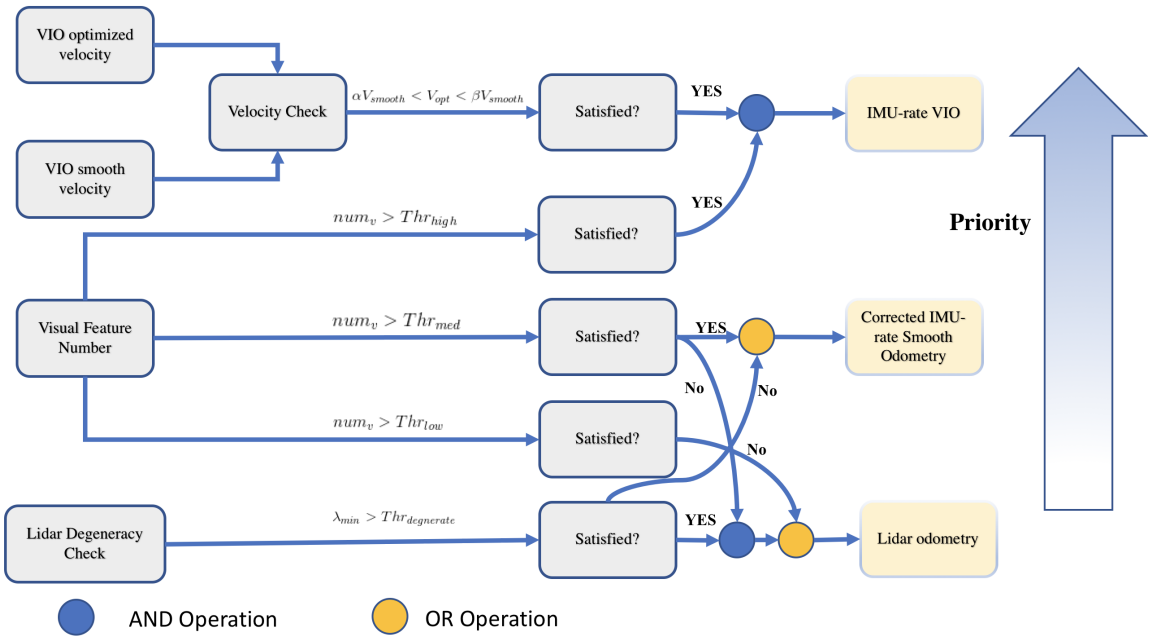

Figure 3.5: Odometry selection framework

velocity from IMU-rate VIO is much larger than the preintegration velocity from IMU-rate smooth odometry. Therefore, the velocity check is implemented to check whether the odometry jumping exists. If the velocity check is satisfied and the visual features are abundant, we use the IMU-rate VIO. If the current condition does not satisfy the requirements of choosing the IMU-rate VIO, the odometry source is selected between corrected IMU-rate smooth odometry and Lidar odometry. If the feature number is too small, considering VIO may drift very quickly, we directly choose Lidar odometry regardless of whether Lidar odometry degeneracy exists. If the visual feature number is larger than Thr_{med}, we directly choose corrected IMU-rate smooth odometry. When the visual feature number is greater than Thr_{med} and fewer than Thr_{low}, the Lidar odometry is selected when the Lidar odometry is not degenerate. Otherwise, we select the corrected IMU-rate smooth odometry. The flowchart of the odometry selection is shown in Figure 3.5.

This framework is used when the VIO and the Lidar odometry both work well. However, if the VIO completely fails, for example when there is no IMU information,

we directly utilize the Lidar odometry. Similarly, when the Lidar odometry completely fails, the VIO odometry is directly used. After determining the odometry source, all Lidar points of the current scan are dewarped to the end of the scan to account for the motion skew. Suppose a Lidar point at t_i as $\boldsymbol{P}_i^L$ in the Lidar frame, the Lidar pose at that time is $\boldsymbol{T}_i^L$, and the Lidar pose at the end of scan is $\boldsymbol{T}_{k+1}^L$, the dewarped Lidar point $\tilde{\boldsymbol{P}}_i^L$ can be obtained by:

$$\tilde{\boldsymbol{P}}_i^L = (\boldsymbol{T}_{k+1}^L)^{-1} \boldsymbol{T}_i^L \boldsymbol{P}_i^L \tag{3.22}$$

if using the VIO source odometry, which is IMU-rate, $\boldsymbol{T}_i^L$ is retrieved by transforming the closest VIO pose from IMU frame to the Lidar frame. If the current odometry source is from Lidar part, the constant velocity model in [40] is exploited to get $\boldsymbol{T}_i^L$.

3.4 Lidar Mapping

In the Lidar mapping algorithm, the transformation $\boldsymbol{T}_{L_k}^W$ which transforms one point from current Lidar frame to world frame is estimated. Denote $\boldsymbol{T}_{L_{k-1}}^W$ is the refined Lidar mapping pose of the previous scan, and $\tilde{\boldsymbol{T}}_{L_k}^{L_{k-1}}$ is the relative pose transformation of the current scan to the previous scan in the Lidar frame. The $\tilde{\boldsymbol{T}}_{L_k}^{L_{k-1}}$ is expressed as:

$$\tilde{\boldsymbol{T}}_{L_k}^{L_{k-1}} = (\tilde{\boldsymbol{T}}_{L_{k-1}}^W)^{-1} \tilde{\boldsymbol{T}}_{L_k}^W \tag{3.23}$$

where $\tilde{\boldsymbol{T}}_{L_{k-1}}^W$ and $\tilde{\boldsymbol{T}}_{L_k}^W$ are from the output of the odometry selection. The initial estimation of $T_{L_k}^W$ can be calculated as:

$$\boldsymbol{T}_{L_k}^W = \boldsymbol{T}_{L_{k-1}}^W (\tilde{\boldsymbol{T}}_{L_{k-1}}^W)^{-1} \tilde{\boldsymbol{T}}_{L_k}^W \tag{3.24}$$

Similar to LOAM, 10 times more feature points are used in the Lidar Mapping module. All feature points are first transformed into the world frame using $\boldsymbol{T}_{L_k}^W$.

Similar to the methods in Lidar odomety, for each edge point, the closest edge points in the previously built map are retrieved, then the point to line residue (3.16) is constructed. For each planar point, the point to plane residue (3.17) is constructed. Different from LOAM, we also add the plane-plane constraint into the optimization framework.

Denote $\boldsymbol{G}_k$ as the set of dewarped ground points of the current scan and $\boldsymbol{G}_m$ as the set of dewarped ground points of the previously built map. For one ground point of the current scan, we first retrieve 20 closest ground points in $\boldsymbol{G}_k$ to fit the plane π_s, and the normal vector $\boldsymbol{n}_s$ is computed. Using the $\boldsymbol{T}_{L_k}^W$, this ground point can be transformed into the map, and 20 closet ground points in $\boldsymbol{G}_m$ are retrieved to fit the plane π_m whose normal vector is $\boldsymbol{n}_m$. For the pair of $\boldsymbol{n}_s$ and $\boldsymbol{n}_m$, they are parallel and the error function is:

$$r_n = \|[\boldsymbol{R}_k^W \boldsymbol{n}_s]_\times \boldsymbol{n}_m\| \tag{3.25}$$

For the Lidar mapping optimization, the optimization function is obtained by stacking (3.16), (3.17), (3.25) formed by edge, planar and ground feature points to get the refined Lidar mapping pose, which is also used as the feedback odometry information of monocular VIO.

Chapter 4

Loop Closure

Even though multiple sensors are used for robust and accurate state estimation, the odometry drift still exists and would be accumulated in the system. For instance, if the robot is moving on a horizontal ground which is represented as a slope in the map due to the inaccurate orientation estimate, after Lidar mapping optimization, there would be Z-direction drift. To eliminate drifts and refine the map distortion, we propose a loop closure module that integrates the visual measurements and sparse Lidar feature points. As shown in Figure 4.1, the loop closure module begins with loop frame detection using Bag of Words (BoW) and geometric check. If the loop candidate is confirmed, relative pose between current keyframe and loop frame is obtained using the ICP matching of sparse Lidar features. For the pose graph optimization, the Lidar Mapping odometry constraint is used as a 6-DoF pose-to-pose factor. Once the loop detection succeeds, the 6-DoF pose-to-pose loop constraint is also added into the pose graph, and iSAM2 is utilized to close the loop. Finally, the refined Lidar poses are generated and the Lidar map is also corrected.

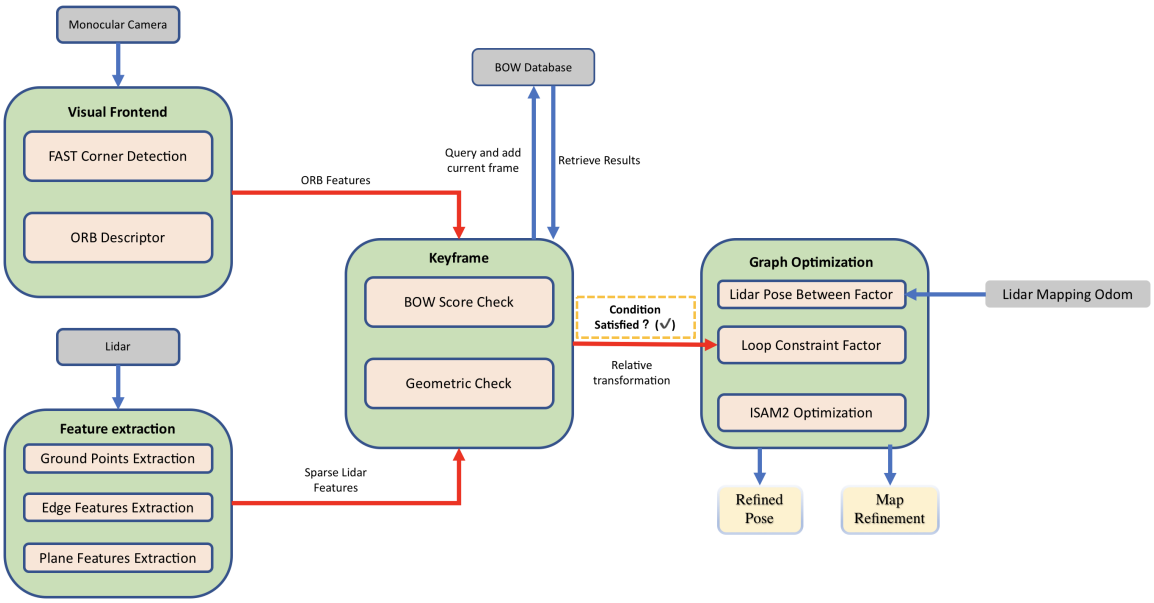

Figure 4.1: Loop closure module

4.1 Loop Detection

For the loop candidates detection, the DBoW3 [18] is utilized for the place recognition. In this thesis, the ORB descriptors extracted by the visual frontend of the monocular VIO are converted into the visual words to query the visual database. In order to limit the size of the visual database, we only add the keyframe into the visual database. For the incoming visual frame with the corresponded odometry, we check its relative translation and orientation to the last keyframe. If the relative motion is larger than the threshold, we treat it as the new keyframe and add into the visual database. Furthermore, in order not to retrieve recent frames as the loop candidates, the loop candidates with small time interval are discarded. Then, the geometric check is conducted for the loop candidate with high similarity score. The brute-force matching is firstly conducted to match the ORB descriptors of the current keyframe and loop candidate, and the GMS filter is utilized again to eliminate outliers. If the number of inliers is above one threshold, this loop candidate is treated as a valid loop

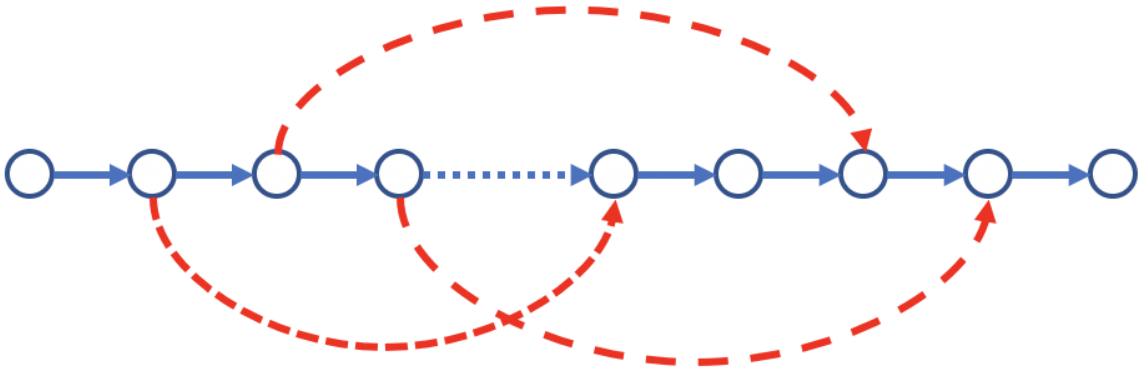

Figure 4.2: Pose-graph example

candidate. But if multiple loop candidates exist, we choose the earliest frame as the final loop candidate due to the smallest odometry drift it contains.

