Robust State Estimation and Mapping for Autonomous Inspection

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Mechanical Engineering

Weikun Zhen

B.S., Mechanical Engineering, Tianjin University, P. R. China
M.S., Mechanical Engineering, Carnegie Mellon University, USA

Carnegie Mellon University
Pittsburgh, PA

December, 2021

Acknowledgments

I will first thank my advisor, chair of my thesis committee, Sebastian Scherer, for his guidance, patience, and support during the development of this work over the years. I am also thankful to all other committee members, Michael Kaess (CMU RI), Kenji Shimada (CMU MechE), and Luca Carlone (MIT), for the in-depth discussions and suggestions. It is such a great honor to work with you.

I am especially grateful to Yaoyu Hu and Huai Yu, with whom we build systems, design algorithms, and conduct experiments. I will not forget the days we spent together in the air lab. My thanks to Hayashi Daisuke, Junbin Yuan, Henry Zhang, Andrew Saba, Jingfeng Liu for your help with the bridge inspection project. My thanks to Sam Zeng and Zhen Dong for together exploring the possibility of bridge inspection in the very beginning. My thanks to Shichao Yang and Yu Song for all those insightful discussions to help me solve system and theoretic problems. My thanks to Silvio Maeta for helping with the successful demonstration for former president Barack Obama. I am truly grateful for the help from Geetesh Dubey, with whom I shared the office for years and from whom I learned so much. Thanks to Sanae Minick and Nora Kazour for the wonderful administrative support.

Thanks to Yaoyu Hu, Chen Wang, Zhongqiang Ren, and Peng Yin for offering such invaluable feedback on this thesis and the final presentation. Thanks to Chaohui Gong, Jin Dai, Yuanfeng Han, Xuexu Xiao, Ke Sun, Yang Ou. It has been such an incredible life experience at CMU having you all by my side. Thanks to Wenhao Luo, Chen Fu, Yiwei Lyu, and Yiming Zhang for bringing so much joy to the trip pursuing my PhD.

I owe so many thanks to my parents Ruijin Zhen and Jihui Li who have offered me unconditional love, patience, sacrifice, faith, and support. My most special thanks to my grandfather Yuting Zhen who has given me so much love and taught me so many life lessons. I will always miss you. It is hard to express my gratitude to my grandmother, uncle, aunt, and siblings for giving me so much support.

Finally, thanks to Zhiqian Qiao, my fiancée, my friend, my strength, and my joy. I could not imagine how my life would be without your consistent patience and tolerance in the past seven years.

This thesis is supported by the Department of Energy and the Shimizu Institute of Technology (Japan). Their support is gratefully acknowledged.

Abstract

Mobile robots are increasingly used for autonomous exploration and inspection because of the advancements in the technology of robot autonomy. In this thesis, we focus on the state estimation and mapping problem in the context of robot autonomous inspection. Although there exist well-established foundations, applying the technology in the real world remains challenging. Therefore, we are particularly interested in the robustness of algorithms when being applied in the real world. We believe addressing the challenges that arise from inspection-specific tasks will have more general impacts on other applications. Specifically, our work contains the following 4 contributions:

Firstly, we present a LiDAR-based pose tracking algorithm to achieve real-time robust state estimation. The proposed method is formulated using a probabilistic filtering framework named Error State Kalman Filter (ESKF) and can be used on low-end computing devices to achieve a robust estimation of agile motions. Despite the successful deployment in various environments, the method finds difficulties in scenarios with limited geometric features, such as long straight tunnels.

Secondly, we think about the failure cases of general LiDAR-based state estimation algorithms in situations where the geometric features are limited. A localizability model is presented and can be used to predict the confidence of localization based on the analysis of sensor readings. We then demonstrate successful localization in a real tunnel by fusing sensors with complimentary localizability.

Thirdly, we turn our attention to dense reconstruction for inspection. Inspired by the idea of complementary sensing, we combine LiDAR and camera data in a joint optimization framework to build globally consistent and locally dense 3D maps. Additionally, the proposed joint optimization framework is used to solve large-scale real-world problems by enhancing Structure from Motion (SfM) algorithms with LiDAR data providing structural constraints. Using the proposed methods, we demonstrate successful dense reconstruction of a real bridge tunnel.

Lastly, having noticed the structured nature of reconstructed dense models, we explore the possibility of using geometric primitives to build high-level compact maps. More specifically, we present a quadrics-based representation to unify multiple types of primitives in man-made environments, which leads to a concise and geometrically meaningful SLAM formulation. We show that using geometric primitives is not only a compact choice, the encoded high-level information also benefits the motion estimation by providing stronger constraints.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Autonomous mobile robots are starting to be used in many real-world tasks to replace humans for enhanced safety and efficiency. Among many applications, this work is motivated by using robots to inspect civil engineering facilities such as bridges and tunnels. As shown in Fig. 1.1, the conventional inspection can be a labor-intensive and time-consuming task since it often needs human workers to reach challenging places and extensively investigate the structure for potential defects. Therefore, our goal is to develop robot-based inspection technology to reduce the burden of human inspectors from the tedious process of data collection and analysis.

Similar to many other applications of autonomous robots, autonomous inspection requires the integration of multiple components (see Fig. 1.1). Firstly, a physical robot system needs to be developed to gather information from the environment. Considering the mobility and sensing capacity, we make use of small Unmanned Aerial Vehicles (UAVs) carrying various onboard sensors (e.g. LiDARs, cameras) as our major data collection system. Secondly, the autonomy stack that, in its simplest form, includes state estimation, planning and control, needs to be integrated with the hardware system to collect data autonomously. Finally, the collected data are processed to extract desired information according to the inspection targets such as building dense 3D models or detecting defects.

Our work focuses on the *state estimation and mapping* problem in the context of robot-based autonomous inspection. This problem, sometimes referred to as Simultaneous Localization and Mapping (SLAM), has been a popular research area in the past several decades. With the tremendous amount of effort been taken, the theory and framework of modern SLAM systems have been well established. However, as we apply the technologies to real-world inspection, many new challenges arise, bringing up several interesting but overlooked research questions that are potentially generalizable to more scenarios. Therefore, we are particularly motivated to study those questions and wish to gain insights that apply to general autonomous robots.

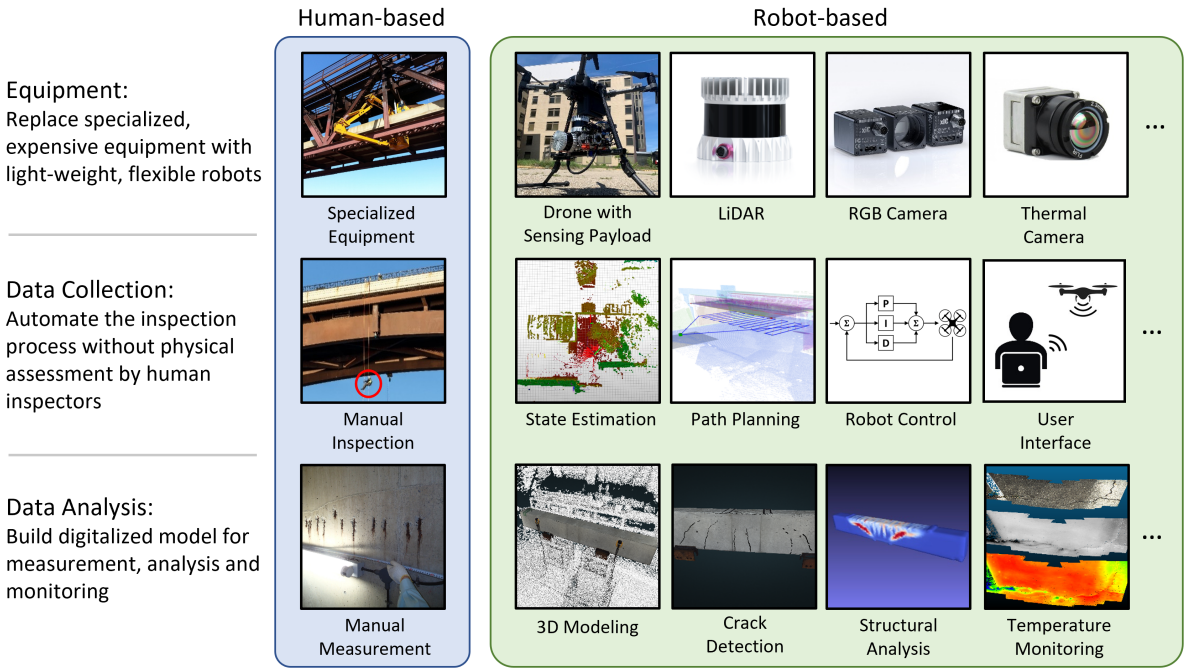


Figure 1.1: Overview of the differences between human-based and robot-based inspection[1].

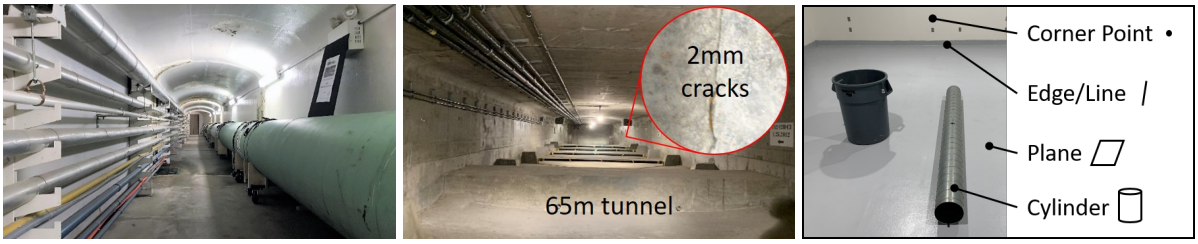
1.2 Challenges

As mentioned above, although the SLAM problem has well-established foundations, it remains challenging to achieve reliable performance in the real world. In this section, we discuss the challenges encountered deploying state-of-the-art SLAM methods in inspection tasks.

Challenge 1: Geometric degeneracy in structured environments.
Inspection tasks usually deal with structured environments that are considered simpler than unstructured natural scenarios. However, geometric degeneracy often exists in structured environments and could easily cause state-of-the-art localization algorithms to fail. Intuitively, geometric degeneracy happens when there is not enough information to constrain the estimation of location. As shown in Fig. 1.2a, a typical degenerated scenario for LiDARs is a long straight tunnel. When a robot navigates through, the measurements are identical everywhere along the tunnel, making the location under-determined. Therefore, our first challenge is to deal with environmental degeneracy such that the robot state can be estimated robustly.

Challenge 2: Large-scale meets high-density.
Inspection tasks also usually deal with large-scale facilities. For example, in bridge inspection, even a small-sized bridge could be over 50 meters. Therefore, long-range LiDAR is needed for large-scale mapping. The real challenge, however, is to build highly detailed dense models for structures at this scale. Highly detailed models may need a resolution of millimeters such that

[1]Specialized equipment and manual inspection photo courtesy: senseFly and MnDOT. Manual measurement photo courtesy: Shimizu Corporation.



(a) Degenerated tunnel (b) Scale and density (c) Structured environment

Figure 1.2: Overview of SLAM challenges for real world inspection.

tiny defects (e.g., cracks on the concrete wall shown in Fig. 1.2b) can be captured. Unfortunately, available robot mapping technologies such as Structure from Motion (SfM) or LiDAR SLAM still have difficulty in building both globally consistent and locally dense 3D maps.

Challenge 3: Redundancy of structured maps
Redundancy happens when the parameterization is overcomplicated for the underlying geometry. A typical way to simplify the map is to introduce parameterized shape priors (see the geometric primitives in Fig. 1.2c). However, choosing the suitable shape prior can be difficult: Firstly, multiple types of primitives are needed to represent different shapes, causing overhead for formulation and implementation. Secondly, primitives can be symmetric and degenerated, causing ambiguities. For example, the observation of a circular cylinder is degenerated in translation and rotation around its central axis, which should be characterized by the resulting representations. For these reasons, we identify shape representation for high-level maps as the third challenge.

1.3 Insights and Thesis Statement

To address the aforementioned challenges, we offer three key insights, the combination of which eventually leads to the central hypothesis of this thesis.

Insight 1: Ensuring robustness to degeneracy by modeling sensor localizability

Our first insight is that degeneracy is an attribute of the environment and, therefore, can be predicted by analyzing the geometry. Our thought roots in the intuition that the localization failure in long-straight tunnels is due to the lack of measurements constraining the robot's position. Then the question of predicting degeneracy becomes: how do sensor measurements contribute to the determination of the unknowns, and which part of unknowns is better constrained than others? To answer these questions, we consider the observation process as a function that outputs measurements based on robot location. Then the degeneracy of the function can be recognized by the diminishing gradient, indicating a weak correlation between inputs and outputs.

Insight 2: Leveraging the sensing capacity of LiDAR and cameras for dense mapping

Our second insight is that dense mapping should leverage the sensing capacities of multi-sensors for global consistency and local density. To guarantee the local density of built maps, we make use of Structure from Motion followed by densification techniques such that the reconstructed

3D model is textured and as dense as the image pixels. On the other hand, the LiDAR data are registered to provide global constraints and ensure overall consistency. Since both subproblems can be formulated as an optimization problem to solve for the optimal unknowns, the costs can be combined to consider all types of observations. Finally, by jointly minimizing the cost, we should expect to obtain improved map consistency and density than SfM or laser mapping algorithms.

Insight 3: Map abstraction using unified geometric primitives

Our third insight partially results from the observation that geometric primitives used for SLAM share a common algebraic representation called *quadrics*, which opens the possibility of a unified parameterization for many types of primitives. However, the algebraic nature makes it vulnerable to sensor noises and difficult to interpret. A more robust strategy is to formulate the geometric observation model of such primitives. Therefore, we will investigate the algebraic structure of quadrics to recover the implicitly encoded geometries, which is expected to render a robust formulation of the symmetric and degenerated quadrics.

With the discussion of the above insights to address the aforementioned challenges, we can now summarize the central statement of this thesis as:

> ***Degeneracy-prediction**, **fusion** and **abstraction** of sensing data are three key ingredients to enable robust, accurate and informative robot-based inspection.*

By predicting the degeneracy of sensing data, one can identify areas in the environment that are not safe for the robot to navigate through. By fusing measurements from multiple sensors, one can leverage the capabilities of each individual sensor to obtain more accurate estimations. Finally, by extracting high-level shape information from raw sensing data, we expect to obtain more compact and informative maps.

1.4 Contributions

Led by the key insights discussed above, the main contribution of this thesis is a series of works that use those ideas to solve the state estimation and mapping problem for inspection. In more detail, we offer the following contributions (structured as in Fig. 1.3):

- In **Chapter 2**, we revisit the fundamental concepts related to this thesis. We will also give a brief review of related works and compare the difference of our contribution to the existing literature.
- In **Chapter 3**, we first present a real-time pose tracking algorithm that fuses LiDAR and IMU measurements. The formulation of Error State Kalman Filter (ESKF) is provided in detail and it is shown to be able to estimate agile robot motion reliably through extensive tests within various environments. Then the proposed pose tracking algorithm is extended to a full-SLAM pipeline by introducing a mapping module to update the prior map. We show that the extended pipeline can build consistent maps using rotating or stationary 2D/3D lasers. The developed localization and SLAM modules serve as our baseline for

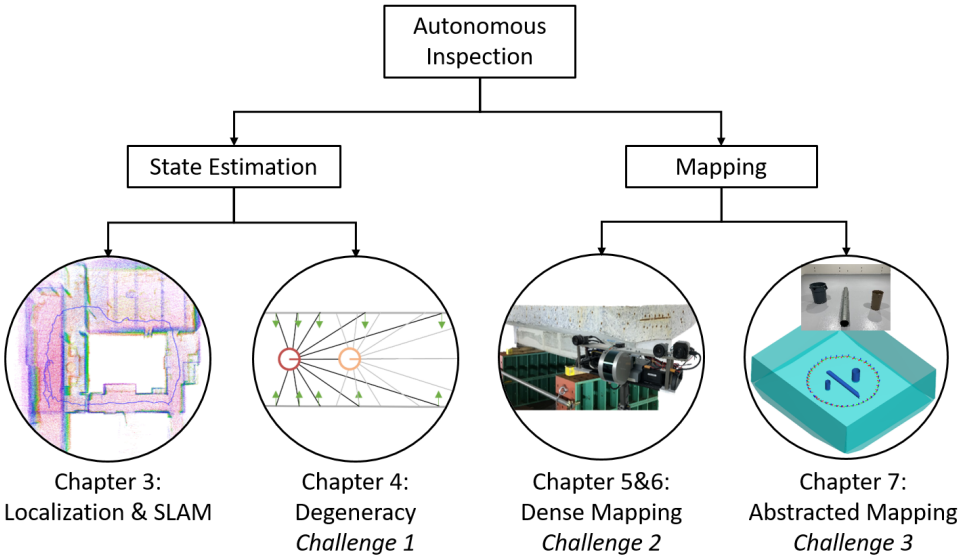


Figure 1.3: Overview of the thesis structure.

state estimation in inspection tasks.

- In **Chapter 4**, we formally introduce the idea of localizability in a more general estimation problem setting. We start from a basic LiDAR measurement model to formulate the constraints on robot poses and then apply the idea of perturbation to evaluate the *strength* of constraints along each dimension of the robot pose. In addition to the LiDAR model, the localizability model is easily generalizable to the Ultra-Wideband (UWB) radio, which can be treated as a complementary sensing module to the LiDAR. In the experiments, we reuse the developed ESKF framework to fuse LiDAR, UWB and IMU measurements and show more robust localization performance in a degenerated tunnel.
- In **Chapter 5**, we shift our focus from state estimation to mapping and meanwhile from online sequential computation to offline batch processing. Our ultimate goal for mapping is to build dense, textured, and accurate 3D maps of large-scale structures. To achieve this goal, diverse sensing modalities are necessary and we propose to fuse LiDAR and camera data in a joint optimization framework. Additionally, in the dense reconstruction phase, LiDAR data are used to refine the depth estimation by providing more information around low-texture areas. In the experiments, we present the reconstruction results of different structures and showed that our algorithm can achieve a similar level of precision to a high-end survey scanner.
- In **Chapter 6**, we consider making the joint optimization pipeline more robust in challenging conditions such as visual and structural ambiguities, large-scale, etc. We propose the LiDAR-enhanced SfM pipeline which introduces LiDAR data to provide additional structural constraints. It is shown that the proposed method can perform robust dense reconstruction of large-scale environments such as buildings and tunnels. Additionally, we report the comparison results of our pipeline to existing reconstruction methods in the literature.

- In **Chapter 7**, we rethink the mapping problem and try to make SLAM maps more lightweight. The maps built by methods discussed in Chapter 5 and 6 are dense and accurate but are often subject to redundancy, especially in man-made environments. Using geometric primitives is an effective way to simplify the map while still accurately capture the overall layout. In this work, we propose a novel unified representation based on *quadrics* such that the formulation of shape priors in SLAM becomes more concise. Particularly, we discuss the geometric properties such as degeneracy and symmetry of quadrics and the effects on the formulation of graph factors. In experiments, we show that the proposed representation allows for a robust and efficient back-end formulation that only uses a few geometric primitives.

With the contributions presented above, we have covered the major requests for state estimation and mapping in the context of autonomous inspection. However, it is worth mentioning again that our hope is to make the approaches and algorithms developed in this thesis not only applicable to specific tasks of inspection, but also useful for general applications of autonomous robots.

1.5 Publications

The following is a list of publications that this thesis is based on:

- Weikun Zhen, et al. "**Robust localization and localizability estimation with a rotating laser scanner**." International Conference on Robotics and Automation (ICRA). IEEE, 2017.
- Weikun Zhen, et al. "**A unified 3D mapping framework using a 3D or 2D lidar**." International Symposium on Experimental Robotics. Springer, Cham, 2018.
- Weikun Zhen, et al. "**Estimating the localizability in tunnel-like environments using LiDAR and UWB**." International Conference on Robotics and Automation (ICRA). IEEE, 2019.
- Weikun Zhen, et al. "**A joint optimization approach of LiDAR-camera fusion for accurate dense 3D reconstructions**." Robotics and Automation Letters. IEEE, 2019.
- Weikun Zhen, et al. "**LiDAR-enhanced structure-from-motion**." International Conference on Robotics and Automation (ICRA). IEEE, 2020.
- Weikun Zhen, et al. "**Unified representation of geometric primitives for graph-sLAM optimization using decomposed quadrics**." *submitted to* International Conference on Robotics and Automation (ICRA). IEEE, 2022.

Videos and open-source code associated with each chapter are listed in Table 1.1.

Table 1.1: Video and code of each chapter

Chapter 3:	www.youtube.com/watch?v=eA6dt_cGI5M www.youtube.com/watch?v=bs-KXuWLFec www.github.com/castacks/lidar_eskf
Chapter 4:	www.youtube.com/watch?v=ZK8wA3pyPyE
Chapter 5:	www.youtube.com/watch?v=dJaaF8POB64
Chapter 6:	www.youtube.com/watch?v=GUcKZ2PLPRQ
Chapter 7:	www.youtube.com/watch?v=dTuwAkVQGnQ

Chapter 2

Background

In this chapter, we introduce the background of inspection as well as the preliminary techniques that will establish the infrastructure for the remaining chapters. Specifically, we will first revisit the robot-based inspection problem in Section 2.1, depicting the typical scenario of how the robot would be used for inspection. Then we discuss the mathematical formulations of several SLAM-related problems in Section 2.2. Finally, in Section 2.3, we briefly review the literature related to our work.

2.1 Robot-based Inspection

What do the inspectors care about? The goal of inspection in the field of civil engineering is to access the structural health condition of the facilities based on the presence of defects. Depending on the materials, several different types of defects are of interest:

- *Rusts*: Rusts are critical features for metal constructions. For example, rusted bridges can be fragile under shape changes such as bending and twist.
- *Cracks*: The length and width of cracks are the metrics used to justify the condition of concrete since it could cause severe degradation of strength.
- *Spalling*: Similar to cracks, spalling concrete could also cause the loss of material strength since it is regarded as a sign of internal corrosion.

Information about those defects, including location, size, etc., needs to be documented in accurate details. And through periodic inspection, data from multiple sessions are accumulated to analyze the changes of defects, helping with the evaluation of the health condition.

How are robots used for inspection? The deployment of autonomous robots for inspection can be broken down into three phases as shown in Fig 2.1:

- *Data Collection*: At the beginning of data collection, the region-of-interest (ROI) is set by humans as the high-level target. Then the robot navigates autonomously to scan the given structure using onboard sensors (e.g. cameras) until the ROI is fully covered. The

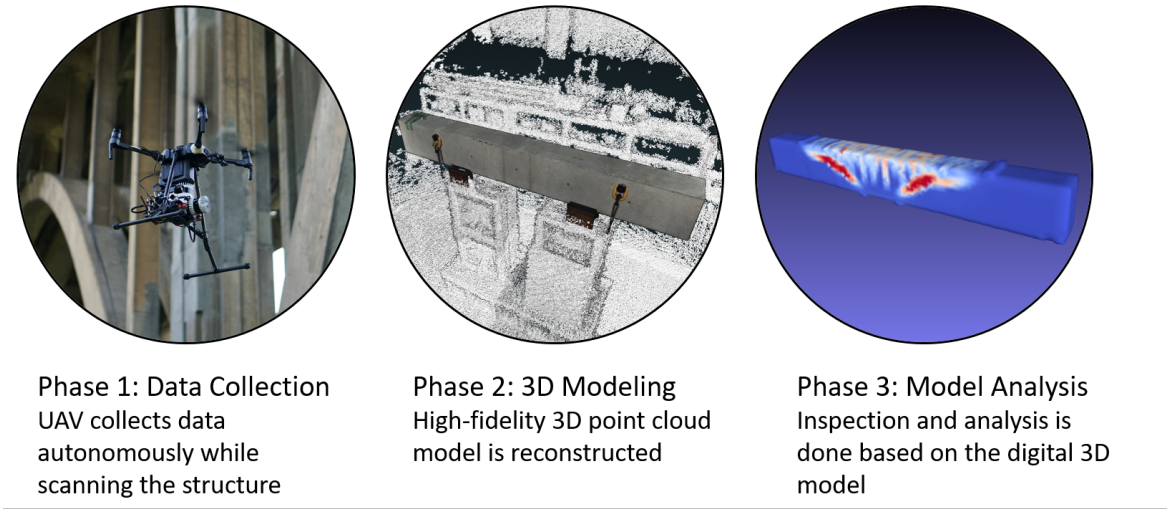


Figure 2.1: Overview of robot-based inspection.

autonomy requires real-time self-localization, a safe path that guarantees coverage, and a controller such that the robot follows the planned trajectory.

- *High-fidelity 3D Modeling*: Based on the collected data, reconstruction algorithms are used to build an accurate digital 3D model, which can be directly used for visual inspection and downstream analysis-related operations.
- *Model Analysis*: Using the digital 3D model as a reference, operations such as the measurement of defects and the analysis of structural strength can be carried out conveniently.