4.2 Pose Graph Optimization

In the loop closure module, the Lidar odometry pose graph as shown in Fig. 4.2 is maintained. The pose graph consists of the sequential edge (blue arrows) and the loop closure edge (red arrows). The sequential edge represents the relative transformation between two consecutive Lidar scans, where value is directly taken from the Lidar mapping refined pose. The loop closure edge represents the relative transformation between the loop frame and the visual frame. In this thesis, we also store the corresponded sparse Lidar feature points of each visual keyframe, and the point-to-plane ICP is utilized to get the relative transformation of the point clouds. The iSAM2 is utilized for the optimization of the pose graph.

4.3 Relocalization and Map Online Correction

The loop closure module also helps to relocalize the robot and correct the distorted map. The Lidar mapping module also stores the sparse feature points of each Lidar scan in the Lidar local frame. Suppose at time t_k, one loop candidate is confirmed,

and at time t_{k+n}, the loop closure optimization is completed. The k denotes the kth lidar scan is received. Once the loop candidate is confirmed, the Lidar mapping module continues scan to map matching optimization but only stores the relative transformations $\boldsymbol{T}_{set} = [\boldsymbol{T}_{L_{k+1}}^{L_k}, \boldsymbol{T}_{L_{k+2}}^{L_{k+1}}, ..., \boldsymbol{T}_{L_{k+n}}^{L_{k+n-1}}]$ instead of publishing the refined pose. After the loop closure module completes the pose graph optimization, the Lidar mapping module receives the corrected poses of each scan $[\tilde{\boldsymbol{T}}_{L_0}^{W}, \tilde{\boldsymbol{T}}_{L_1}^{W}, ..., \tilde{\boldsymbol{T}}_{L_n}^{W}]$, and using the stored relative transformations $\boldsymbol{T}_{set}$ to get the current relocalized pose. Finally, the Lidar mapping module clears the original map and transforms the sparse feature points of each scan from Lidar frame to the world frame using the refined poses. In the process of map online correction, the Lidar mapping optimization is suspended and the refined pose output is just purely integrated with the relative transformation obtained from the odometry selection.

Chapter 5

Semantic 3D Occupancy Mapping

In this chapter, we present one semantic 3D occupancy mapping framework by fusing the monocular camera and a 3D Lidar for large-scale environments. We first introduce this multimodal sensor-based semantic mapping framework, followed by the method of data association of the 3D Lidar point cloud and the 2D image semantic labels. Then, we detail the semantic label fusion approach used in this framework.

5.1 Multimodal Sensor-based Semantic Mapping Framework

Recently, some of the semantic 3D mapping frameworks based on stereo cameras are proposed [24, 37]. However, these methods utilize the stereo cameras to estimate the depth, which is not accurate sometimes, thus generating the inaccurate 3D semantic map. Furthermore, the stereo configuration prefers the wide baseline for the accurate estimation of depth, which is challenging for the small-size robots. To address the limitations of the stereo-based method, as shown in Fig. 5.1, one multimodal sensor-based semantic mapping framework fusing the 3D Lidar and the camera is proposed. The main procedures are listed as follows:

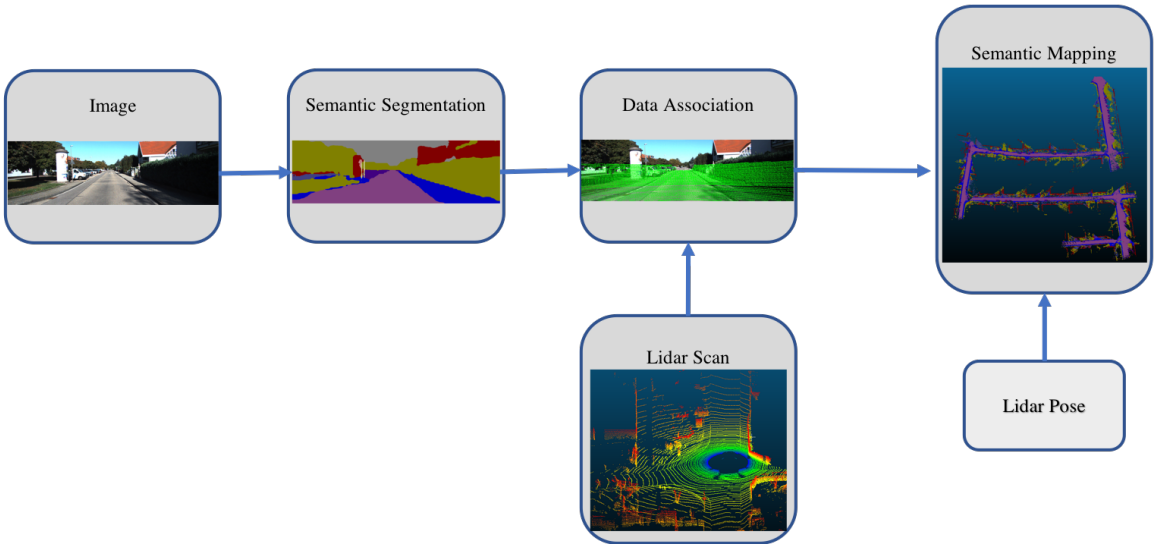

Figure 5.1: Multimodal sensor-based semantic mapping framework

1. Acquire the image of the current scene and the corresponded full 3D Lidar point cloud.

2. Use the dilated CNN [39] to get the semantic label of each pixel for current image.

3. Project 3D Lidar points into the image frame to get the probability distribution of valid projected Lidar points.

4. Estimate the Lidar pose using the method in Chapter 3.

5. Continuously perform the semantic 3D occupancy mapping over time with Lidar pose and standard Bayes' update for label fusion.

5.2 Data Association

For each frame, we use the dilated CNN to get the label probability distribution for each pixel. The dilated convolutions are used to systematically aggregate multiscale contextual information without losing resolution, which is suitable for the dense prediction such as semantic segmentation [39]. In this thesis, we only construct the 3D occupancy map for outdoor environments. Therefore, we select eleven labels (*building, road, vegetation, car, signage, sidewalk, fence, pedestrian, pole, and cyclist*). For

each pixel $u_{i,j}$, after the 2D semantic segmentation, the probability distribution set $\boldsymbol{P} = [p_1, p_2, ..., p_{11}]$ is generated and each element p_i denotes the probability of the ith class. In order to get the semantic probability distribution of the Lidar point, the Lidar points are firstly transformed into the camera coordinate system using the extrinsic T_L^C:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = T_L^C \begin{bmatrix} X_l \\ Y_l \\ Z_l \end{bmatrix} \tag{5.1}$$

where $[X_l, Y_l, Z_l]^T$ and $[X_c, Y_c, Z_c]^T$ denote the point in the Lidar frame and camera frame, respectively. The transformed points with negative Z value are eliminated because they cannot be projected into the image. Then, denote $\boldsymbol{K}$ as the camera intrinsic, the pixel coordinate could be obtained:

$$s \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{5.2}$$

where $[u_x, u_y]^T$ is the pixel coordinate in the image, if it is within the boundary of the image, the semantic information of this pixel is used for the Lidar point.

5.3 Label Fusion for 3D Occupancy Mapping

In order to save the storage memory and construct the semantic map in real-time. The 3D occupancy grids [10] are used to fuse the observations from different views. In our implementation, the grid size is set as $0.1m \times 0.1m \times 0.1m$. Each grid stores the occupancy value and label distribution, which are updated incrementally. For the occupancy value and label distribution update, the standard Bayes's update method is utilized [37]. At time t, denote the label probability distribution of one grid is

x_t, and all the measurements till now are represented as $z_{1:t}$. For the incoming new measurement z_t, the label of the occupancy grid is updated following:

$$p(x_t|z_{1:t}) = \frac{p(x_t|z_t)p(z_t)}{p(x_t)} \frac{p(x_{t-1}|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{5.3}$$

$$\approx \frac{1}{Z} p(x_t|z_t)p(x_{t-1}|z_{1:t-1}) \tag{5.4}$$

where the class prior $p_(x_t)$ is treated as a constant and $\frac{p(z_t)}{p(z_t|z_{1:t-1})}$ is represented by the normalization constant. Following (5.4), the label of each grid of is incrementally updated by simply multiplication followed with normalization.

Chapter 6

Experimental Results

In this chapter, a series of experiments are conducted to test our proposed SLAM framework. First, the odometry sequences of KITTI [14] dataset are used to validate the accuracy and the robustness of the multi-sensor state estimation system. Then, we also collected several indoor and outdoor challenging datasets to further evaluate the robustness of our state estimation system. Furthermore, some experiments related to the loop closure module are conducted to evaluate the absolute translation error of the refined pose graph and the accuracy of the corrected point cloud map. Finally, the KITTI dataset is used to demonstrate the scalability of the semantic 3D occupancy mapping.

6.1 Odometry Test

The KITTI odometry dataset is first used to validate the accuracy and the robustness of the multi-sensor state estimation system. In this work, we utilize the 10 Hz Lidar point cloud from Velodyne HDL-64E, 10 Hz color images from left color camera and 100 Hz IMU measurements of the dataset. For our state estimation system, there are three odometry sources: Lidar odometry, IMU-rate odometry and corrected IMU-rate smooth odometry. We first evaluate the accuracy of our Lidar odometry compared

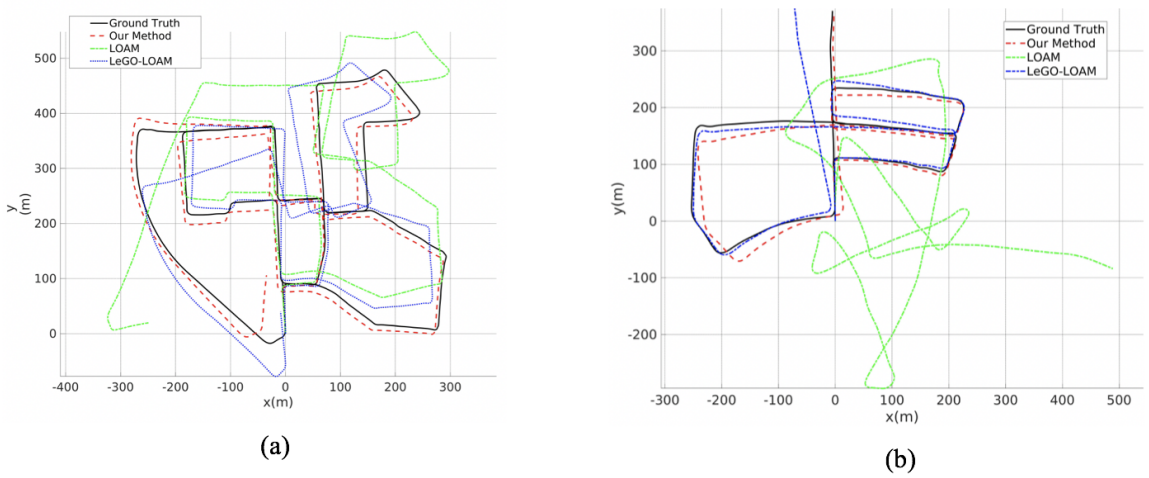

Figure 6.1: Comparative odometry results of KITTI (a) 00 sequence (b) 05 sequence

with the odometry results of LOAM [40] and LeGO-LOAM [31], which are shown in Fig. 6.1. The results of KITTI 00 and 05 sequences show that the angular components are not estimated well for LOAM but LeGO-LOAM improves the accuracy because it filters unreliable feature points. Furthermore, our method outperforms the LOAM and LeGO-LOAM due to the utilization of the visual measurements.

We use KITTI odometry dataset to demonstrate the accuracy of three odometry sources implemented in this thesis. Results of our implemented monocular VIO and Lidar odometry are shown in Fig. 6.2 and Fig. 6.3. For some sequences, the IMU information is lost and the VIO might fail, so the VIO can only run several datasets of KITTI. From the results, we can conclude that the VIO results have better accuracy, especially in the Z direction. Furthermore, the results of VIO optimized odometry (IMU-rate VIO) and VIO smooth odometry (corrected IMU-rate smooth odometry) are close, and there is some big jump for IMU-rate VIO due to the lack of visual constraints sometimes.

Following the odometry selection procedure, the appropriate odometry source is selected, which is used for the Lidar mapping module. Finally, the refined Lidar pose is obtained after the Lidar mapping optimization. In this thesis, the KITTI evaluation tool is used to get the relative translation errors for all possible sub-sequences of length

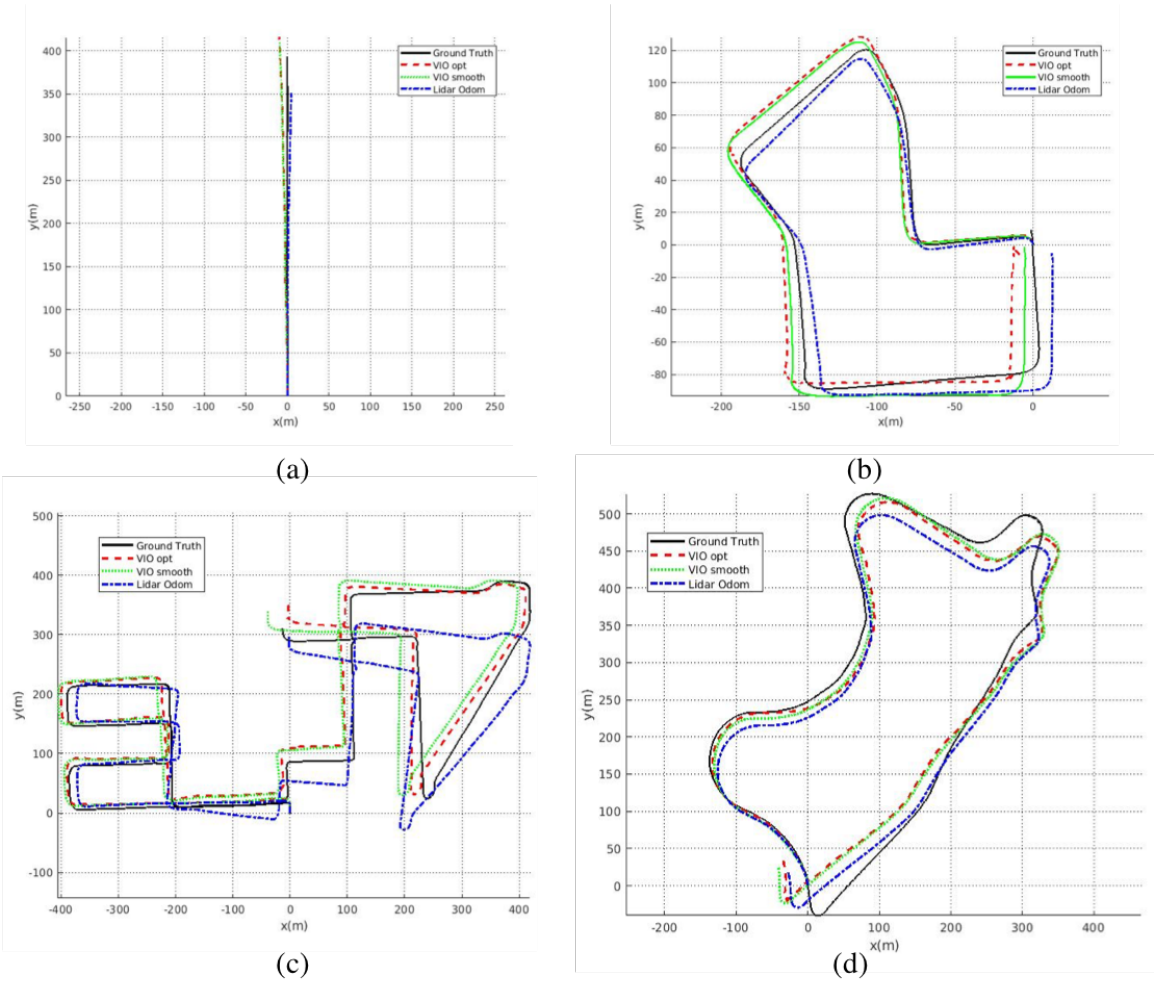

Figure 6.2: VIO and Lidar odometry results of KITTI (X-Y Value) (a) 04 sequence (b) 07 sequence (c) 08 sequence (d) 09 sequence

(100,...,800) meters. We also compare our method with LOAM, LeGO-LOAM and LIMO [16], and the mean translation errors of these methods are shown in Table 6.1. Fig. 6.4, 6.5, 6.6, 6.7 and 6.8 present the comparative results of each algorithm in the KITTI odometry dataset. The results of LOAM on KITTI dataset outside the brackets are obtained from [42], and those in the brackets are obtained by running the open sourced code. The results of LeGO-LOAM and LIMO are both obtained by running the code. From the experiments, we can see that our method can deal with urban, highway and country environments well and obtain satisfactory accuracy for all sequences. The results of LOAM in [42] are good but when we run the open

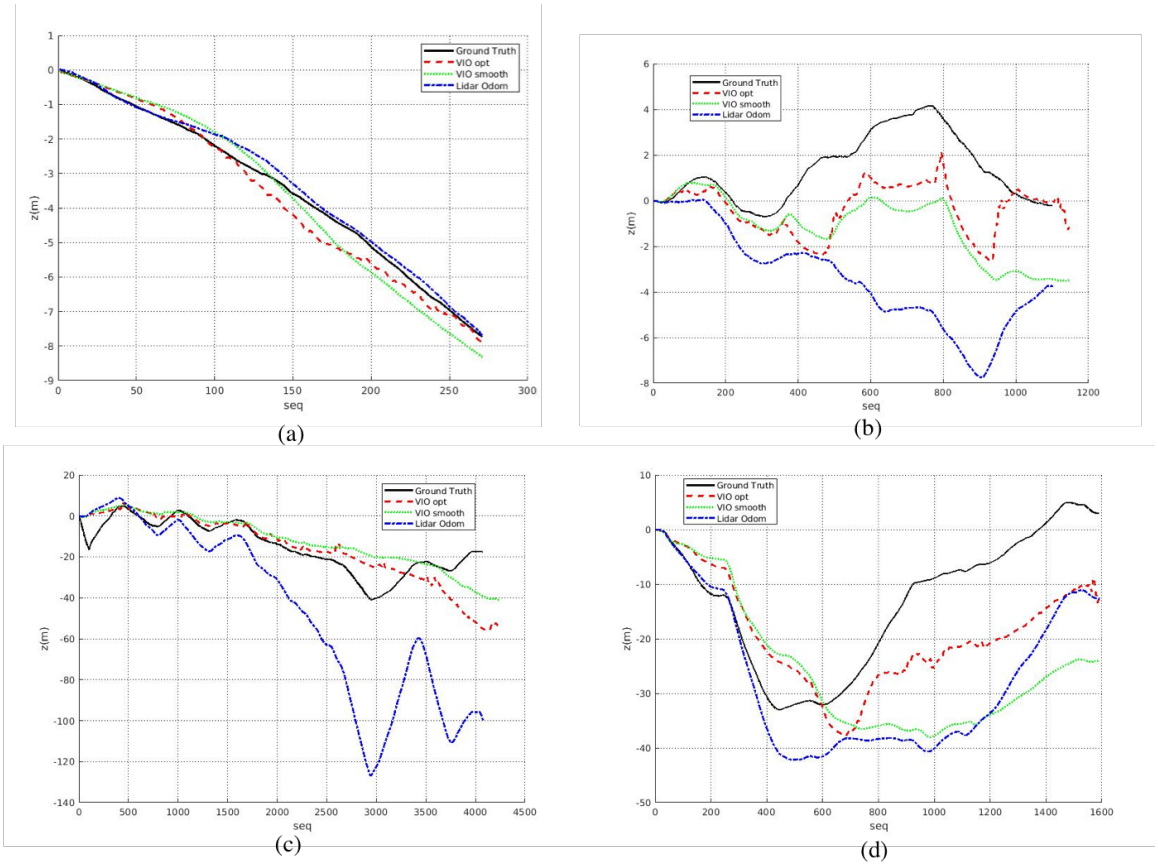

Figure 6.3: VIO and Lidar odometry results of KITTI (Z Value) (a) 04 sequence (b) 07 sequence (c) 08 sequence (d) 09 sequence

sourced code, the performance for the highway is bad because the lacking of edge features causes the degeneration of Lidar odometry. LeGO-LOAM, which is designed for lightweight real-time Lidar odometry, has the lowest accuracy. The results of LIMO are terrible for some sequences such as 6 and 10 due to it is easily influenced by the outliers from the moving objects. Furthermore, from the results, we can also conclude that it's challenging for Lidar-based methods to work well in highway environments because the lacking of edge features. As shown in Fig. 6.9, only a few number of edge feature points are extracted, which is difficult for the ICP alignment. For the sequence 07, all algorithms work well because the abundance of edge features as shown in Fig. 6.10. Theoretically, The degeneracy of the Lidar odometry could be detected by computing the minimum eigenvalue λ_1 of $\boldsymbol{J}^T\boldsymbol{J}$, where $\boldsymbol{J}$ is the Jacobian

Table 6.1: Mean translation error (%) on the KITTI odometry dataset

Sequence	Environments	Our method	LOAM	LeGO-LOAM	LIMO
00	Urban	0.89	**0.78**(0.94)	1.76	0.81
01	Highway	**1.35**	1.43(20.14)	29.25	3.89
02	Urban + Country	1.30	**0.92**(2.46)	7.95	2.10
04	Country	**0.46**	0.71(0.85)	1.01	0.75
05	Urban	0.60	**0.57**(0.88)	4.96	1.04
06	Urban	**0.41**	0.65(0.56)	2.81	11.07
07	Urban	**0.56**	0.63(0.64)	1.46	2.07
08	Urban + Country	**0.96**	1.12(1.21)	1.51	1.36
09	Urban + Country	0.79	**0.77**(1.45)	1.68	10.12
10	Urban + Country	**0.76**	0.79(2.34)	9.49	293.87

of the nonlinear of the optimization system. Fig. 6.11 shows the λ_1 for sequence 00 and 07, respectively. We can see the minimum eigenvalue is smaller than 100 many times in 01 sequence but all larger than 100 in 07 sequence. Due to the odometry selection strategies used in this thesis, the VIO source odometry would be used in the Lidar odometry degenerate environments. In contrast, when the dataset misses the IMU measurements sometimes (KITTI 00, 02, 05), the odometry selection module would select the Lidar odometry source at this time. The selected odometry status for KITTI 00 and 01 are shown in Fig. 6.12 and Fig. 6.13, where 0 represents the IMU-rate VIO, 1 denotes the corrected IMU-rate smooth odomerry and 2 represents the Lidar odometry. For KITTI sequence 00, at some time, the visual feature number of VIO is 0 where the IMU message is lost causing the VIO fail. In this way, the odometry selection module would select Lidar odometry as the odometry source. For KITTI sequence 01, the visual feature number for VIO at some time is small and the odometry selection module would choose Lidar odometry when the robot is not in the Lidar degenerate case. Due to the flexible odometry selection, our algorithms can handle the sensor failure and degenerate cases and obtain the satisfactory performance on the KITTI odometry dataset.

We also use our platforms to collect some indoor and outdoor datasets to further evaluate the accuracy and robustness of our proposed method. As shown in Fig.