In this thesis, we mainly focus on the first two phases where phase 1 involves the technology of autonomous navigation while phase 2 focuses on 3D dense reconstruction.

2.2 State Estimation and Mapping

2.2.1 Notations

Before diving into the technical details, we first clarify the notation conventions. For this thesis, bold lower-case letters (e.g. $\mathbf{x}$) are used to represent vectors and bold upper-case letters (e.g. $\mathbf{R}$) are for matrices. Scalars are denoted as light font letters such as x and N. Besides, we use a *set* to define a collection of a certain type of elements and use upper-case calligraphic letters as the notation. For instance, $\mathcal{P} = \{\mathbf{p} \in \mathbb{R}^3\}$ defines a set of 3D points.

2.2.2 Bayesian Filters

Bayesian filters are a collection of algorithms used to estimate unknown variables based on a series of measurements [141]. A Bayesian filter infers the belief of the current system state $\mathbf{x}_t$ defined by the posterior:

$$bel(\mathbf{x}_t) = P(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}) \tag{2.1}$$

where $\mathbf{u}_{1:t}$ and $\mathbf{z}_{1:t}$ are a sequence of control inputs and sensor measurements respectively. $bel(\mathbf{x}_t)$ can be understood as a function that returns the probability (in continuous state space) or density (in discrete state space) for all possible $\mathbf{x}_t$. The estimation of $bel(\mathbf{x}_t)$ is recursive and can be summarized in the following two steps:

$$\begin{aligned} \overline{bel}(\mathbf{x}_t) &= \int P(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1}) bel(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1} \\ bel(\mathbf{x}_t) &= \eta \; P(\mathbf{z}_t|\mathbf{x}_t) \overline{bel}(\mathbf{x}_t) \end{aligned} \tag{2.2}$$

where the first step *predicts* the robot's current state based on the system dymamics model $P(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1})$, and the second step *corrects* the prediction by incorporating the sensor measurement model $P(\mathbf{z}_t|\mathbf{x}_t)$. Note that η is a normalization constant such that $bel(\mathbf{x}_t)$ sums to 1 over all possible $\mathbf{x}_t$. The description of Bayesian filters here only touches the surface of this complete framework. Readers can find more comprehensive explanation from [141].

System Dynamics Model: The system dynamics model defines the principles of how the state changes over time. In the case of a mobile robot, system dynamics typically models the robot motion including position, orientation and velocities. In discrete time terms, the system dynamics is represented by a transition function:

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) + \mathbf{w} \tag{2.3}$$

where $\mathbf{w}$ is the system transition uncertainty that is often assumed to be Gaussian. The transition function g can be linear or non-linear. For linear systems, the most seminal Bayesian filtering algorithm is the Kalman Filter (KF) [78] invented by Kalman in the 1960s. For non-linear systems, one popular choice is linearizing the system with its first-order Tylor expansion, also known as the Extended Kalman Filter (EKF). Other choices for non-linear systems include the Unscented Kalman Filter (UKF) [74] and the Particle Filter (PF) [122]. In the following discussions, we elaborate on the EKF since it is closely related to Chapter 3.

State Uncertainty: In (2.3), the Gaussian random variable $\mathbf{w}$ is used to model the state transition uncertainty. The accumulation of uncertainty makes our estimation of $\mathbf{x}$ less confident over time. More formally, the state uncertainty is a covariance matrix $\mathbf{\Sigma}$ measuring the covariances between each component of the state vector $\mathbf{x}$. For instance, in a 2D world, $\mathbf{x} = [x, y, \theta]$ is defined by the 2D location coordinate (x, y) and the heading angle θ. Then $\mathbf{\Sigma} \in \mathbb{R}^{3\times3}$ has the form of

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} \end{bmatrix} \tag{2.4}$$

It is hard to overstate the importance of uncertainties for probabilistic filtering algorithms since it governs how information is passed through the system. One could understand the state uncertainty as the inversed weighting factor when fusing data from multiple sources. For example, in the case of a robot carrying multiple ranging sensors with different noise profiles, the uncertainty matrices determine how their measurements are weighted to give the best estimate.

Propagation: This is the first step of a standard EKF. In the propagation step, a prediction of the current system state $\bar{\mathbf{x}}_t$ is computed as $\bar{\mathbf{x}}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ based on (2.3). More importantly, the state uncertainty is also predicted as $\bar{\mathbf{\Sigma}}_t$:

$$\bar{\mathbf{\Sigma}}_t = \mathbf{G}\mathbf{\Sigma}_t\mathbf{G}^T + \mathbf{\Sigma}_\mathbf{w} \tag{2.5}$$

where $\mathbf{G} = \frac{\partial g}{\partial \mathbf{x}}|_{\mathbf{x}_{t-1}}$ is the Jacobian of system dynamics evaluated at $\mathbf{x}_{t-1}$. $\mathbf{\Sigma}_\mathbf{w}$ is the covariance of random noise $\mathbf{w}$. The idea behind (2.5) is to predict the state uncertainty based on a linearized approximation of g. Although the model is less accurate after linearization, the propagation becomes quite simple since only linear operations (e.g. matrix multiplications) are involved.

In (2.5), it can be seen that the predicted uncertainty keeps increasing over time, meaning the estimation becomes less confident. Eventually, the uncertainty increases to a state that the prediction hardly contains any useful information. Fortunately, new sensor data can be used to correct the errors by providing extra constraints to the system state.

Sensor Measurement Model: The measurement model describes the relationship between sensor readings $\mathbf{z}$ (e.g. laser range measurements and visual features) and the system state $\mathbf{x}$. Again, the relationship is often non-linear and can be described as:

$$\mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{v} \tag{2.6}$$

where h is the non-linear measurement function and $\mathbf{v}$ is the Gaussian random sensing noise. One could understand function h as a way to predict the sensor measurements based on the prediction of the system state $\bar{\mathbf{x}}_t$. Later on, the difference between predicted and real measurements will provide clues to correct the state estimate.

Correction: How to use the measurements to correct or update the prediction that is subject to drifts? The intuition is to compute a weighted average between the prediction $\bar{\mathbf{x}}_t$ and the *implict* state encoded in sensor measurements. The word *implict* is used since the sensor may not directly measure the variables in $\mathbf{x}$, but are related to them through the measurement model h. Mathematically, the correction is done via the following steps:

- Compute the difference between real measurements and predicted measurements:
  $\mathbf{y}_t = \mathbf{z}_t - h(\bar{\mathbf{x}}_t)$
- Prediction or observation? Compute the weighting factor named Kalman gain:
  $\mathbf{K} = \bar{\mathbf{\Sigma}}_t\mathbf{H}^T(\mathbf{\Sigma}_\mathbf{w} + \mathbf{H}\bar{\mathbf{\Sigma}}_t\mathbf{H}^T)^{-1}$
- Compute a new state based on weighted average:
  $\mathbf{x}_t = \bar{\mathbf{x}}_t + \mathbf{K}\mathbf{y}_t$
- Compute the new state uncertainty:
  $\mathbf{\Sigma}_t = (\mathbf{I} - \mathbf{K}\mathbf{H})\bar{\mathbf{\Sigma}}_t$

In here, $\mathbf{H} = \frac{\partial h}{\partial \mathbf{x}}|_{\bar{\mathbf{x}}_t}$ is the Jacobian of h evaluated at $\bar{\mathbf{x}}_t$. The completion of the above steps yields a new set of estimation $(\mathbf{x}_t, \mathbf{\Sigma}_t)$ which will participate in the next propagation step.

2.2.3 Factor Graph

Factor graph in SLAM is a graphical model used to unify measured information from different sensing modalities. Thanks to the advancements in graph-based smoothing and mapping [76][77][35], factor-graph has become the backbone of modern SLAM systems [37][81][77] [65][159][130]. In this section, we briefly revisit the concepts of factor graph and graph optimization.

Nodes: The nodes $n \in \mathcal{N}$ in a factor graph represent unknown variables to be estimated. For example, a robot's pose or the 3D location of a landmark could be defined as nodes of different types.

Edges: The edges $e \in \mathcal{E}$ in a factor graph encode the relationship between two nodes. From the perspective of constraints, the edge e_{ij} can be translated into a constraint between the states $\mathbf{x}_i$ and $\mathbf{x}_j$. For instance, a measurement of the relative motion between two consecutive poses $\mathbf{z}_{ij}$ makes a transform constraint. The constraint error function $\mathbf{e}_{ij}$ can be defined as the residual of observed measurement $\mathbf{z}_{ij}$ and predicted measurement $h(\mathbf{x}_i, \mathbf{x}_j)$:

$$\mathbf{e}_{ij} = \mathbf{z}_{ij} - h(\mathbf{x}_i, \mathbf{x}_j) \tag{2.7}$$

The metric of error function follows a simple idea: the predicted measurements should be similar to the real measurements under true state estimates.

Factors: In the context of graph-SLAM, factors can be treated as constraints imposed on a single node (unary factor) or between two nodes (binary factor). In this thesis, we treat the word *factor* interchangeable with *constraints*.

Uncertainty: In factor graphs, uncertainties are defined for the factors. Intuitively, it indicates the *strength* of a constraint. Similar to the uncertainty representation in EKF, the uncertainty in factor graphs is also represented as a covariance matrix in the space of observations $\mathbf{z}$.

Graph Optimization: The high-level goal of graph optimization is to estimate the system states that match the observations. More specifically, the errors associated with each factor should be minimized to find the optimal state estimate $\mathbf{x}^*$. Therefore, a cost function f is defined as a sum of constraint residuals:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f \quad \text{where } f = \sum_{ij \in \mathcal{E}} \mathbf{e}_{ij}^T \mathbf{\Sigma}_{ij} \mathbf{e}_{ij} \tag{2.8}$$

Here $\mathbf{\Sigma}_{ij}$ is used to define the Mahalanobis distance which has the effects of weighting each dimension based on the confidence. (2.8) minimizes the squared error and is non-linear with respect to $\mathbf{x}$. Therefore, it is often called Non-Linear Least Square (NLLS) problem.

Gauss-Newton: To solve the NLLS problem defined in (2.8), iterative descent methods can be used. Among many descent strategies, the Gauss-Newton (GN) method is a popular choice

to solve optimization problems in robotics. The key idea of GN is to repeatedly approximate the cost function f with a quadratic equation F and perform exact minimization on F. The approximation can be done by linearizing $\mathbf{e}$ using its first-order Tylor expansion:

$$\mathbf{e}(\mathbf{x}_0 + \Delta\mathbf{x}) \approx \mathbf{e}(\mathbf{x}_0) + \mathbf{J}\Delta\mathbf{x} \tag{2.9}$$

where $\mathbf{J} = \frac{\partial \mathbf{e}}{\partial \mathbf{x}}|_{\mathbf{x}_0}$ is the error Jacobian evaluated at the initial guess $\mathbf{x}_0$ and $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_0$ is a small perturbation around $\mathbf{x}_0$. Then the linearized cost function has a desired quadratic form:

$$\begin{aligned} f(\mathbf{x}) &\approx F(\Delta\mathbf{x};\ \mathbf{x}_0) \\ &= \sum \mathbf{e}^T\mathbf{e} + 2\mathbf{e}^T\mathbf{J}\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{J}^T\mathbf{J}\Delta\mathbf{x} \end{aligned} \tag{2.10}$$

In here, we drop the subscripts to simplify the notations. To minimize F, the optimal $\Delta\mathbf{x}$ can be found analytically:

$$\Delta\mathbf{x} = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{e} \tag{2.11}$$

Here we abuse the notion of $\mathbf{J}$ and let it denote the assembled Jacobian from all observations. Then $\mathbf{x}$ is updated by $\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$. Finally, by repeating the approximation, minimization and update steps, the final optimal solution $\mathbf{x}^*$ can be found.

One limitation of the GN method is that it will have difficulty when $\mathbf{J}^T\mathbf{J}$ (the Hessian matrix) is ill-conditioned, which is an issue frequently encountered in practice. In this case, the GN method may not decrease at every iteration. Alternatively, the Levenberg-Marquart (LM) method [97] provides a simple modification of $\Delta\mathbf{x}$ to stabilize the descent procedure:

$$\Delta\mathbf{x} = -(\lambda\mathbf{I} + \mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{e} \tag{2.12}$$

where λ is a scalar factor that governs the tendency to the gradient descent (GD) update $-\mathbf{J}^T\mathbf{e}$. Since GD guarantees to give a descending direction, λ is increased such that $\Delta\mathbf{x}$ approaches closer to GD when GN becomes not stable.