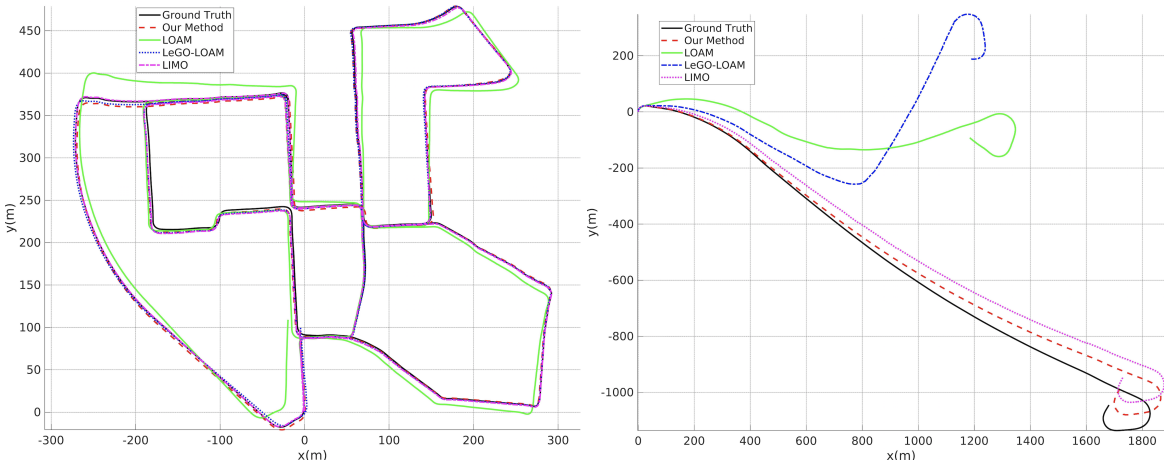

Figure 6.4: Trajectory results for KITTI sequence 00 and 01

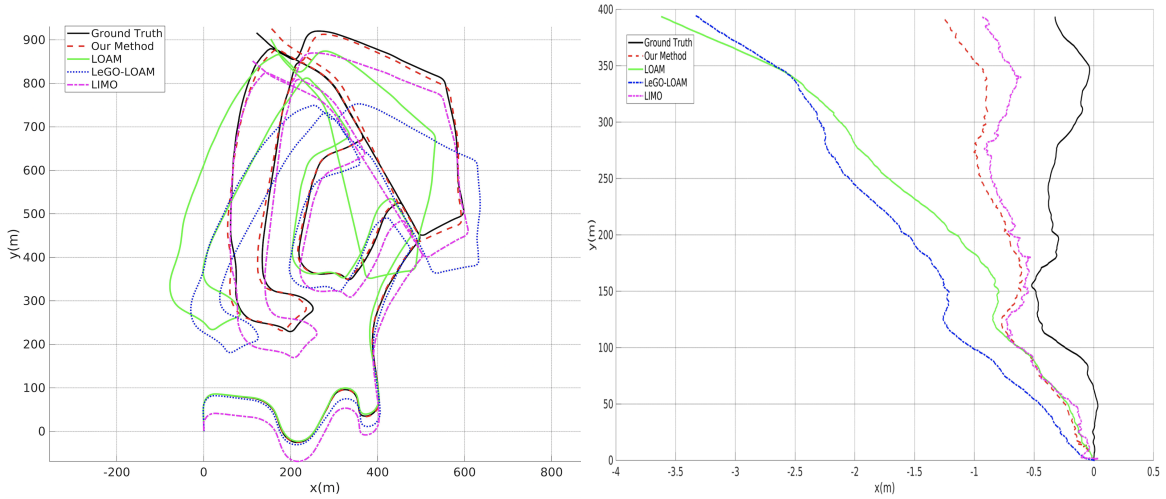

Figure 6.5: Trajectory results for KITTI sequence 02 and 04

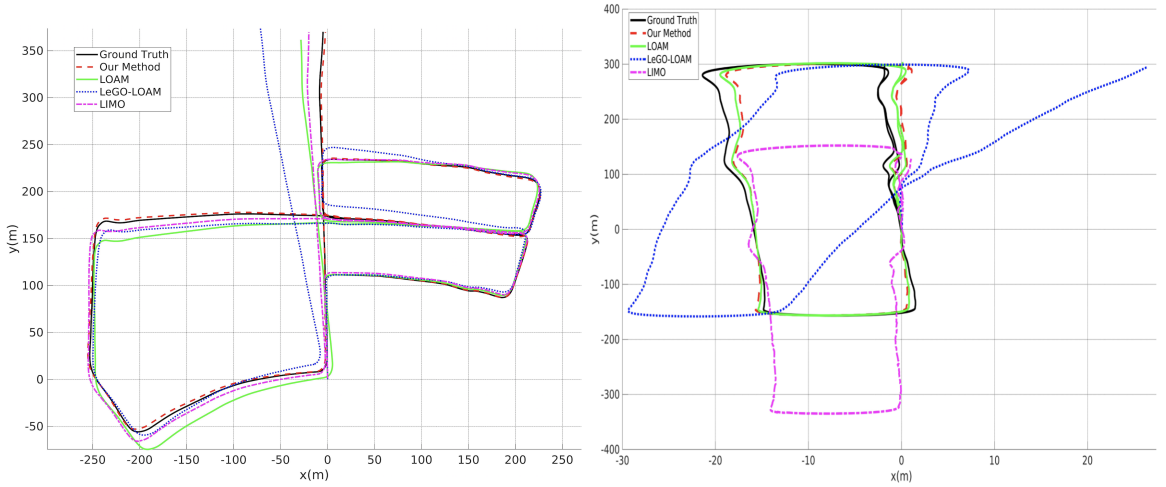

Figure 6.6: Trajectory results for KITTI sequence 05 and 06

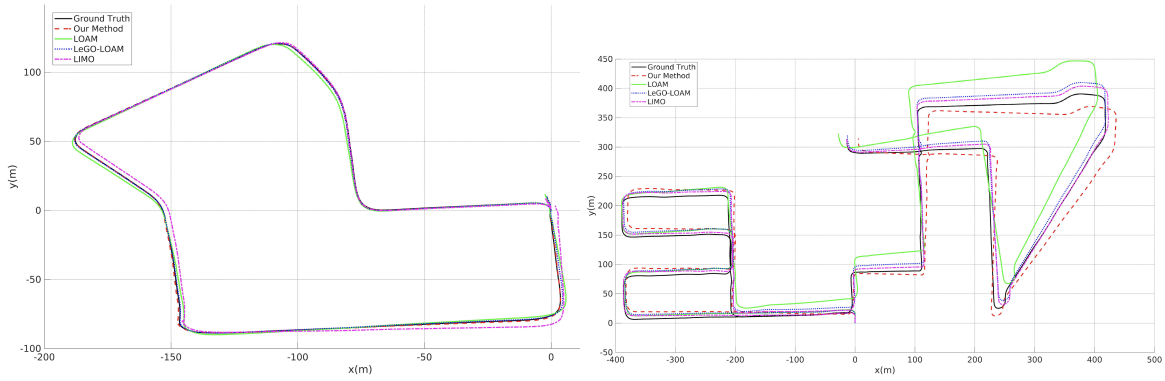

Figure 6.7: Trajectory results for KITTI sequence 07 and 08

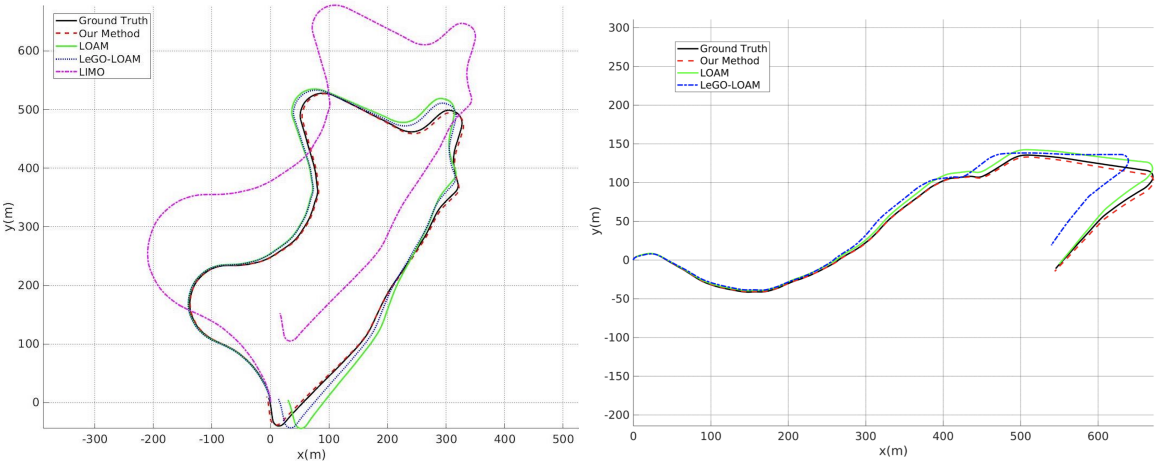

Figure 6.8: Trajectory results for KITTI sequence 09 and 10

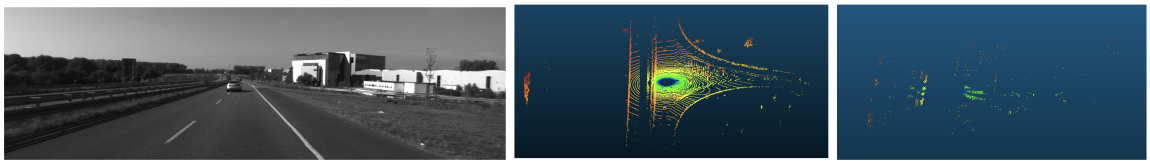

Figure 6.9: Degenerate environments for Lidar (a) Image (b) Full Lidar scan (c) edge feature points

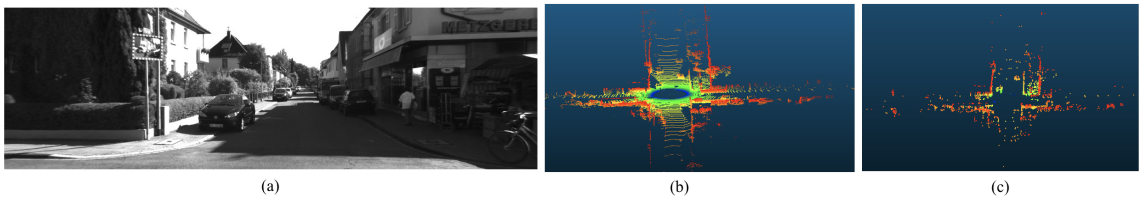

Figure 6.10: Normal environments for Lidar (a) Image (b) Full Lidar scan (c) edge feature points

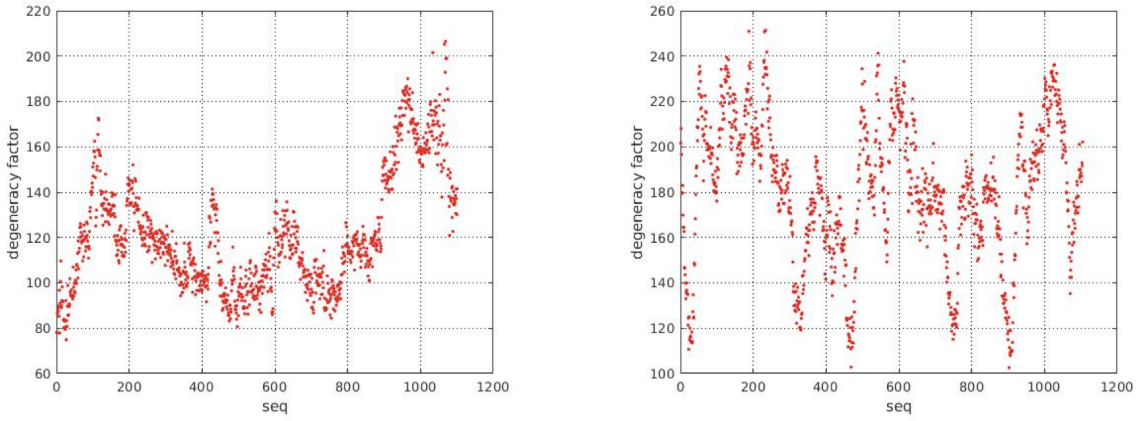

Figure 6.11: Degeneracy Factor of Lidar Odometry (a) 00 sequence (b) 07 sequence

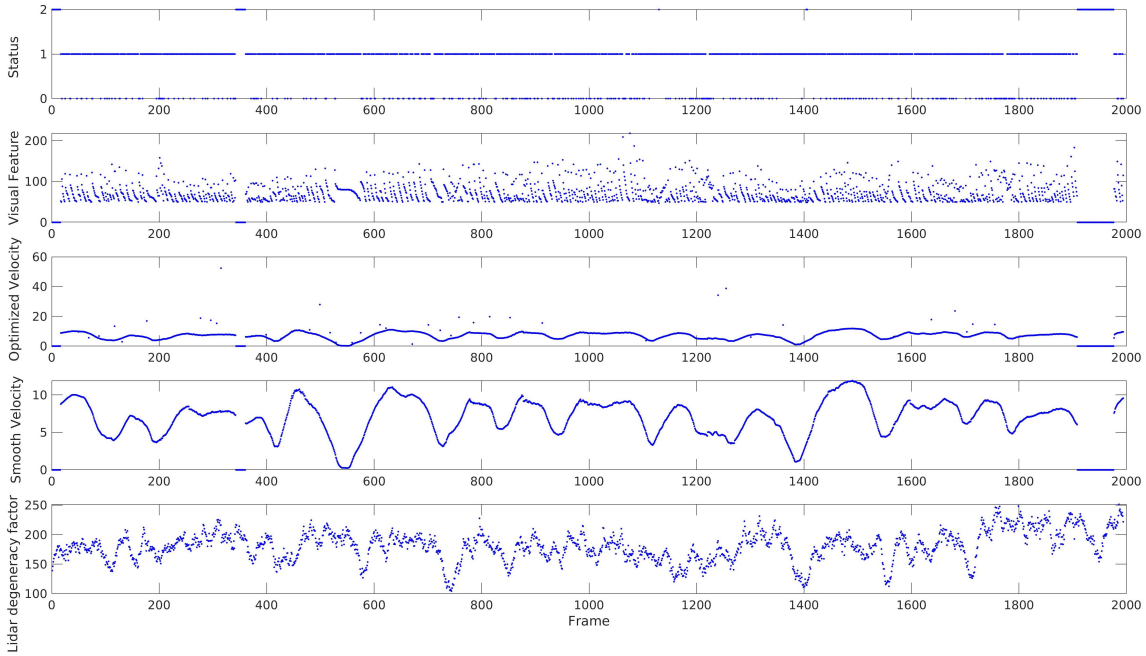

Figure 6.12: Odometry selection for KITTI sequence 00

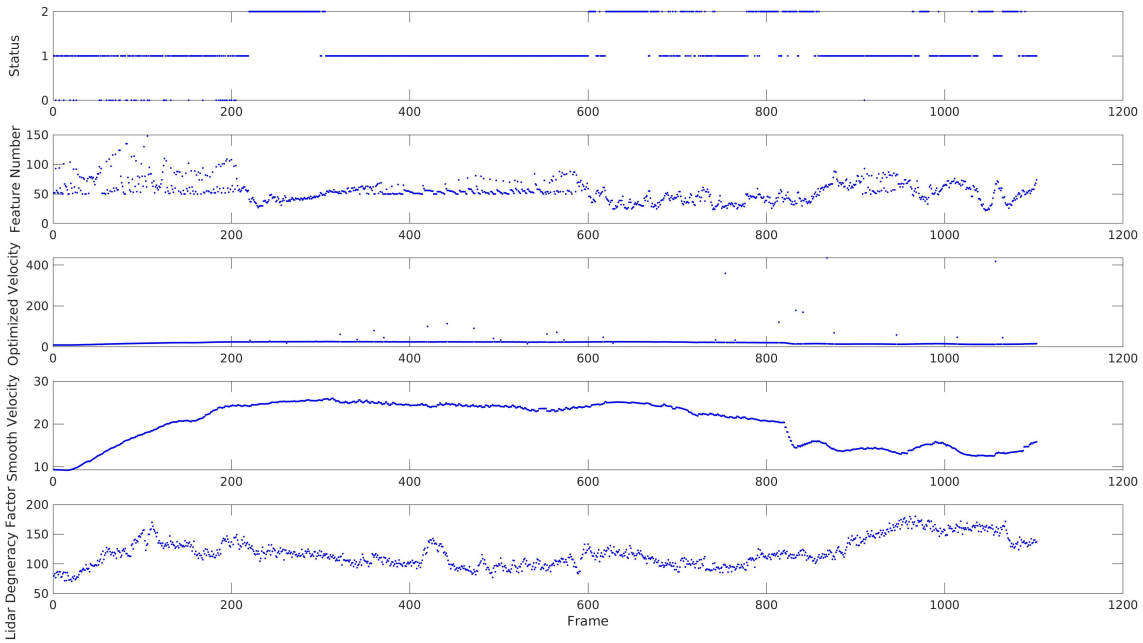

Figure 6.13: Odometry selection for KITTI sequence 01

6.14, we build one platform with four cameras, a Velodyne 16-line Lidar and an IMU. Sensors and the computer of the platform are time synchronized based on one custom microcontroller. We use this platform to collect two challenging indoor datasets including featureless and structureless long hallways and cluttered warehouse. Furthermore, one FARO Focus3D survey lidar scanner is utilized to get the dense 3D ground truth point cloud of this dataset as shown in Fig. 6.15 from different viewpoints. Fig. 6.16 shows the results of our method compared with LOAM in the challenging indoor environment. From the results, we can see that our methods can deal with featureless and structureless environments well but LOAM fails. The robot finally returns to the starting position, thus the final drift can be computed. For the indoor huge loop dataset, the drift of our method is 8 centimeters but the final drift is 1.43 meters for the indoor inverse huge loop dataset. In the first indoor dataset, the robot returns to the region of the warehouse where the map has been previously built, and the drift is eliminated by the scan to map optimization. To evaluate the absolute error, the 3D point cloud model generated by our algorithm is first aligned with the

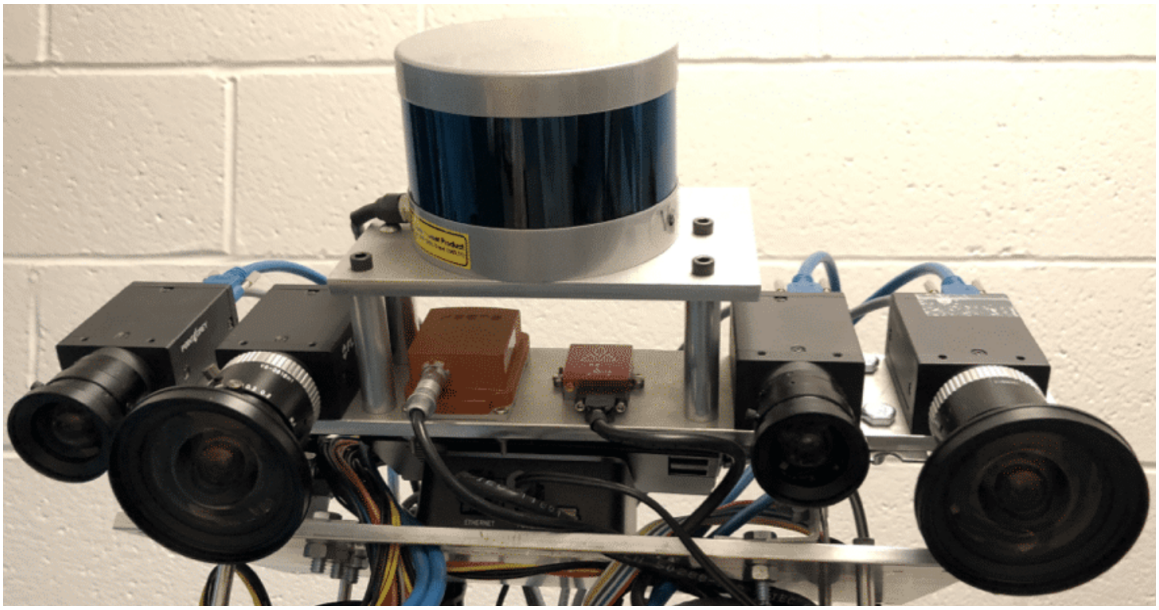

Figure 6.14: Data collection platform

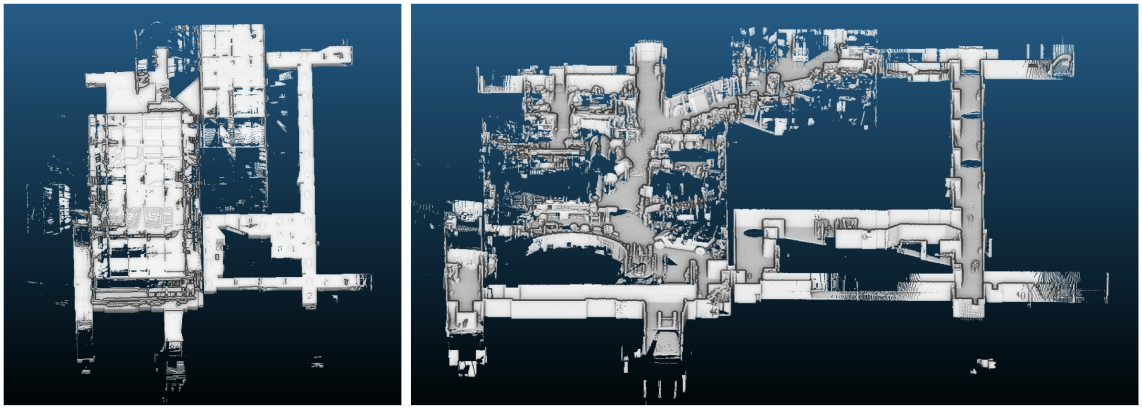

Figure 6.15: Ground truth point cloud of indoor dataset

dense 3D ground truth point cloud, and the Euclidean distance between each point in our model and its closest point in the model is computed. The map registration error for the indoor huge loop dataset is shown in Fig. 6.17, and the color in the image represents the error. It should be noted that almost all red regions (error $>$ 50 centimeters) are due to the missing objects caused by the occlusion of the Faro scans. From the result, we can see the Euclidean distance errors of most point clouds are smaller than 20 centimeters, demonstrating the accuracy of our approach in the challenging indoor environments.

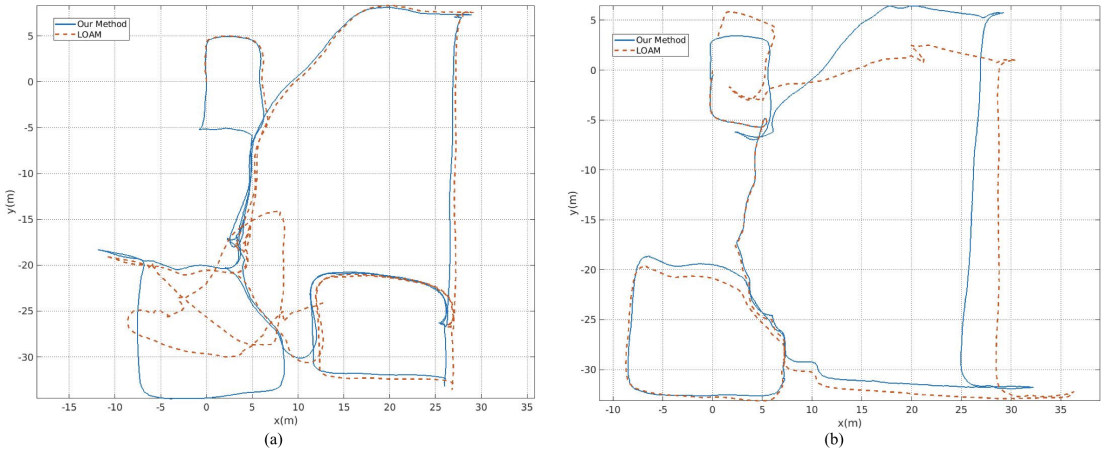

Figure 6.16: Trajectory results for the indoor test (a) Indoor huge loop (b) Inverse indoor huge loop

Table 6.2: RMSE of ATE (m) on the off-road driving dataset

Location	Distance (m)	Our method	LeGO-LOAM	ORB-SLAM2	EKF
Triangle loop	684	**0.87**	2.05	1.88	2.88
Slope loop	657	**1.40**	2.23	7.15	2.69

Finally, our algorithm is tested with several off-road driving datasets. The sensors used in the off-road vehicle contain one stereo camera, one 64-line Velodyne Lidar, one IMU, and the RTK GPS that provides the ground truth measurements. The off-road vehicles would traverse harsh terrain as shown in Fig. 6.18 and it's challenging for vehicle state estimation due to the lack of distinct structured landmarks. We evaluate our approach and compare with LeGO-LOAM which is designed for the variable terrains in off-road driving dataset, and the resulting trajectories are shown in Fig. 6.19. We use the RMSE of Absolute Trajectory Error (ATE) to evaluate the results which are shown in Table 6.2. Results of ORB-SLAM2 and EKF fusing GPS measurements are from [5]. The experimental results demonstrate our approach can also work well in the off-road driving environments.