Linearization on SO(3): The problem arises from computing Jacobian w.r.t the rotation, which belongs to the non-linear *special orthogonal group* noted as SO(3). There are multiple ways to parameterize the rotation, including orthogonal matrix, unit quaternion, angle-axis and Euler angles. In this work, we tend to use the rotation matrix more often since it directly applies to coordinate transform. To linearize a rotation matrix, the key ingredient is the small-angle approximation:

$$\mathbf{R} \approx \mathbf{I} + [\boldsymbol{\theta}]_\times \tag{2.13}$$

where $[\boldsymbol{\theta}]_\times$ is the skew symmetric matrix of the angle vector $\boldsymbol{\theta} = [\theta_x, \theta_y, \theta_z]$, and $\mathbf{I}$ is the identity matrix. The approximation is only valid when the angle vector $\boldsymbol{\theta}$ approaches zero. When the small angle condition is satisfied, (2.13) is now linear in terms of $\boldsymbol{\theta}$.

In practice, any rotation $\mathbf{R}$ can be regarded as the composition of an initial rotation $\mathbf{R}_0$ and a small perturbation $\Delta\mathbf{R}$: $\mathbf{R} = \mathbf{R}_0\Delta\mathbf{R}$. Once $\Delta\mathbf{R}$ is linearized, $\mathbf{R}$ is also linear in terms of $\boldsymbol{\theta}$. Hence the linearization completes.

2.2.4 Bundle Adjustment

Bundle adjustment [142] is a technique to simultaneously estimate the camera motion and 3D structures and has been widely used in visual SLAM and SfM algorithms. Given an initial guess of the camera poses $\{\mathbf{R}_i, \mathbf{t}_i\}$ and 3D points $\{\mathbf{P}_j\}$, the goal of bundle adjustment is to minimize the reprojection error of feature points. Specifically, the optimization problem is defined as:

$$\min_{\{\mathbf{R}_i, \mathbf{t}_i, \mathbf{P}_j\}} \sum_{i,j} ||\mathbf{p}_j - \pi(\mathbf{R}_i \mathbf{P}_j + \mathbf{t}_i;\ \mathbf{K})||^2 \tag{2.14}$$

where $\mathbf{p} \in \mathbb{R}^2$ is the pixel coordinate of an observed point. π is the function that projects 3D points to the image plane. $\mathbf{K}$ is the camera intrinsic matrix

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

that models the parameters of a pinhole camera, including the focal length (f_x, f_y), sensor skew s and optical center coordinates (c_x, c_y).

2.2.5 Point Cloud Registration

Point cloud registration is the core problem of motion estimation using depth or range sensors (e.g. LiDAR). By aligning two given point clouds, the relative motion between two frames can be recovered. Depending on the strategies of data association, point clouds can be registered using two approaches:

- *Direct registration* relies on 3D geometric features [59] to find point-to-point correspondences between point clouds. Once the correspondences are found, the relative transform can be readily computed using direct estimation techniques such as SVD [8]. Despite the convenience of computing a solution, direct registration methods may suffer from outliers hence are usually coupled with robust mechanisms such as RANSAC [44]. Additionally, the location precision of 3D features is usually lower than the 2D counterpart, which further deteriorates the registration accuracy.
- *Iterative registration*, or the Iterative Closest Point (ICP) algorithm [14], finds correspondences using the nearest neighbor search strategy and updates the matches as the estimation progressively becomes more accurate. This strategy requires a good initial guess to start the iteration. Otherwise, the algorithm could converge to a local minimum or even diverge. However, due to its simplicity, efficiency, and the fact that an initial guess is often easy to obtain, ICP and ICP-variants [123] are widely used for online registration.

Apart from the difference in correspondence search, direct and iterative registration share a common basic point set registration formulation:

$$\min_{\mathbf{R},\mathbf{t}} \sum_i ||\mathbf{p}'_i - (\mathbf{R}\mathbf{p}_i + \mathbf{t})||^2 \tag{2.16}$$

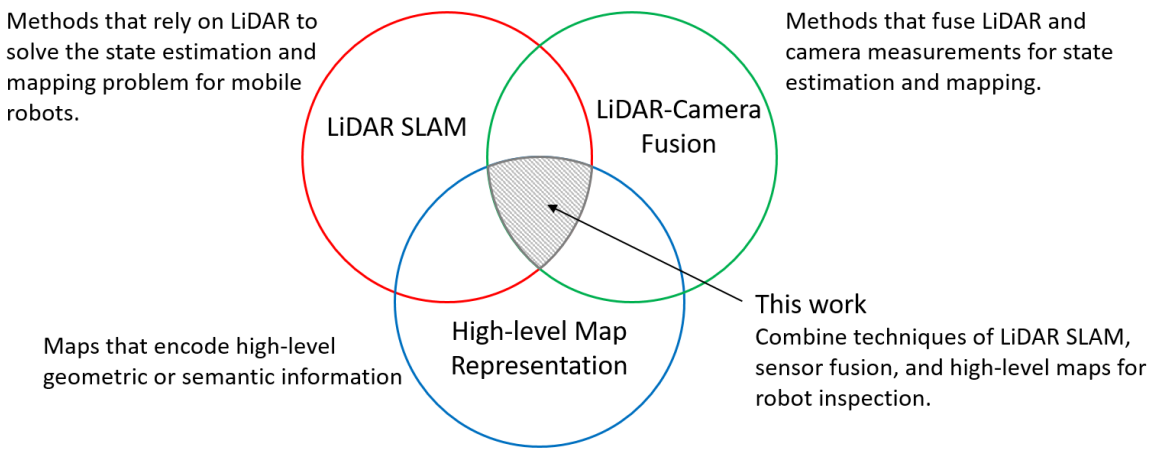


Figure 2.2: Overview of the related work and the intersection to which this thesis belongs.

where $\mathbf{p}_i$ and $\mathbf{p}'_i$ are matched 3D points in two different frames. In (2.16), point-to-point distances are minimized to align given point clouds. Although this error metric is straightforward, the underlying assumption of point-to-point match hardly holds in practice. This is because a point cloud is only a sparse sample of the underlying surface. Thus there is no guarantee that the same point is sampled in both point clouds. For this reason, alternative distance metrics, such as point-to-line and point-to-plane, can be used. Here we provide two examples of point-to-line and point-to-plane metrics:

$$\begin{aligned} \text{point } \mathbf{p} \text{ to line } (\mathbf{q}, \mathbf{w}): &\quad (\mathbf{I} - \mathbf{w}\mathbf{w}^T)\,(\mathbf{q} - (\mathbf{R}\mathbf{p} + \mathbf{t})) \\ \text{point } \mathbf{p} \text{ to plane } (\mathbf{q}, \mathbf{n}): &\quad \mathbf{n}^T\,(\mathbf{q} - (\mathbf{R}\mathbf{p} + \mathbf{t})) \end{aligned} \tag{2.17}$$

where a line in 3D space is defined by a point $\mathbf{q}$ and a direction $\mathbf{w}$, while a plane is defined by a point $\mathbf{q}$ and its normal vector $\mathbf{n}$. Note that in (2.17), the definitions are only two examples of distance metrics. Depending on the parameterization of lines and planes, the distance formulation could be different.

2.3 Related Work

We find our work roughly belongs to 3 themes of state estimation and mapping as shown in Fig. 2.2. LiDAR-based SLAM is closely related to Chapter 3 and 4 where we use LiDAR as our primary sensing module to solve the robot localization problem. Then sensor fusion becomes the major theme of Chapter 5 and 6 where LiDAR is combined with cameras for dense mapping. Finally, high-level map representations are relevant to Chapter 7 where geometric primitives are used as abstraction of dense maps. Our goal for this section is to briefly review the related work of each theme, depicting the big picture of where our work is positioned in the literature.

2.3.1 LiDAR-based SLAM

LiDAR SLAM relies on LiDAR to solve the online state estimation and mapping problem for mobile robots. Unlike cameras, LiDAR provides accurate range measurements and is robust to environmental conditions such as lighting and weather. Therefore, the LiDAR SLAM technology has gained large popularity in safety-critical areas such as self-driving.

A fundamental technique of LiDAR SLAM is scan matching, which is also known as the point cloud registration problem (see Section 2.2.5). The goal of scan matching is to find the relative motion between two consecutive scans using ICP or its variants. The standard ICP aligns full point clouds, which can be inefficient in practice. Therefore, geometric features such as edges and planes have been used for fast registration. For example, Zhang et al. [153] use edge and planar points to perform ICP, which significantly improved the speed and accuracy.

LiDAR is rarely used alone. This is because the sensor's egomotion could cause distortion in scanned point clouds. To accommodate this issue, LiDAR is usually combined with an IMU to compensate for the egomotion in a short time range. Depending on how LiDAR is combined with IMU, one can roughly categorize the available methods into loosely-coupled and tightly-coupled approaches.

In loosely-coupled approaches, IMU measurements are not directly involved in the optimization. For instance, Zhang et al. [153] integrates IMU to predict the sensor motion, which is used as an initial guess for the ICP. In this process, laser scan still dominates the optimization procedure. Another type of loosely-coupled method is using Bayesian filters for state estimation. As introduced in Section 2.2, Bayesian filters have two key steps: propagation and correction. IMU measurements are integrated in the propagation step to give state prediction, while laser scans are registered to find the relative transform used in the correction step to update the prediction. For examples, works in [22][91][63] implement EKF to fuse LiDAR and IMU measurements for robust state estimation.

On the other hand, for tightly-coupled approaches, IMU directly participates in the optimization process. This is often achieved by using the factor graph framework [35] where sensor measurements are encoded as factors. To be more precise, IMU measurements are pre-integrated [46] as IMU-factors providing a transform constraint between two frames. Meanwhile, laser scans are registered yielding another transform constraint. Then the factor graph can be optimized such that both types of constraints are satisfied. This scheme has become the primary choice in latest LiDAR SLAM systems [65][129][130][159].

Which choice is better: loosely-coupled or tightly-coupled methods? Loosely-coupled methods benefit from their simplicity and have dominated early works of LiDAR SLAM. Tightly-coupled methods typically offer better accuracy hence have become the focus of ongoing research. One argument of the loosely-coupled strategy is that it has a less computational cost, thus suitable for online applications. However, thanks to the advancements of incremental mapping techniques [76][77] and maturing software tools [35][2], the efficiency of tightly-coupled methods is getting improved constantly.

Our work in this thesis is related to the theme of LiDAR SLAM in two folds:

- We offer a loosely-coupled online localization algorithm that combines a rotating 2D LiDAR with IMU. Why is loosely-coupled strategy the choice when tightly-coupled methods provide higher accuracy? The key reason is that loosely-coupled methods offer a better trade-off between accuracy and the computational costs considering the limited computing resources (e.g., ODROID XU4[1]) on the robot.
- Although the methods discussed above build high quality maps as a result of accurate pose estimation, the map precision (typically at the level of several centimeters) are limited by LiDAR hence is not enough to capture small details (at the scale of several millimeters) for inspection. Our work, on the one hand, relies on similar techniques such as scan registration to process LiDAR data; on the other hand, it focuses on fusing LiDAR and camera for locally dense and globally consistent reconstruction to suit the needs of inspection.

2.3.2 LiDAR-camera Fusion

In this section, we discuss the literature of LiDAR-camera fusion for dense mapping. Before delving into sensor fusion techniques, we first take a look at dense mapping solutions. The goal of dense mapping is to build high-resolution 3D models to encode the rich details of the environment. Then how dense should a dense model be? To answer this question, one has to relate to the specific context. In the field of laser mapping [153], the built maps are often regarded as dense maps in the sense that raw laser points are denser than sparse geometric features. In the area of visual mapping, the term dense tends to indicate per-pixel density, which is usually of higher resolution than LiDAR maps (assuming moderate image resolution and close-range). Therefore, for the purpose of inspection that concentrates on detailed models, visual mapping seems to be the preferred solution. Available methods in the literature can build dense maps in an online manner using normal or RGB-D cameras [105][42], or in an offline manner using batch dense reconstruction techniques [11] [127]. In the industrial world, many visual dense mapping solutions have emerged. For instance, mapping solutions provided by Pix4D[2] and Skydio[3] have been used for construction site monitoring and inspection. However, despite the high resolution, those methods are typically not as robust as LiDAR against environmental factors such as darkness and low texture, making it difficult to build maps in such challenging situations.

For the reasons discussed above, combining camera and LiDAR using sensor fusion techniques has attracted many researchers [34]. From a general point of view, LiDAR-camera fusion aims to compensate for the disadvantages of each sensor, closing the modality gap between structural and textural data. Here, we borrow concepts from robot perception to help characterize different strategies of fusing LiDAR and camera for SLAM. In the area of perception, sensor fusion techniques are categorized into early fusion (data-level), middle fusion (feature level), and late fusion (decision level), depending on the levels of data abstraction [83]. The classification rule seems to apply to fusion techniques for SLAM as well:

[1]ODROID: `https://wiki.odroid.com/odroid-xu4/odroid-xu4`
[2]Pix4D: `https://www.pix4d.com/`
[3]Skydio: `https://www.skydio.com/`

- In the early fusion strategy, LiDAR scanned point clouds are projected to align with the images. For instance, Maddern [92] uses projected LiDAR depth to improve the quality of stereo disparity maps.
- In the middle fusion strategy, LiDAR and camera interact at the level of features. One example is the work from Levinson et al. [86] that extracts and matches edge features from laser scans and images for automatic extrinsic calibration. Another example of middle fusion is to assign LiDAR depth to visual features as in [156] to improve the performance of visual odometry.
- For the late fusion strategy, both LiDAR and camera data are first processed separately to estimate the robot motion; then multiple estimations are fused in a common framework. For instance, the work from Shao et al. [131] uses a pose graph to join visual and LiDAR pose estimations.

The discussion of different fusion levels helps clarify multiple ways for fusing camera and LiDAR data. For the purpose of online application, most LiDAR and camera fusion techniques for SLAM choose to abstract the sensing data progressively to reduce the computation burden for downstream components, which essentially follows the late fusion strategy. Although highly efficient, this architecture will have difficulty in handling compound errors accumulated through the pipeline. In the case of a factor graph, errors due to imperfect feature matching are passed to the back-end as biased constraints, whose effects can be hard to correct. In this work, we trade off efficiency for improved accuracy by following the early fusion strategy. More specifically:

- We consider combining the bundle adjustment and point cloud registration problems based on raw sensing data (e.g. visual features and point clouds), which falls into the category of early and middle fusion. In this way, one can preserve the chance to reject mismatches in the raw data.
- Despite the early fusion strategy is used for batch optimization, we deploy late fusion techniques for initialization. For instance, relative motions are used to solve a pose graph to initialize the poses of the robot, providing a starting point for the final joint optimization.

2.3.3 High-level Maps for SLAM

Map representation determines how the map encodes the information of the environment. The encoded information could be purely geometric, resulting in a metric map, or semantic, leading to a semantic map. In this section, we offer a brief discussion of the available choices for map representation. Specifically, we start with low-level representations that are widespread in the literature, and then focus on the high-level geometric and semantic representations that are gaining more attention in the latest works.

Low-level representations have been successfully used in a large number of available SLAM systems. For example, point-based sparse representation, encoding discriminative features such as corners [61][12], has been the main stream solution for visual odometry and recognition. However, feature-based representations are considered too sparse for many purposes, such as

metric mapping and navigation. Therefore, dense models, such as point clouds [153], occupancy grids [22][68], and meshes [56], have been used. Those methods use simple non-parametric entities to approximate the geometry of the environment. A more complex model known as the Truncated Signed Distance Function (TSDF) [33] represents the surface of an object by storing the signed distance values in each voxel. Although visually appealing, TSDF is expensive to store, thus difficult to generalize to large environments.

Recently, object-based SLAM has drawn great attention as it allows for a more intelligent understanding of maps. Early works from Castle et al. [25] and Salas-Moreno et al. [124] integrate object models into an online SLAM framework that optimizes the robot and object poses jointly. Rosinol et al. [118] construct meshes with sematic labels, which is then extended to build hierarchical high-level maps with 3D dynamic scene graphs [119]. Ok et al. [108] and Yang et al. [151] use abstracted shapes to approximate the objects in the scene.

In addition to semantics, higher-level geometric shapes have also been introduced to SLAM. Compared to low-level maps, high-level shapes can be extremely compact while capturing the scene layout. For instance, line features have been used for both visual SLAM [115] and LiDAR odometry [153] to map the edges in an environment. Planes [75][158][69] are one of the most widely seen primitives in structured environments. More complex geometric primitives such as cuboids [151], ellipsoids [107], and superquadrics [144] are introduced as approximated shapes for objects. It should be noted that even though those high-order shapes are typically embedded with semantic labels, it is the underlying geometry that directly affects the formulation of SLAM.

Driven by the goal of compact representations for structured environments, this thesis will explore the usage of high-level geometric primitives for the purpose of SLAM. We now highlight the relationship of this work to available methods in the literature:

- Instead of using a fixed type of shape prior, we look for a general representation based on quadrics, which allows for parameterizing multiple geometric primitives uniformly. We believe a unified representation could greatly simplify the usage of primitives in SLAM.
- Instead of modeling objects in the environment, we focus on high-level representation of large structures such as walls and pillars. We particularly examine how the introduction of high-level shapes affects the sensing model in a SLAM framework.

Chapter 3

Robust Localization using LiDAR and IMU

This chapter takes the first step towards autonomous robot inspection: developing a physical system that can localize itself. We start with the discussion of the localization problem for UAVs in Section 3.1. In this discussion, we try to highlight the characteristics of localizing UAVs and explain how the sensing system and algorithms are designed to meet certain requirements. Then in Section 3.2, we review prior works that are closely related to this chapter. In Section 3.3, we introduce the proposed localization algorithm in detail. Section 3.4 describes how the proposed localization algorithm fits in an off-the-shelf SLAM pipeline such that the global map can be updated with newly perceived information. Experimental results and demonstrations are presented in section 3.5 to show how the proposed algorithm performs in various environments. Finally, in section 3.6, we discuss the conclusions and lessons learned from this work.

3.1 The Localization Problem

UAVs are selected as our major experimental platforms because of their compact form-factor and outstanding mobility. Although those advantages make UAVs suitable for exploration and inspection tasks, they also come with limitations. Firstly, the small form-factor often indicates limited payload capacity, restraining both computing power and flight time. Secondly, the robot motion can be more aggressive in the air, requiring special considerations about the design of sensing systems and algorithms.

Our main sensing module is a LiDAR which gives accurate ranging measurements and is robust to lighting conditions. Despite the ranging precision, the Field of View (FOV) of LiDAR sensors, however, is usually limited. 2D LiDARs take measurements within the scanning plane, while 3D LiDARs only have slightly wider vertical FOV. In this work, we consider LiDARs mounted on a continuously rotating motor so that the robot has a spherical FOV. Besides the FOV limits, the scan frequency of LiDAR sensors is usually not high enough to capture the fast motion of the robot. Therefore, IMU is often used to measure the system acceleration and angular velocity at a much higher sampling rate (typically 100Hz or even more). Our setup follows the same strategy to have both LiDAR and IMU installed as shown in Fig. 3.4.

Based on the sensing setup, now our goal is to estimate robot states upon the arrival of the latest measurements. More specifically, the problem can be stated as follows:

> Given a prior map and an initial guess of the robot pose, the robot localization problem is to localize the robot by sequentially processing the measurements from the LiDAR and the IMU.

The term *localization* is beyond the estimation of location only. It also involves the estimation of robot orientation and velocities. In fact, in this work, we also consider modeling the sensor biases to compensate for IMU drifts.

3.2 Related Work

The work in this chapter is closely related to filtering-based state estimation. Popular filtering methods rely on the framework of Bayesian filters to estimate robot state from given sensor readings. Early works such as Monte Carlo Localization (MCL) [47] successfully applies particle filter to solve robot localization problem. The sampling nature of this method makes it suitable for non-Gaussian uncertainties. However, it has difficulties in higher dimension spaces due to exponentially increasing computational cost. Differently, EKF can efficiently handle non-linear systems with Gaussian noise assumption and has been used for laser mapping in [91][38][63]. It is also reported that EKF-like models do not have significant differences from sample-based models in terms of performance. Instead of linearizing system dynamics as in EKF, Error State Kalman Filter (ESKF) [120] directly models the system state as near-zero errors such that the transition function becomes linear in the new error space. This technique has been used for pose estimation and mapping [117]. Our work is closely related to [22] where a Gaussian Particle Filter (GPF) is used to convert raw LiDAR measurements into robot pose estimates. But differently, we represent the system dynamics and measurement model in the error state space, and also include IMU bias terms to the state vector to compensate for inaccurate predictions from inertial data.

3.3 Localization Formulation

To recap the problem: given a pre-built map and the initial pose of the robot, its current pose, velocity, and IMU biases are estimated immediately when a new set of range data is available. The localization is achieved by combining an Error State Kalman Filter (ESKF) with a Gaussian Particle Filter (GPF) as shown in Figure 3.1. From a high-level perspective, the ESKF is responsible to maintain an estimate of system states while the GPF is used to recover the *implicit* poses from laser scans.

The ESKF shares the same propagation and update steps as the KF except that the system state is represented in the error space. The error state representation, compared to a nominal state

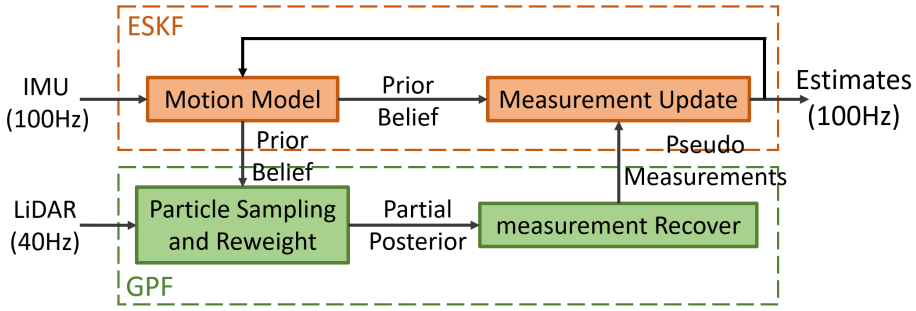


Figure 3.1: The localization algorithm overview.

representation, has several benefits [93]. First of all, error states are always close to zero. Therefore the small-angle approximation in (2.13) can be applied to compute the derivatives of the matrix exponential map. Secondly, in the error state, the rotation error can be represented as a 3D vector, which is intuitive to interpret the physical meaning. Finally, the rotation error is far from a singular configuration compared to the Euler angle representation.

3.3.1 The Motion Model

Error States: The state vector of the system contains the following components:

$$\begin{aligned} \text{velocity in global frame: } & \mathbf{v} \in \mathbb{R}^3 \\ \text{position in global frame: } & \mathbf{p} \in \mathbb{R}^3 \\ \text{rotation matrix: } & \mathbf{R} \in SO(3) \\ \text{accelerometer bias: } & \mathbf{a}_b \in \mathbb{R}^3 \\ \text{gyroscope bias } & \boldsymbol{\omega}_b \in \mathbb{R}^3 \end{aligned}$$

Here the sensor bias is different from sensor noise in that it drifts slowly around zeros instead of randomly sampled. The sensor readings are corrupted with sensor bias and sensor noise at the same time.

We then define the true state $\mathbf{x}$ as the hidden variables to be estimated, the nominal state $\hat{\mathbf{x}}$ as the current best estimation:

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} \mathbf{v} & \mathbf{p} & \mathbf{R} & \mathbf{a}_b & \boldsymbol{\omega}_b \end{bmatrix}^T \\ \hat{\mathbf{x}} &= \begin{bmatrix} \hat{\mathbf{v}} & \hat{\mathbf{p}} & \hat{\mathbf{R}} & \hat{\mathbf{a}}_b & \hat{\boldsymbol{\omega}}_b \end{bmatrix}^T \end{aligned} \tag{3.1}$$

The error state $\delta\mathbf{x}$:

$$\delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{v} & \delta\mathbf{p} & \delta\boldsymbol{\theta} & \delta\mathbf{a}_b & \delta\boldsymbol{\omega}_b \end{bmatrix}^T \tag{3.2}$$

is defined as the difference between the true value and its estimation:

$$\mathbf{x} = \hat{\mathbf{x}} \oplus \delta\mathbf{x} \tag{3.3}$$

Here $\oplus$ indicates a generic composition where rotations are composed on the SO(3) manifold. Although the system dynamics are now represented in the error states, one still needs the nominal

states to keep track of the actual physical states of the system for the purpose of downstream applications (e.g. control).