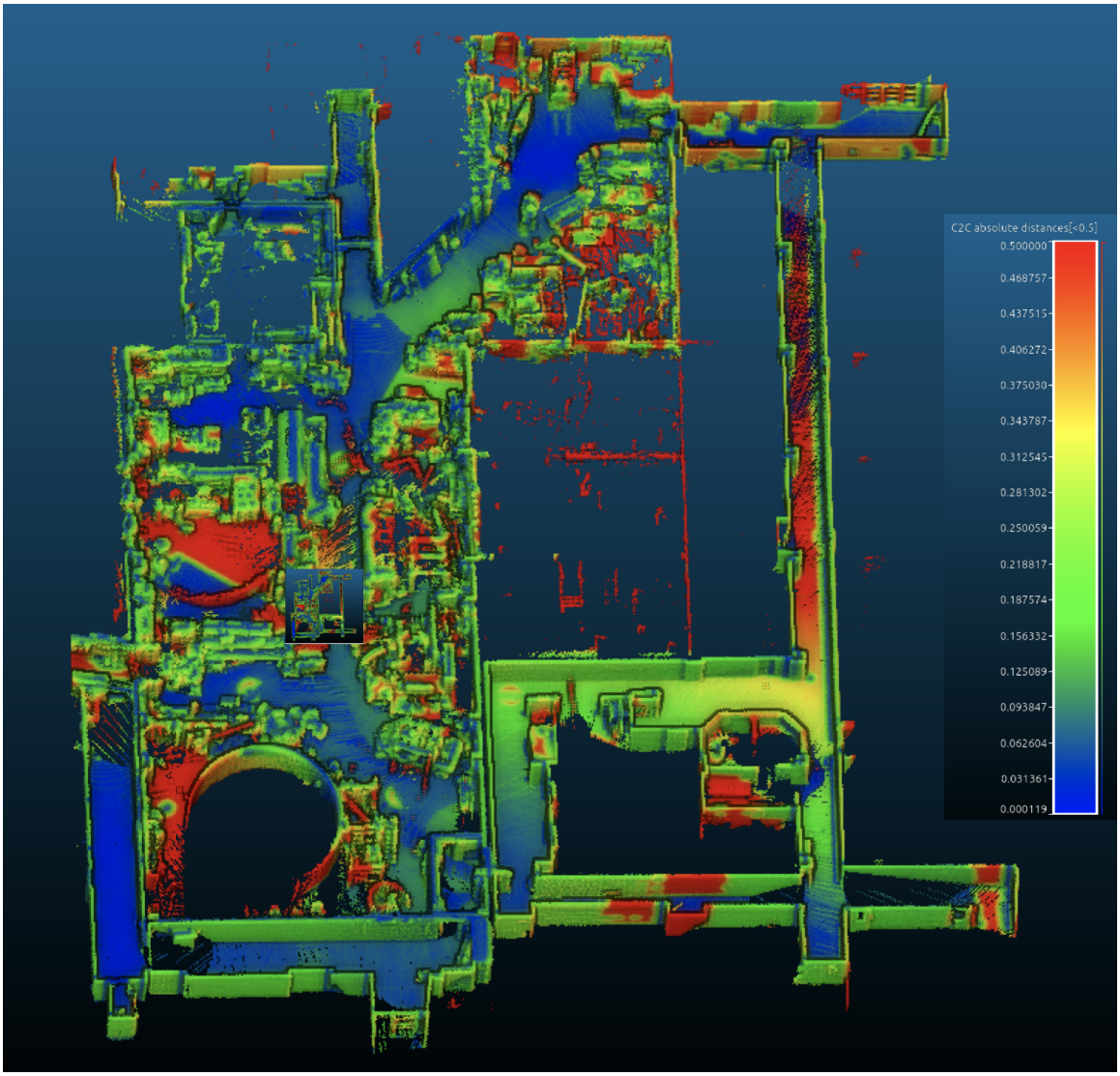

Figure 6.17: The map registration error for the indoor huge loop dataset

Figure 6.18: Images of off-road environment

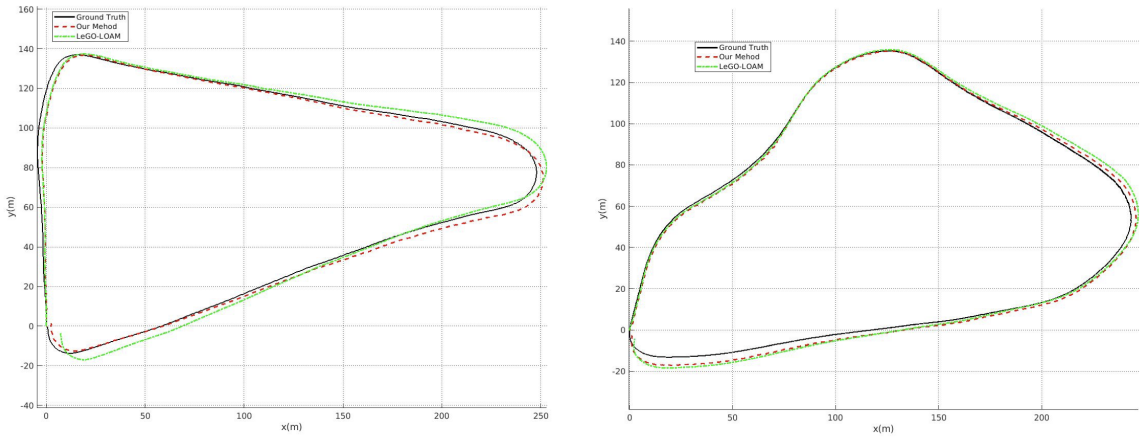

Figure 6.19: Our method, LeGO-LOAM and ground-truth at (a) triangle loop (b) slope loop

6.2 Loop Closure Test

In this thesis, the loop closure module of our proposed SLAM system is evaluated using KITTI, indoor and off-road driving datasets. For the KITTI odometry dataset, we choose sequence 00 and 05 to evaluate the effect of loop closure correction. The trajectories before and after loop clousre are shown in Fig. 6.20. The loop closure results for the off-road driving are shown in Fig. 6.21. Table 6.3 presents the comparative RMSE of ATE on trajectories before and after loop closure correction on the KITTI and off-road driving dataset. For the indoor inverse huge loop dataset, the final drift exists generating the ghost map. The map registration error for the indoor inverse huge loop dataset before loop closure is shown in 6.22. After loop-closure correction, the map is corrected based on the optimized pose graph, and we compare the corrected map with the ground truth point cloud as shown in Fig. 6.23. From the results, we can see that the errors of most points are smaller than 20 centimeters, demonstrating the effectiveness of our loop closure module.

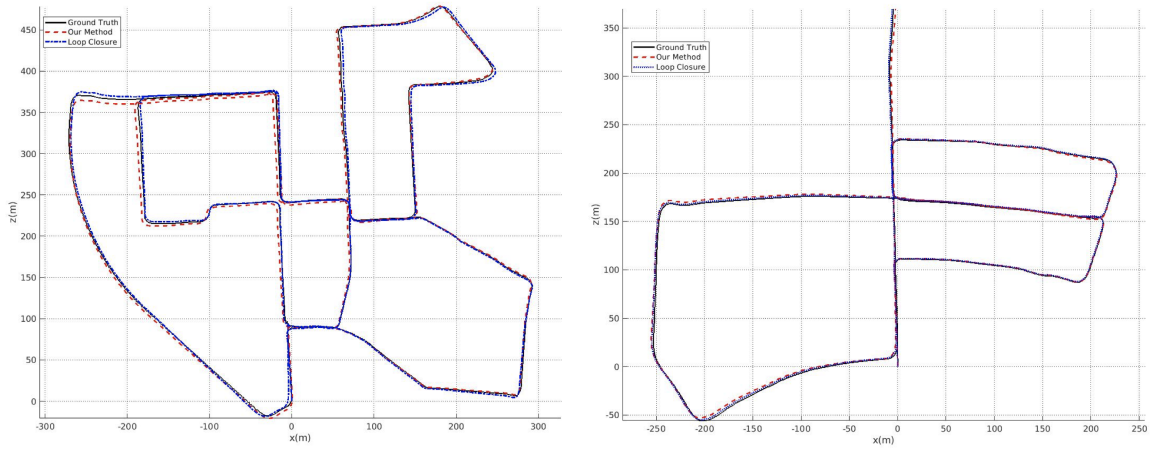

Figure 6.20: Trajectory results before and after loop closure (a) 00 sequence (b) 05 sequence

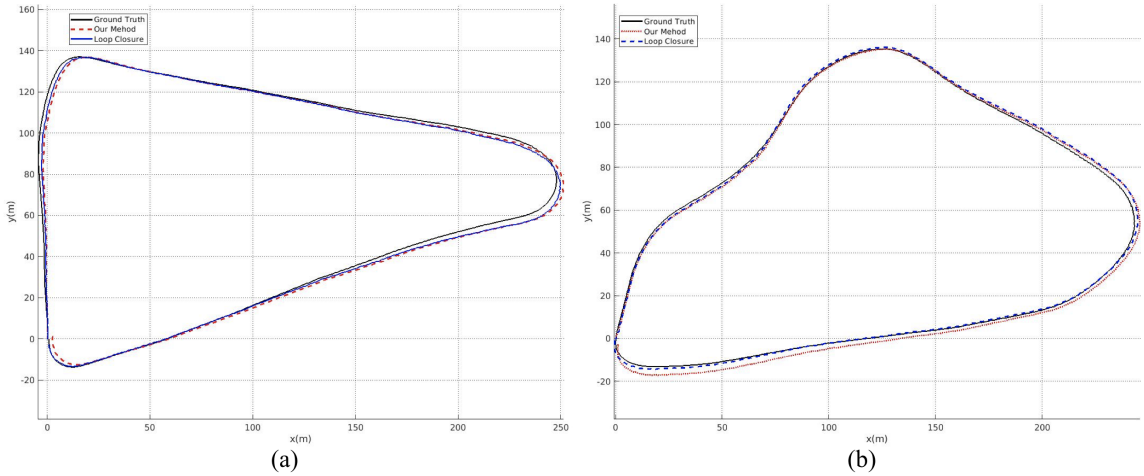

Figure 6.21: Trajectory results before and after loop closure (a) Triangle Loop (b) Slope Loop

Table 6.3: Comparison of our method before and after loop-closure

Location	Distance (m)	Before loop-closure	After loop-closure
KITTI 00	3714	4.91	1.88
KITTI 05	2223	1.73	1.04
Triangle loop	684	0.87	0.69
Slope loop	657	1.40	1.05

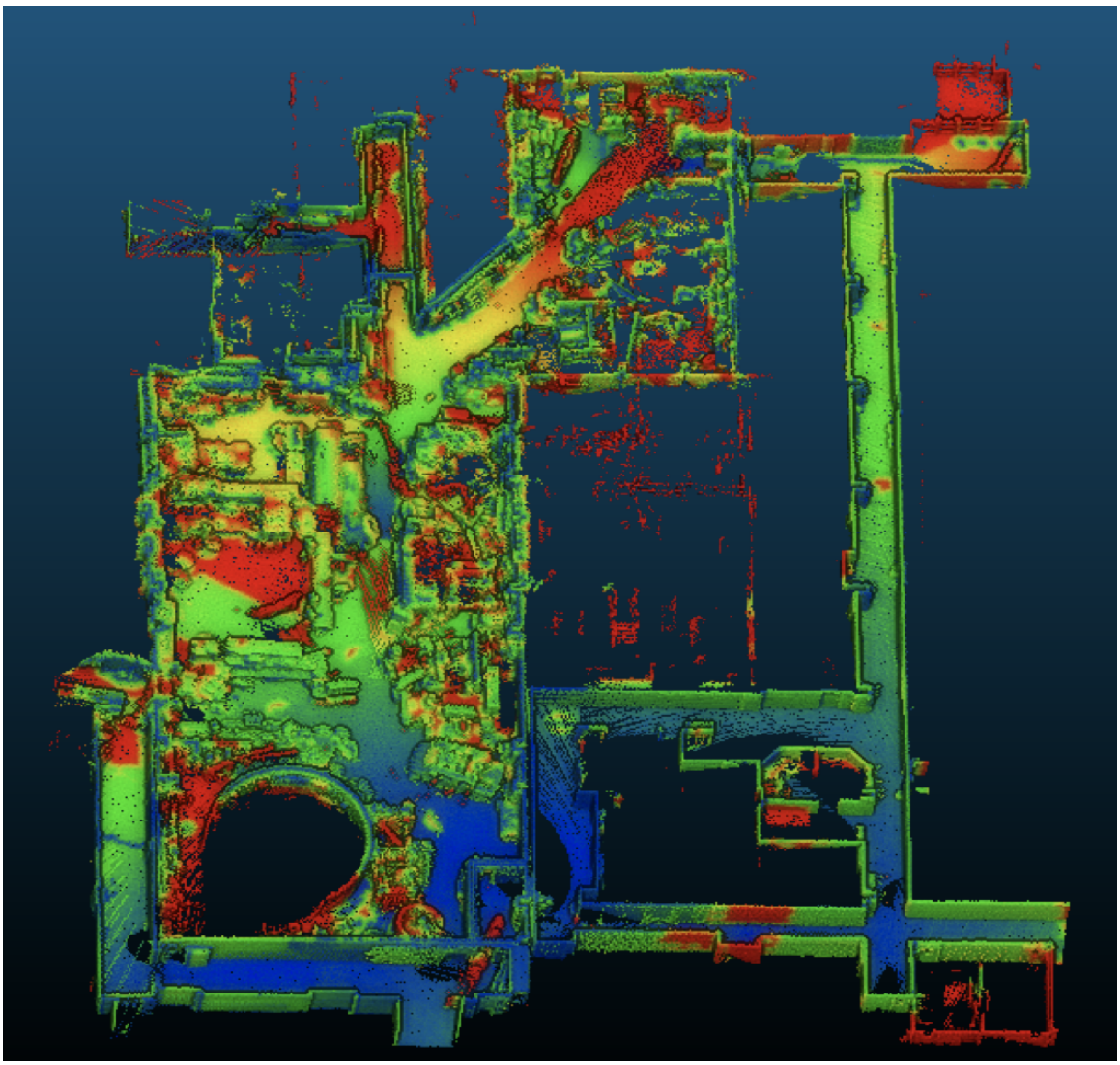

Figure 6.22: The map registration error for the indoor inverse huge loop dataset before loop closure