Error Dynamics: To derive the error state dynamics, we start from formulating the norminal state dynamics which is given by:

$$\dot{\hat{\mathbf{x}}} = \begin{bmatrix} \dot{\hat{\mathbf{v}}} \\ \dot{\hat{\mathbf{p}}} \\ \dot{\hat{\mathbf{R}}} \\ \dot{\hat{\mathbf{a}}}_b \\ \dot{\hat{\boldsymbol{\omega}}}_b \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{R}}(\mathbf{a}_m - \hat{\mathbf{a}}_b) + \mathbf{g} \\ \hat{\mathbf{v}} \\ \hat{\mathbf{R}}[\boldsymbol{\omega}_m - \hat{\boldsymbol{\omega}}_b]_\times \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{3.4}$$

where $\mathbf{a}_m, \boldsymbol{\omega}_m$ are the acceleration and angular velocity measurements. Then the error dynamics equation can be obtained by replacing $\hat{\mathbf{x}}$ with $\hat{\mathbf{x}} \oplus \delta\mathbf{x}$. Particularly, the rotation $\mathbf{R}$ is replaced with $\mathbf{R}(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times)$:

$$\delta\dot{\mathbf{x}} = \begin{bmatrix} \delta\dot{\mathbf{v}} \\ \delta\dot{\mathbf{p}} \\ \delta\dot{\boldsymbol{\theta}} \\ \delta\dot{\mathbf{a}}_b \\ \delta\dot{\boldsymbol{\omega}}_b \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{R}}[\mathbf{a}_m - \hat{\mathbf{a}}_b]_\times \delta\boldsymbol{\theta} - \hat{\mathbf{R}}\delta\mathbf{a}_b - \hat{\mathbf{R}}\mathbf{a}_n \\ \delta\mathbf{v} \\ -[\boldsymbol{\omega}_m - \hat{\boldsymbol{\omega}}_b]_\times \delta\boldsymbol{\theta} - \delta\boldsymbol{\omega}_b - \boldsymbol{\omega}_n \\ \mathbf{a}_w \\ \boldsymbol{\omega}_w \end{bmatrix} \tag{3.5}$$

where $\mathbf{a}_n, \boldsymbol{\omega}_n$ denote the accelerometer and gyroscope Gaussian noise, and $\mathbf{a}_w, \boldsymbol{\omega}_w$ are the Gaussian random walk noise of biases.

Note that (3.4) and (3.5) are dynamics represented in the continuous-time form. In the continuous form, the first-order derivative of states is predicted from the current state thus governing the changes of the whole system.

Propagation: As mentioned in Chapter 2, the propagation step contains the state propagation and the covariance propagation. The state $\mathbf{x}$ is propagated through a direct Euler integration of (3.4), while the covariance is propagated using the linear error state dynamics (3.5). The propagation rule in discrete forms is shown as below:

$$\begin{bmatrix} \hat{\mathbf{v}}_{t+1} \\ \hat{\mathbf{p}}_{t+1} \\ \hat{\mathbf{R}}_{t+1} \\ \hat{\mathbf{a}}_{b(t+1)} \\ \hat{\boldsymbol{\omega}}_{b(t+1)} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{v}}_t + [\hat{\mathbf{R}}_t(\mathbf{a}_m - \hat{\mathbf{a}}_{b(t)}) + \mathbf{g}]\Delta t \\ \hat{\mathbf{p}}_t + \hat{\mathbf{v}}_t\Delta t + \frac{1}{2}[\hat{\mathbf{R}}_t(\mathbf{a}_m - \hat{\mathbf{a}}_{b(t)}) + \mathbf{g}]\Delta t^2 \\ \hat{\mathbf{R}}_t \exp\left((\boldsymbol{\omega}_m - \hat{\boldsymbol{\omega}}_{b(t)})\Delta t\right) \\ \hat{\mathbf{a}}_{b(t)} \\ \hat{\boldsymbol{\omega}}_{b(t)} \end{bmatrix} \tag{3.6}$$

where Δt is the time length between two adjacent states and the function $\exp(\cdot) : \mathbb{R}^3 \to SO(3)$ maps the rotation vector $\boldsymbol{\theta}$ to the rotation matrix using the Rodrigues formula [102]:

$$\exp(\boldsymbol{\theta}) = \mathbf{I} + \boldsymbol{\Theta}\sin(\|\boldsymbol{\theta}\|) + \boldsymbol{\Theta}^2(1 - \cos(\|\boldsymbol{\theta}\|)),\ \boldsymbol{\Theta} = \left[\frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}\right]_\times \tag{3.7}$$

Then the covariance is propagated by linearizing the error state dynamics (3.5):

$$\bar{\mathbf{\Sigma}}_{t+1} = \mathbf{F}_x \mathbf{\Sigma}_t \mathbf{F}_x^T + \mathbf{F}_n \mathbf{Q}_n \mathbf{F}_n^T \tag{3.8}$$

where

$$\mathbf{F}_x = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & -\hat{\mathbf{R}}_t[\mathbf{a}_m - \hat{\mathbf{a}}_{b(t)}]_\times \Delta t & -\hat{\mathbf{R}}_t \Delta t & \mathbf{0} \\ \mathbf{I}_3 \Delta t & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}^T \left\{(\boldsymbol{\omega}_m - \hat{\boldsymbol{\omega}}_{b(t)})\Delta t\right\} & \mathbf{0} & -\mathbf{I}_3 \Delta t \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3 \end{bmatrix} \tag{3.9}$$

$$\mathbf{F}_n = \begin{bmatrix} \hat{\mathbf{R}}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_9 \end{bmatrix} \tag{3.10}$$

are the linearized coefficient matrices and

$$\mathbf{Q}_n = \begin{bmatrix} (\sigma_{a_n}\Delta t)^2 \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\sigma_{\omega_n}\Delta t)^2 \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\sigma_{a_w}\Delta t)^2 \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & (\sigma_{\omega_w}\Delta t)^2 \mathbf{I}_3 \end{bmatrix} \tag{3.11}$$

models the system transition noise being added to the integration. Here $\sigma_{a_n}, \sigma_{\omega_n}, \sigma_{a_w}, \sigma_{\omega_w}$ represent the Gaussian noise of accelerometer, gyroscope, accelerometer bias and gyroscope bias, respectively. The values of variances are assigned manually in practice.

3.3.2 The Measurement Model

Observation Model: With error state representation, the observation model is simply linear.

$$\delta\mathbf{y} = \mathbf{H}\delta\mathbf{x} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \end{bmatrix} \delta\mathbf{x} \tag{3.12}$$

Here $\delta\mathbf{y}$ denotes the pseudo pose measurement encoded in the laser scan. Other components in the state vector such as velocity and sensor biases are not directly observable from the sensing data. Therefore the corresponding entries of $\mathbf{H}$ are zeros.

Recover Pseudo Measurement: Intuitively, the pseudo measurement can be thought of as a pose measurement extracted from sensor readings. Fig. 3.2 illustrates the key steps to recover the pseudo measurements. More specifically, it is computed via the following procedures:

- Firstly, based on pose error priors $\delta\bar{\mathbf{x}}_{t+1}^m \in \mathbb{R}^6, \bar{\mathbf{\Sigma}}_{t+1}^m \in \mathbb{R}^{6\times6}$, a set of particles $\{\delta\mathbf{x}_i\}$ are sampled.
- Secondly, a weighting factor w_i is computed for each particle based on the likelihood field model [141] which characterizes the alignment quality between the laser scan and the prior map.

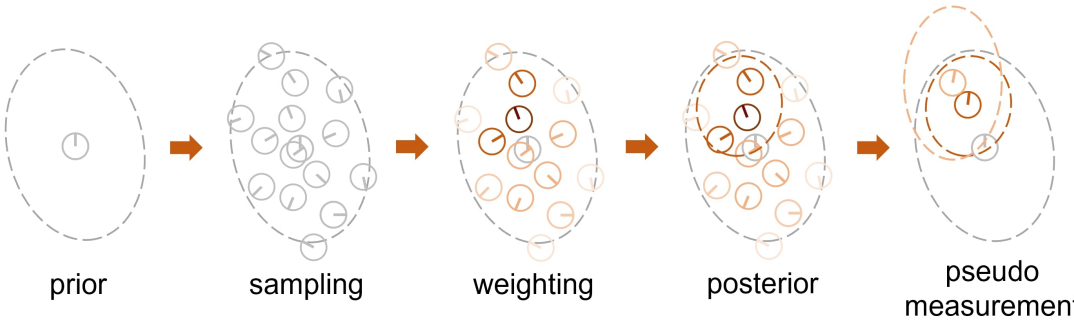


Figure 3.2: An illustration of the key steps of GPF in 2D. Pose samples and mean are represented as small circles with a straight bar to indicate the direction. The dashed ellipses illustrate the Gaussian distributions of prior, posterior and pseudo measurement.

- Thirdly, the pose posterior $\delta\mathbf{x}_{t+1}^m \in \mathbb{R}^6, \mathbf{\Sigma}_{t+1}^m \in \mathbb{R}^{6\times6}$ are computed as the weighted mean and covariance of the particles:

$$\delta\mathbf{x}_{t+1}^m = \frac{\sum w_i \delta\mathbf{x}_i^m}{\sum w_i} \tag{3.13}$$

$$\mathbf{\Sigma}_{t+1}^m = \frac{\sum w_i (\delta\mathbf{x}_i^m - \delta\mathbf{x}_{t+1}^m)(\delta\mathbf{x}_i^m - \delta\mathbf{x}_{t+1}^m)^T}{\sum w_i} \tag{3.14}$$

- Finally, a pseudo measurement $\delta\mathbf{y}_{t+1}^m \in \mathbb{R}^6$ and pseudo noise $\mathbf{C}_{t+1}^m \in \mathbb{R}^{6\times6}$ is recovered by inversing the KF measurement update:

$$\delta\mathbf{y}_{t+1}^m = (\mathbf{K}^m)^{-1}(\delta\mathbf{x}_{t+1}^m - \delta\bar{\mathbf{x}}_{t+1}^m) + \delta\bar{\mathbf{x}}_{t+1}^m \tag{3.15}$$

$$\mathbf{C}_{t+1}^m = ({\mathbf{\Sigma}_{t+1}^m}^{-1} - (\bar{\mathbf{\Sigma}}_{t+1}^m)^{-1})^{-1} \tag{3.16}$$

  where $\mathbf{K}^m = \bar{\mathbf{\Sigma}}_{t+1}^m \mathbf{H}^{mT}(\mathbf{H}^m \bar{\mathbf{\Sigma}}_{t+1}^m \mathbf{H}^{mT} + \mathbf{C}_{t+1}^m)^{-1}$ is the pseudo Kalman gain. $\mathbf{H}^m = \mathbf{I}_6$ is the pose-relevant block of $\mathbf{H}$.

Correction: Once the pseudo measurements are computed, it is used to update the full error states by a normal KF update. The Kalman gain is

$$\mathbf{K} = \bar{\mathbf{\Sigma}}_{t+1}\mathbf{H}^T(\mathbf{H}\bar{\mathbf{\Sigma}}_{t+1}\mathbf{H}^T + \mathbf{C}_{t+1}^m)^{-1} \tag{3.17}$$

Note $\mathbf{K} \in \mathbb{R}^{15\times6}$ is different to $\mathbf{K}^m \in \mathbb{R}^{6\times6}$ in that it now involves all dimensions of the state. The full error state posterior and the covariance are updated as

$$\delta\mathbf{x}_{t+1} = \mathbf{K}(\delta\mathbf{y}_{t+1}^m - \mathbf{H}\delta\bar{\mathbf{x}}_{t+1}) \tag{3.18}$$

$$\mathbf{\Sigma}_{t+1} = (\mathbf{I}_{15} - \mathbf{K}\mathbf{H})\bar{\mathbf{\Sigma}}_{t+1} \tag{3.19}$$

Update Nominal States: The updated errors are integrated into the previous nominal state:

$$\begin{bmatrix} \hat{\mathbf{v}}_{t+1} \\ \hat{\mathbf{p}}_{t+1} \\ \hat{\mathbf{R}}_{t+1} \\ \hat{\mathbf{a}}_{b(t+1)} \\ \hat{\boldsymbol{\omega}}_{b(t+1)} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{v}}_t + \delta\mathbf{v}_{t+1} \\ \hat{\mathbf{p}}_t + \delta\mathbf{p}_{t+1} \\ \hat{\mathbf{R}}_t \cdot \exp(\delta\boldsymbol{\theta}_{t+1}) \\ \hat{\mathbf{a}}_{b(t)} + \delta\mathbf{a}_{b(t+1)} \\ \hat{\boldsymbol{\omega}}_{b(t)} + \delta\boldsymbol{\omega}_{b(t+1)} \end{bmatrix} \tag{3.20}$$

Reset Error States: It is important to note that before the next iteration, the error states are set to zero:

$$\delta\mathbf{x}_{t+1} = \mathbf{0} \tag{3.21}$$

3.4 Localization Extended to Full SLAM

In this section, we discuss how the ESKF localization pipeline can be integrated with an off-the-shelf SLAM back-end library named Cartographer SLAM. We will briefly introduce the Cartographer SLAM then describe the combined pipeline in more detail.