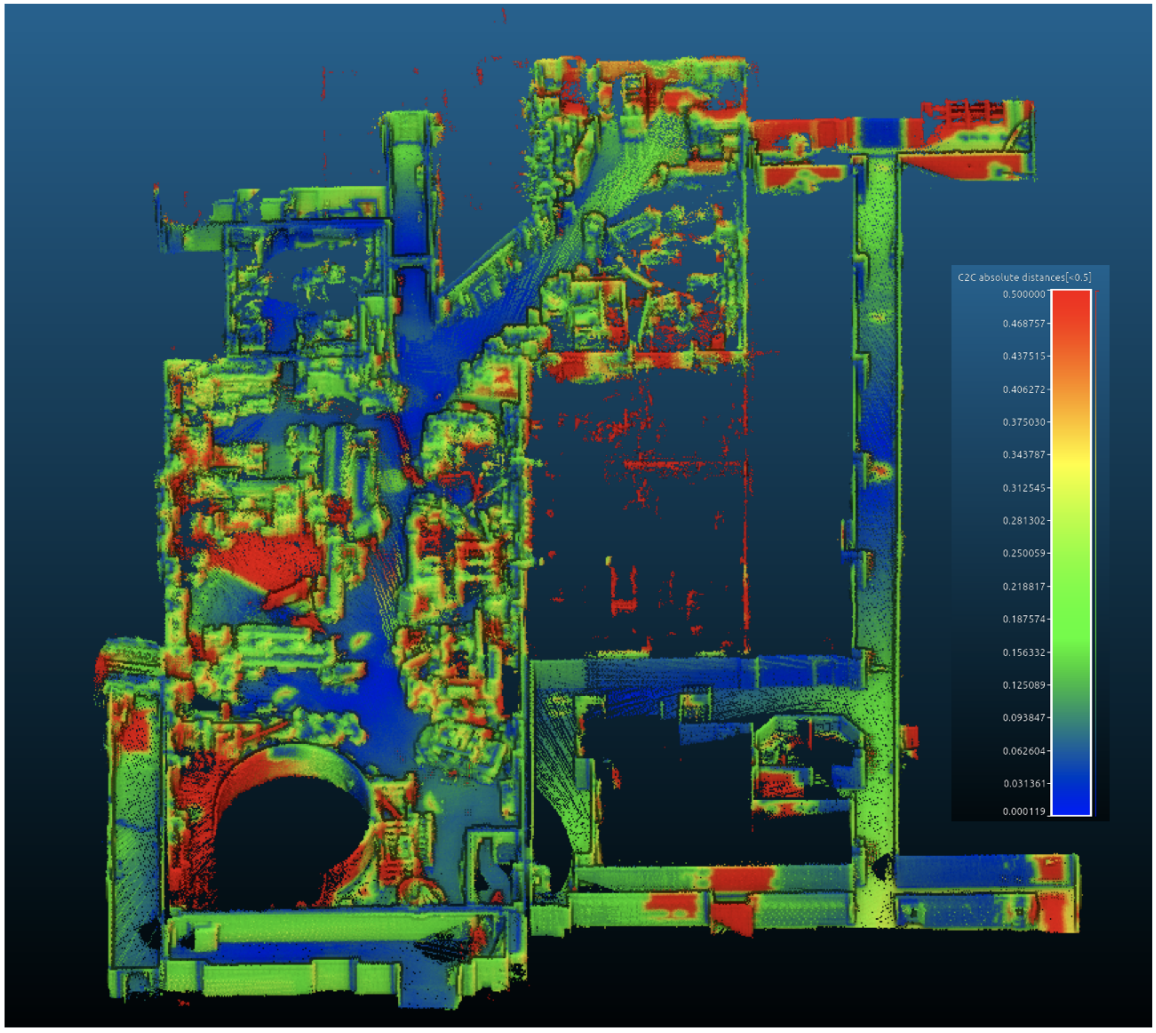

Figure 6.23: The map registration error for the indoor inverse huge loop dataset after loop closure

6.3 Semantic 3D Occupancy Mapping Results

In this section, we use the KITTI dataset to construct large-scale semantic 3D occupancy maps. For each image, we first utilize the dilated CNN to get the semantic label of each pixel in the image as shown in Fig. 6.24. Then, as shown in Fig. 6.25, the Lidar point clouds are projected to the image to get the semantic labels distribution. For each 3D Lidar point, the standard Bayes' update rule is utilized to refine the semantic label. Fig. 6.26 shows the qualitative results of our map. Our method can successfully reconstruct the objects such as cars and trunk of trees. Furthermore,

(a) (b)

Figure 6.24: 2D semantic segmentation results (a) Original image (b) 2D semantic segmentation using CNN

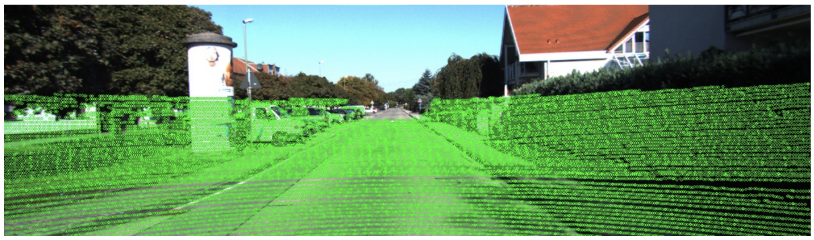

Figure 6.25: Lidar-Image association results

by fusing semantic labels of multiple frames, the 3D mapping can have more accurate semantic distribution for the point clouds. As shown in Fig. 6.27, the fence within the rectangular area is classified as the wrong object, but the fence is continuous and assigned with the right label in the 3D space.

The semantic 3D occupancy maps of KITTI sequence 01, 05 and 06 are shown in Fig. 6.28, Fig. 6.29 and Fig. 6.30, respectively. For each sequence, the zoom-in views are also added to visualize the details of the 3D map. From the reconstruction results, we can see our method can deal with large-scale environments reconstruction

(a) (b)

Figure 6.26: Results of 3D semantic mapping (a) Image for current scene (b) 3D semantic mapping results

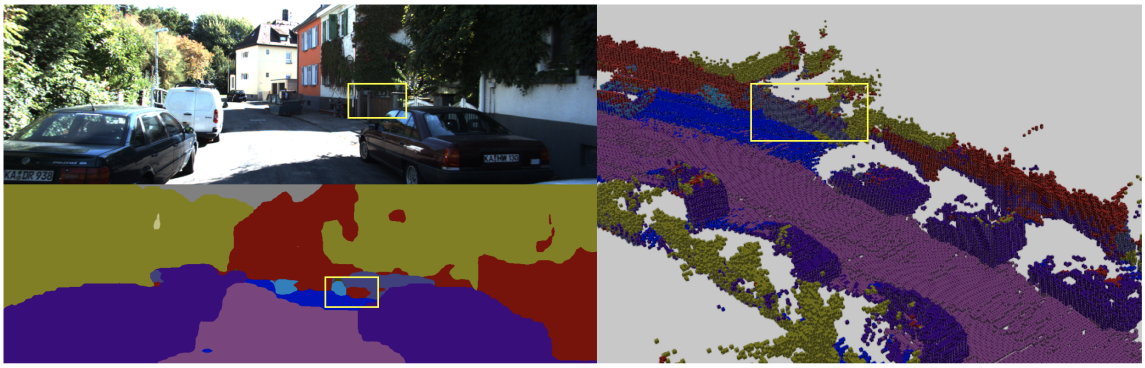

Figure 6.27: Examples showing the advantage of 3D semantic mapping

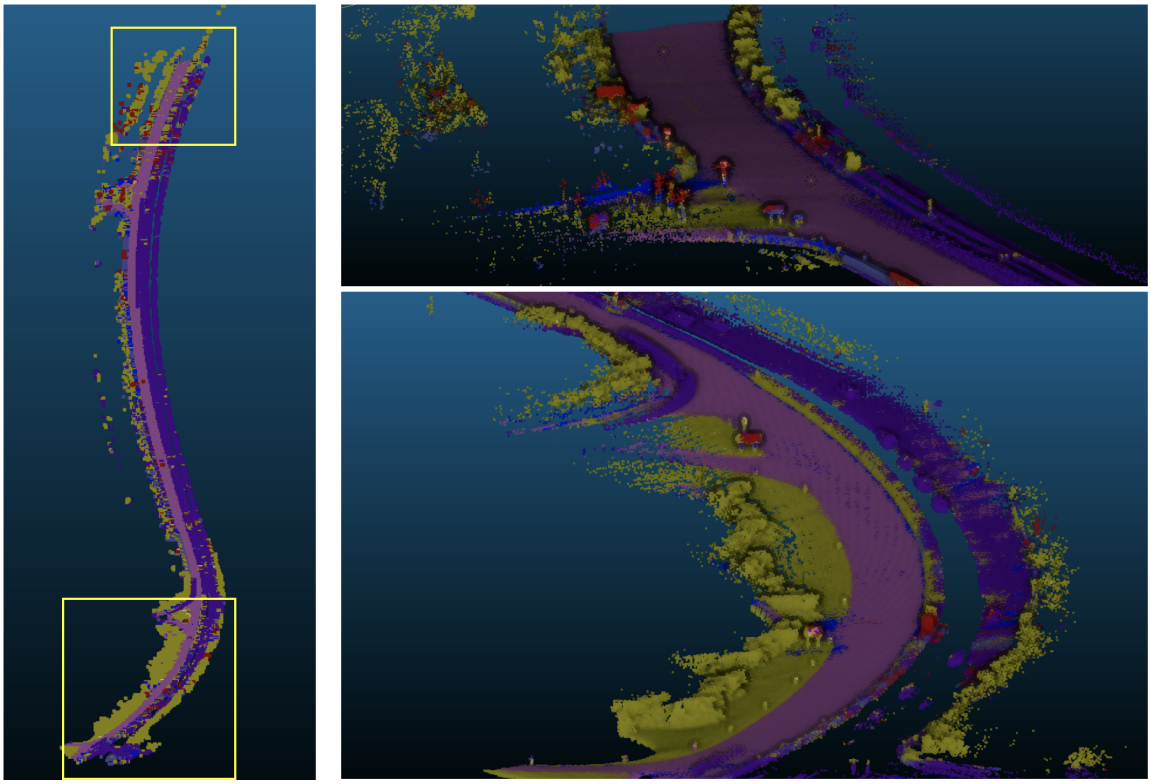

Figure 6.28: Semantic 3D Occupancy mapping for KITTI 01

well.