Cartographer SLAM: Cartographer is an open-source LiDAR SLAM library developed at Google [65]. There are four key components in Cartographer:

- *Trajectory builder* implements an EKF-based state estimator that fuses IMU measurements with the LiDAR data. The estimator relies on scan to grid matching thus needs a 3D LiDAR to ensure a successful registration.
- *Map builder* builds local maps (also named submaps) by accumulating scans from a sliding window and creates new submaps when necessary.
- *Loop-closure detector* looks for possible matches between the latest scan and previous submaps. New constraints are created based on the found closures.
- *Pose graph optimizer* maintains the sparse pose graph (SPG) of the robot and submap poses and runs graph optimization to minimize the constraint residuals.

The idea of integration is to replace the trajectory builder with our ESKF-based state estimator and then make use of the back-end components of Cartographer for loop-closure detection and graph optimization.

Submaps and Distance Maps: A local occupancy grid map is defined as a submap by Cartographer. Since a different localization method is used, we need to adjust the submap management scheme so that the submaps can be accessed by the localization module. As shown in blue dashed box in Fig. 3.3, there exist two stages of scan accumulation. In the first stage, N 2D scans are accumulated to form a 3D scan (as the motor rotates) that is matched and inserted into the active submaps. The active submaps include a matching submap (blue) and a growing submap (yellow). The formed 3D scan is matched and inserted into both submaps. In the second stage, if M 3D scans are inserted into the matching submaps, the growing submap is switched to be the new

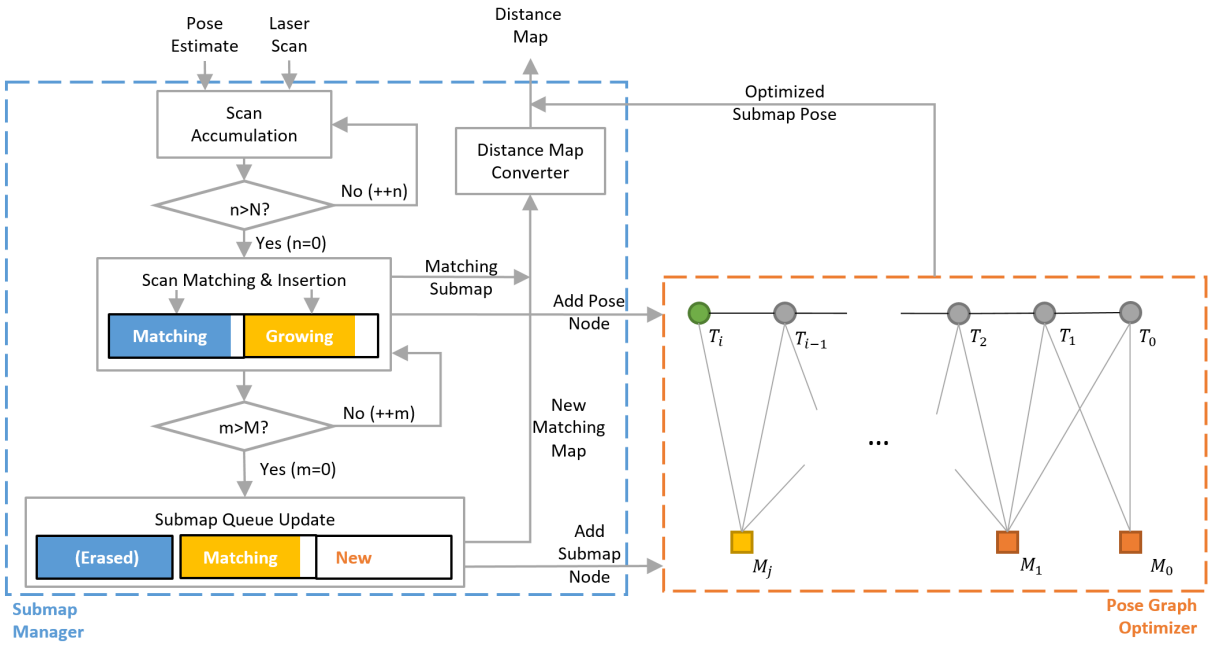


Figure 3.3: The modified pipeline based on Cartographer to construct submaps and manage the sparse pose graph.

matching submap and the old matching submap is erased. Meanwhile, a new submap (orange) is created and starts growing. During the two stages, whenever a 3D scan is formed or a new submap is created, new pose nodes are added to the SPG.

The adjustments are done by adding an octomap [68] beside the original grid map. The formed 3D scan is inserted into the octomap and the corresponding distance map is updated immediately. The octomap library provides an efficient method to detect changes so that the distance map can be computed efficiently. Additionally, the update of octomap and distance map are implemented in different threads to reduce time delay.

Sparse Pose Graph: As for the SPG, we use the graph solver of Cartographer to refine the poses of robot and submaps (see Fig. 3.3). The optimization is performed every few seconds and the updated submaps are converted into a distance map to be used for localization.

3.5 Experiments

3.5.1 System Overview

A customized quadrotor, as shown in Fig. 3.4, is used to test the localization algorithm. The quadrotor carries an onboard computer (Quad-core, 2GB RAM), an IMU (100 Hz), and a Hokuyo 2D laser scanner (270°FOV, 0.25° angle resolution, 40Hz) mounted on a continuously rotating motor (about 30 RPM). A motor encoder is used to project laser range data into the robot body

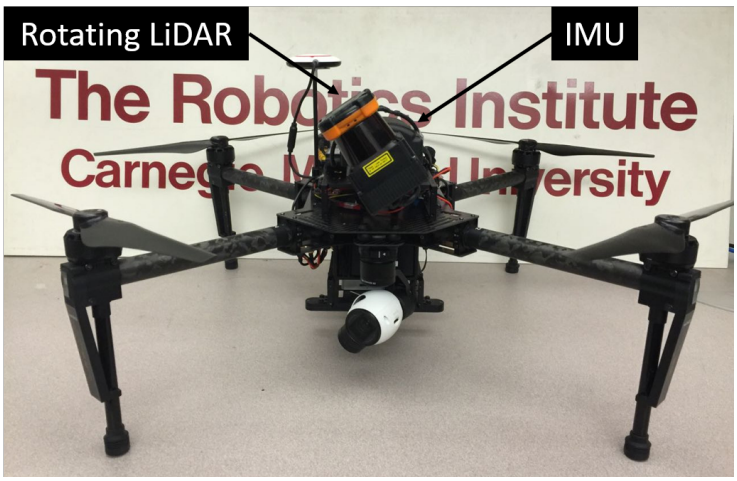


Figure 3.4: The customized robot platform with on board computation and sensing system (rotating laser scanner and IMU). The front camera is for inspection thus is not used for state estimation.

frame (z-axis downward, x-axis forward). The IMU frame is defined to coincide with the body frame as well.

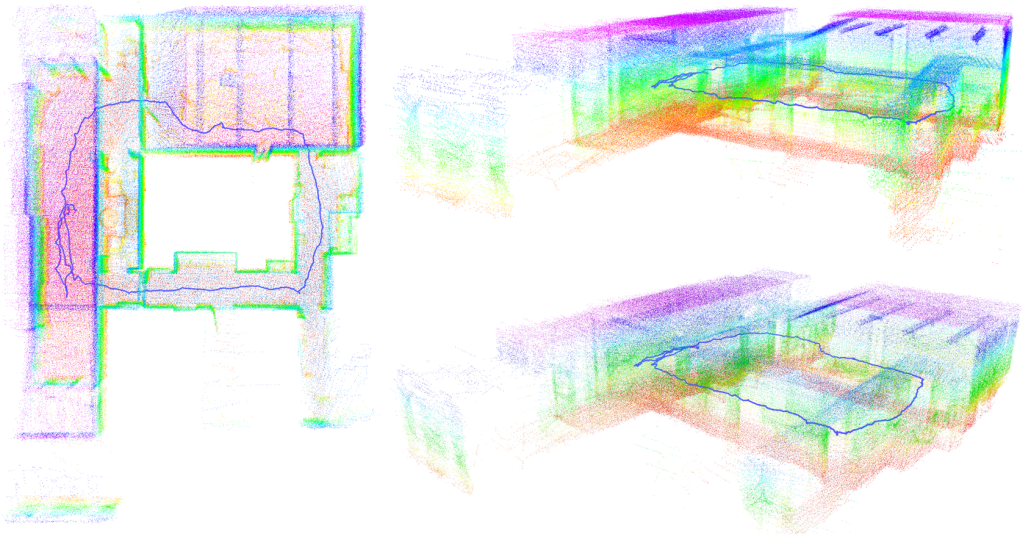
3.5.2 Indoor and Outdoor Tests

Fig. 3.5a shows an indoor test conducted around an office area. The prior map is converted into an occupancy map using the Octomap library [67]. The occupancy map has a scale of approximately $15 \times 20 \times 5m$, with $0.05m$ resolution. The robot is hand-held passing through corridors, a conference room, coffee lounge and eventually returns to the start point. The maximum velocity is about $1.2m/s$. In Fig. 3.5a, the reconstructed map aligns nicely with the original map, which implies the estimated poses are accurate.

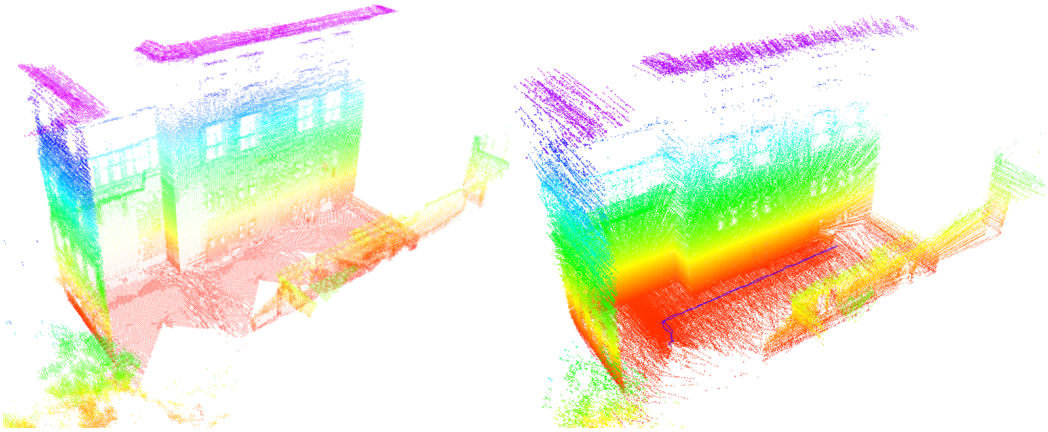
Outdoor spaces are wider open and pre-built maps have larger grid size ($0.1m$) than indoor ones, resulting in fewer details of structure to be captured for localization. This will cause an increase in the estimation error and uncertainties. Fig. 3.5b visualizes the localization results in an inter-building area. The map is of size $15 \times 28 \times 9m$. The robot moves laterally with a maximum speed at about 0.8 m/s while yawing between $-45°$ and $45°$ before returning to the origin (as indicated by Fig. 3.9, upper plots). By observation, one can tell that the overall estimation is quite accurate except some reconstructed building edges are slightly 'fuzzy' due to the estimation noise.

3.5.3 Robustness Tests

The robustness of the localization algorithm is measured by its capacity to recover from errors. It is tested in various settings using simulated or real data and also compared to an implementation of nominal state EKF. The two algorithms are sharing identical parameters such as sensor noise,



(a) Localization results in office area. Point clouds on the left and upper right are reconstructed with estimated pose trajectory (blue lines). Original reference map is shown at the lower right for comparison with reconstructed map.



(b) Localization results in an outdoor patio area. Point cloud on the left is the prior map while the point cloud on the right is the reconstructed map using estimated poses.

Figure 3.5: Indoor and outdoor tests.

map resolution and so on. We show that the ESKF outperforms the EKF in terms of robustness.