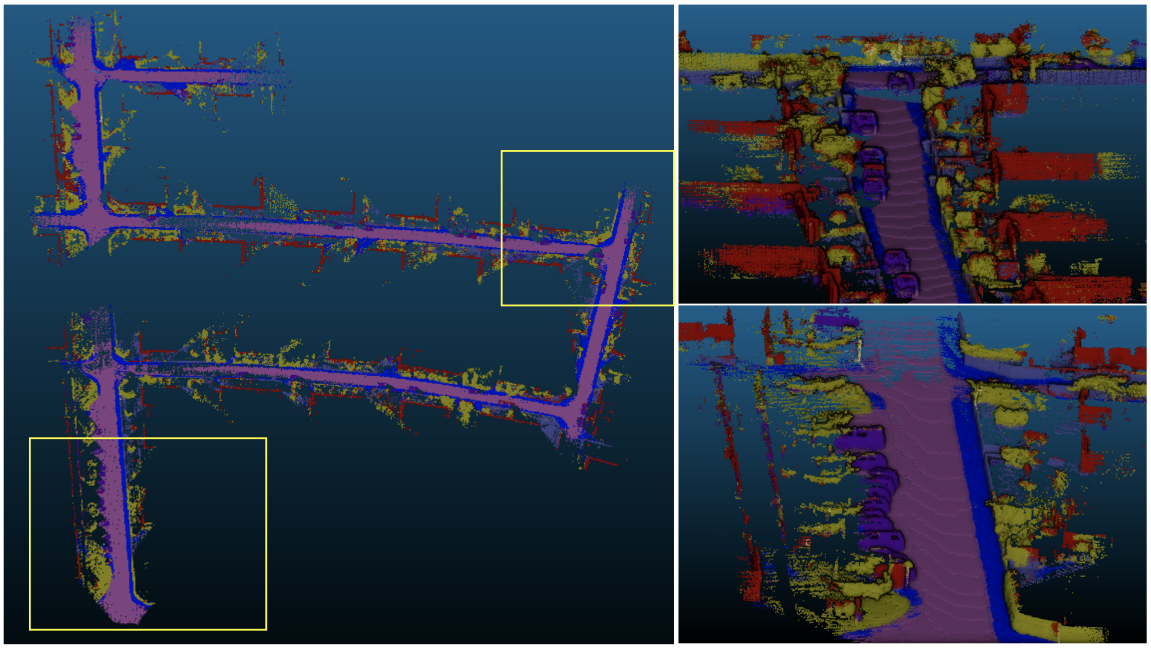

Figure 6.29: Semantic 3D Occupancy mapping for KITTI 05

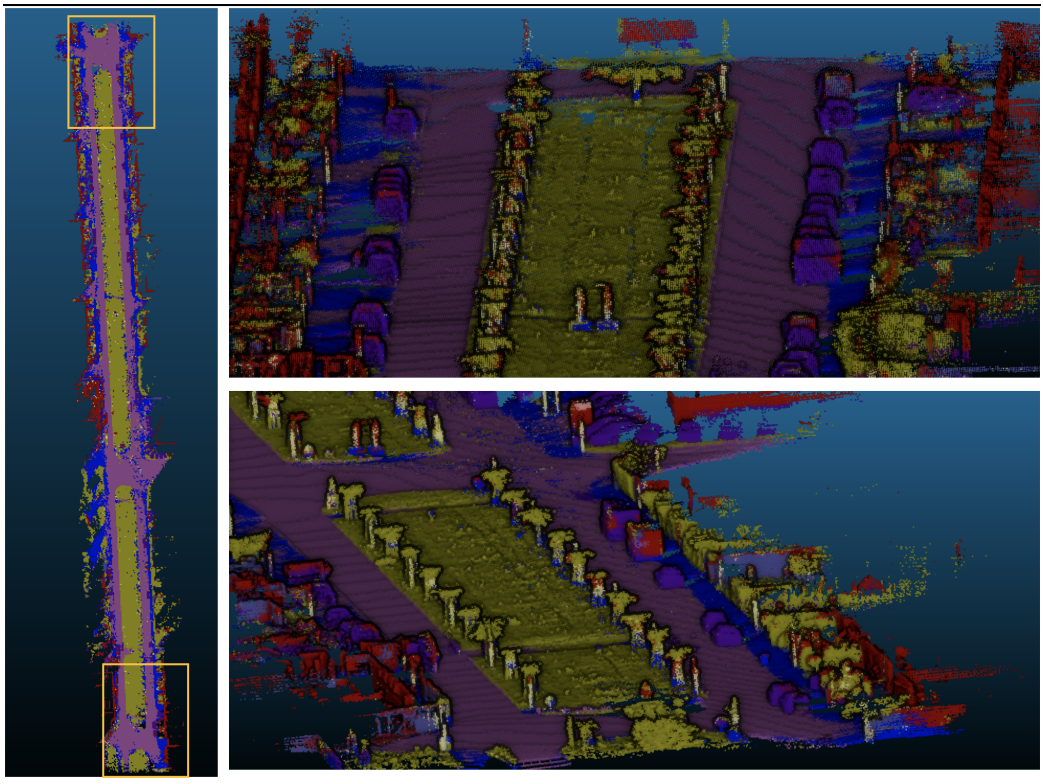

Figure 6.30: Semantic 3D Occupancy mapping for KITTI 06

Chapter 7

Conclusion and Future Work

This thesis utilizes multi-sensor fusion for robust simultaneous localization and mapping. To achieve robust and accurate state estimation, the monocular VIO with Lidar feedback is implemented to output IMU-rate odometry. Furthermore, another Lidar odometry fusing visual measurements is used as another odometry source. Then, the odometry selection framework selects the appropriate odometry source, which is used as the initial estimation of the Lidar mapping module. Finally, the refined pose of the whole system is obtained in the Lidar mapping module. Series of experiments are conducted to evaluate the accuracy and the robustness of our approach. The comparative experiments on KITTI odomtry dataset demonstrate that our algorithm can work well in different environments including urban, country and highway. Even when the sensor failure exists such as missing IMU message, our algorithm can still work. Furthermore, we also test our approach in challenging customized datasets including structureless and featureless environments, and the experimental results show that our algorithm performs robustly in these datasets.

We also explore the loop closure correction using the measurements from camera and Lidar. The DBOW3 is used to detect the loop candidate frames, then the feature matching between the current frame and each loop candidate is conducted and the

GMS filter is utilized to eliminate the feature matching outliers. Once the loop candidate is confirmed, the point to plane ICP between sparse point cloud of the current frame and loop frame is used to get the relative transformation. Finally, the pose graph optimization based on iSAM2 is used to obtain the corrected pose graph. We test our proposed loop closure module in KITTI and our collected odometry datasets. The experimental results demonstrate that our loop closure module can reject the false-positive loop candidates and the trajectories after loop closure are close to the ground-truth having smaller absolute translation error.

Finally, we propose one pipeline using the visual and Lidar measurements to construct the semantic 3D occupancy map. Experimental results on the KITTI dataset shows that our pipeline can construct the accurate large-scale semantic map.

For future work, there are several potential research directions:

1. Design one tightly-coupled optimization framework to fully utilize the measurements of camera, IMU and Lidar. In this thesis, the monocular VIO only uses the Lidar pose feedback information and the relative Lidar 6-DOF transformation is added into the factor graph. However, the optimization error term could be directly obtained from the Lidar point cloud, and these optimization residues could be added into the factor graph to achieve the visual-IMU-Lidar tightly-coupled optimization.

2. Utilization of the semantic information. In this thesis, we only construct the semantic map. But semantic information is very useful for the high-level tasks of robots. For example, the path planning algorithm can be developed based on our semantic 3D occupancy map to let robot avoid obstacles and walk on the roads. Furthermore, the semantic information can be used for the loop closure detection to deal with extreme viewpoint variations.

Bibliography

[1] Marhaban M.H. Saripan M.I. Aqel, M.O. and N.B. Ismail. Review of visual odometry: types, approaches, challenges, and applications.

[2] Lin W.Y. Matsushita Y. Yeung S.K. Nguyen T.D. Bian, J. and M.M. Cheng. GMS: grid-based motion statistics for fast, ultra-robust feature correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4181–4190, 2017.

[3] M. Bosse and R. Zlot. Place recognition using keypoint voting in large 3D lidar datasets. In *IEEE International Conference on Robotics and Automation*, pages 2677–2684, 2013.

[4] Carlone L. Carrillo H. Latif Y. Scaramuzza D. Neira J. Reid I. Cadena, C. and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, pages 1309–1332, December 2016.

[5] Chou, W. State Estimation and Vision-based Occupancy Mapping for Off-Road Driving. `https://www.ri.cmu.edu/wp-content/uploads/2017/08/weihsinc_thesis_final.pdf`, August 2017.

[6] F. Dellaert. *Factor graphs and GTSAM: A hands-on introduction.* Georgia Institute of Technology, 2012.

[7] J.E. Deschaud. IMLS-SLAM: scan-to-model matching based on 3D data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2480–2485, 2018.

[8] Koltun V. Engel, J. and D. Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, pages 611–625, March 2018.

[9] Schöps T. Engel, J. and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European conference on computer vision)*, pages 834–849, 2014.

[10] Yang S. Jain S. Dubey G. Roth S. Maeta S. Nuske S. Zhang Y. Fang, Z. and S. Scherer. Robust autonomous flight in constrained and visually degraded shipboard environments. *Journal of Field Robotics*, pages 25–52, September 2016.

[11] D. Filliat. A visual bag of words method for interactive qualitative localization and mapping. In *IEEE International Conference on Robotics and Automation*, pages 3921–3926, 2007.

[12] Carlone L. Dellaert F. Forster, C. and D. Scaramuzza. On-Manifold Preintegration for Real-Time Visual-Inertial Odometry. *IEEE Transactions on robotics*, pages 1–21, October 2017.

[13] Pizzoli M. Forster, C. and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE international conference on robotics and automation*, pages 15–22, 2014.

[14] Lenz P. Stiller C. Geiger, A. and R. Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, pages 1231–1237, August 2013.

[15] Roser M. Geiger, A. and R. Urtasun. Efficient large-scale stereo matching. In *Asian conference on computer vision*, pages 25–38, 2010.

[16] Wilczynski A. Graeter, J. and M. Lauer. Limo: Lidar-monocular visual odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7872–7879, 2018.

[17] Schön T.B. Nieto J.I. Granström, K. and F.T. Ramos. Learning to close loops from range data. *The international journal of robotics research*, pages 1728–1754, June 2011.

[18] D. Gálvez-López and J.D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, pages 1188–1197, October 2012.

[19] Jacobson A. Hausler, S. and M. Milford. Multi-Process Fusion: Visual Place Recognition Using Multiple Image Processing Methods. *IEEE Robotics and Automation Letters*, pages 1924–1931, April 2018.

[20] Krainin M. Herbst E. Ren X. Henry, P. and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 2010.

[21] K. Huang and C. Stachniss. Joint Ego-motion Estimation Using a Laser Scanner and a Monocular Camera Through Relative Orientation Estimation and 1-DoF ICP. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 671–677, 2018.

[22] M. Kaess. Simultaneous localization and mapping with infinite planes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611, 2015.

[23] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, 2007.

[24] Li Y. Dellaert F. Li F. Kundu, A. and J.M. Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *European Conference on Computer Vision)*, pages 703–718, 2014.

[25] J.H. Manton. Optimization algorithms exploiting unitary constraints. *IEEE Transactions on Signal Processing*, pages 635–650, March 2002.

[26] R. Mur-Artal and J.D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on robotics*, pages 1255–1262, October 2017.

[27] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, pages 756–777, June 2004.

[28] Li P. Qin, T. and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on robotics*, pages 1004–1020, August 2018.

[29] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443, 2006.

[30] Rabaud V. Konolige K. Rublee, E. and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[31] T. Shan and B. Englot. LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765, 2018.

[32] Vijayarangan S. Li C. Shao, W. and G. Kantor. Stereo Visual Inertial LiDAR Simultaneous Localization and Mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, page pp., 2019.

[33] O'Brien J.F. Shen, C. and J.R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *ACM Siggraph 2005 Courses*, page 204, 2005.

[34] J. Shi and C. Tomasi. *Good features to track*. PhD thesis, Cornell University, 1993.

[35] Matthies L. Sibley, G. and G. Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, pages 587–608, August 2010.

[36] Grisetti G. Steder, B. and W. Burgard. Robust place recognition for 3D range data based on point features. In *IEEE International Conference on Robotics and Automation*, pages 1400–1405, 2010.

[37] Huang Y. Yang, S. and S. Scherer. Joint semantic segmentation and 3d reconstruction from monocular video. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 590–597, 2017.

[38] Wang R. Gao X. Yang, N. and D. Cremers. Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect. *IEEE Robotics and Automation Letters*, pages 2878–2885, June 2018.

[39] F. Yu and V. Koltun. Multi-Scale Context Aggregation By Dilated Convolutions.

[40] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Robotics: Science and Systems*, pages 1–10, 2007.

[41] J. Zhang and S. Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181, 2015.

[42] J. Zhang and S. Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, pages 25–52, February 2017.

[43] J. Zhang and S. Singh. Laser–visual–inertial odometry and mapping with high robustness and low drift. *Journal of Field Robotics*, pages 1242–1264, March 2018.

[44] Kaess M. Zhang, J. and S. Singh. On degeneracy of optimization-based state estimation problems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 809–816, 2016.