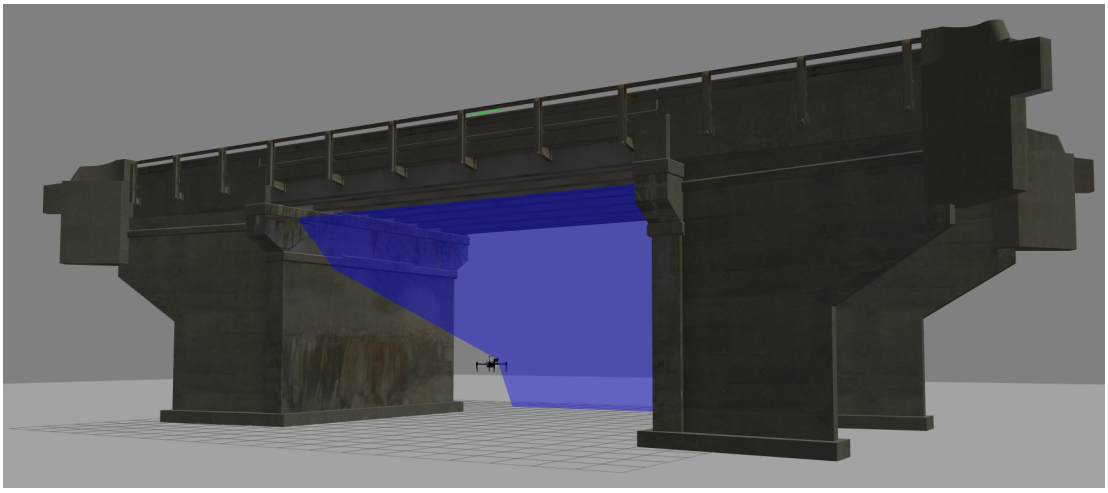


Figure 3.6: A simulation of robot localization under a bridge model.

The first robustness experiment is a kidnapped robot test in simulation (see Fig. 3.6). In this test, the algorithm assumes robot starts at the origin, however the actual initial height is set with different values (from 0.5m to the maximum error that can be corrected eventually). As shown in Fig. 3.7, the ESKF implementation has an error tolerance of about 4 meters while the EKF implementation can only handle error no larger than 0.6 meter. Additionally, ESKF converges faster than the EKF. For instance, it takes about 2 seconds for the ESKF algorithm to recover from an error of 0.5 meter, while the EKF algorithm needs more than 8 seconds to converge.

The second experiment is designed to compare the localization algorithms using real world data. In this experiment, the robot orientation is kept pointing to the wall, while moving laterally towards right and then left. The effective range of laser is reduced from 30m to 10m, which eliminates some useful structures for localization and degrades the state estimate. The 30m-range state estimates are then used as ground truth. Fig. 3.8 shows the localization error of the two algorithms. We can observe that the ESKF algorithm recovers from some small errors (about $0.7m$ at maximum), while the EKF algorithm diverges at the end.

One interesting observation from this experiment is that the localization algorithm performance could be affected by the heading direction of the robot while following the same route. When moving from left to right, the robot orientation is fixed. Thus at some places most of the captured data points lie on the wall and ground (as shown by the range plot in Fig. 3.9), which weakens the constraints along the direction of motion. This results in relatively larger pose errors (blue lines of lower plots in Fig. 3.9).

When moving back, the robot is pointed at nearby structures. Since the robot has a rotating 2D laser, its measurements are densest along the axis of rotation (denoted as the thick bar in the upper plot of Fig. 3.9). The second path is observed to have decreased position errors.

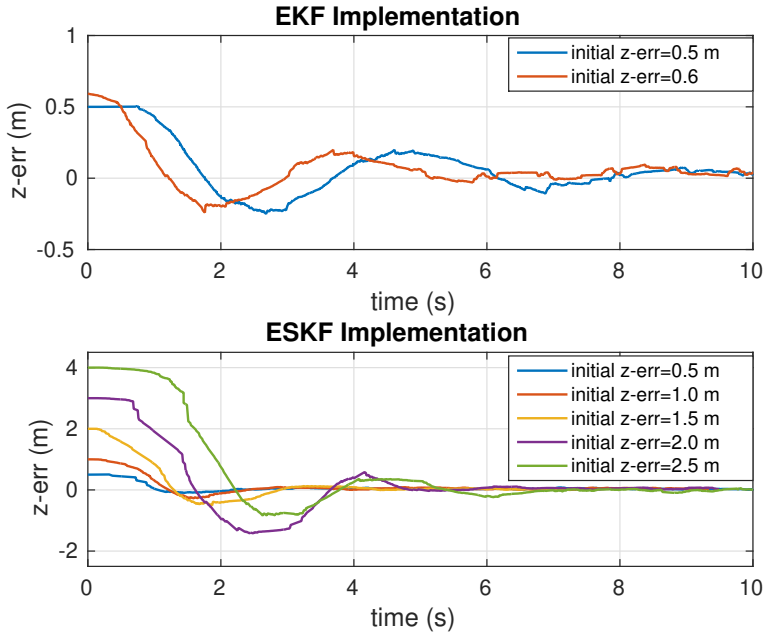


Figure 3.7: Kidnapped robot test. The initial error tolerance is found by increasing the initial error until the algorithm diverges. Note that diverging cases are not visualized here.

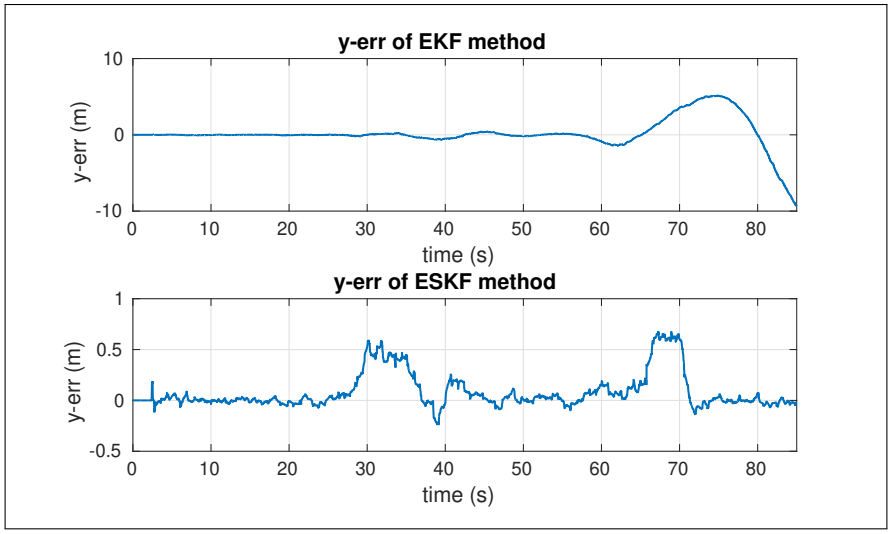


Figure 3.8: Localization error in y-axis with reduced range laser measurements.

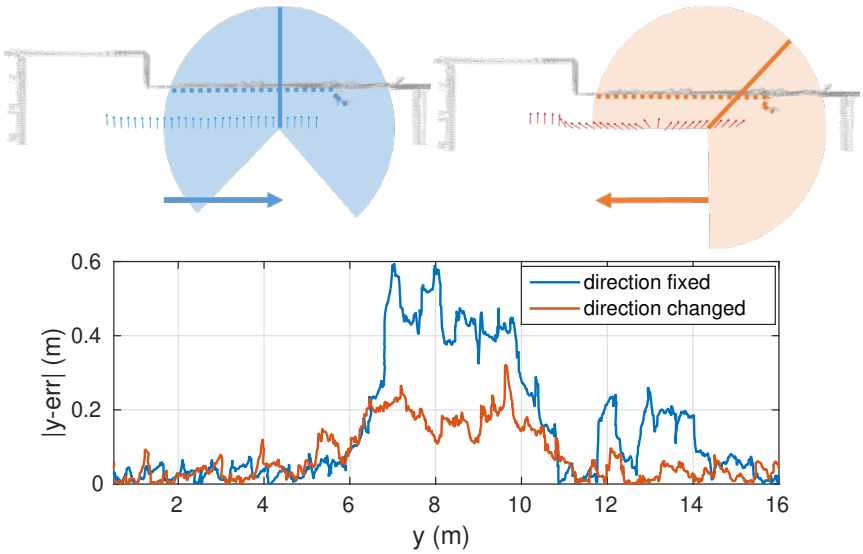


Figure 3.9: *Up:* Illustration of the robot movement and laser ranges. *Bottom:* The positioning errors of two moving trajectories. Only error on y-axis are visualized here since errors on the other two axis are close to zero.

3.5.4 Fast Motion Test

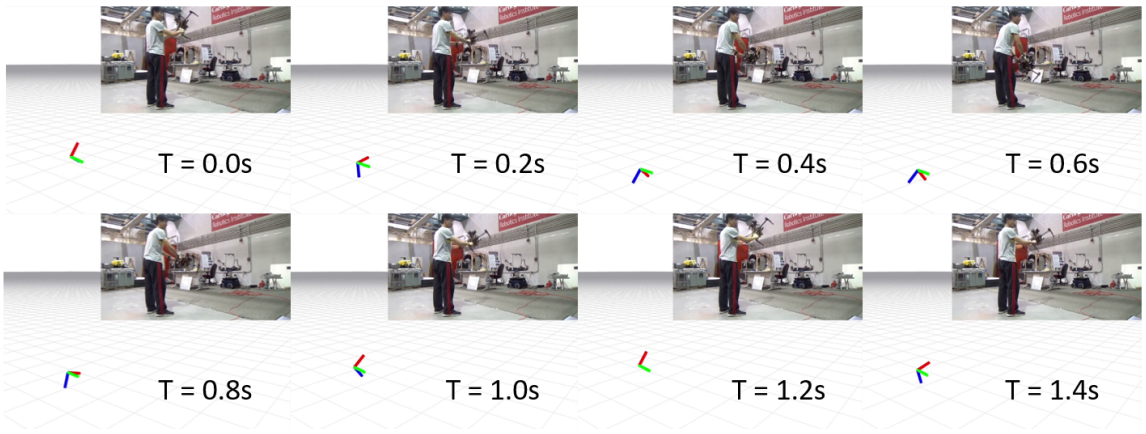


Figure 3.10: A sequence of video frames showing state estimation for aggressive motions.

Now we report the algorithm's performance under aggressive motions in both translation and rotation. In Fig. 3.10, we show a sequence of frames from a video[1] where the robot is hand-held and changes its position and rotation aggressively. In this sequence, the maximum rotational speed is about $150°/s$ and the maximum translational speed is about $3m/s$. We can see that even with a 2D LiDAR, the robot can reliably estimate its motion thanks to the fast responses from IMU. Additionally, fast flight tests[2] are conducted on the CMU campus, with a top flight speed of

[1]Aggressive motion video: `https://www.youtube.com/watch?v=TTQVluAu2eU`
[2]Fast flight video: `https://youtu.be/r8FA-OtnaMo?t=20`

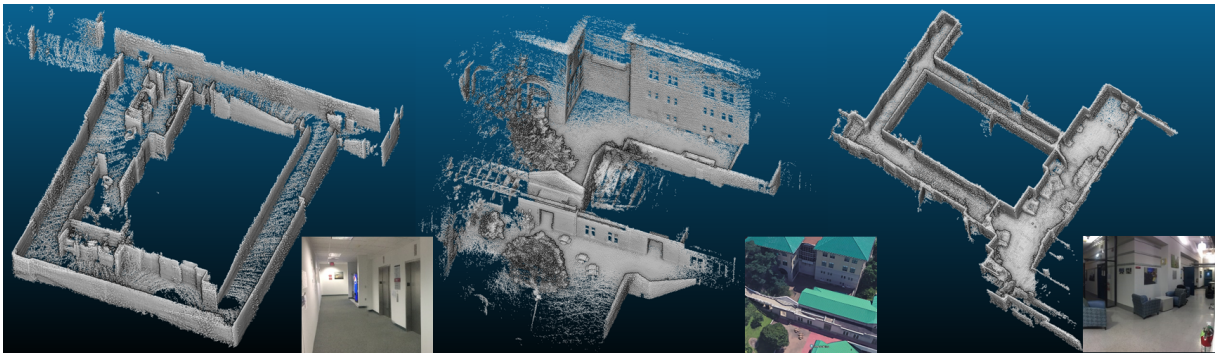


Figure 3.11: Three tests are conducted in indoor and outdoor environments. *Left*: Test with a fixed 3D LiDAR in a hallway loop. *Middle*: Test with a rotating 3D LiDAR around the CMU patio. *Right*: Test with a rotating 2D LiDAR in a corridor loop.

$2.5m/s$. Robot poses are estimated in real-time using the onboard computer (i.e. Nvidia TK1).

3.5.5 SLAM Tests

Tests of the SLAM pipeline are carried out in the real world on multiple platforms: (1) fixed VLP-16 (10Hz, range 100m), (2) rotating VLP-16, (3) rotating Hokuyo (40Hz, range 30m).

The first experiment is conducted inside a corridor loop (see Fig. 3.11 left). In this test, the VLP-16 is fixed horizontally on the robot. We found that although the VLP-16 measures 3D structures, its 30° vertical FOV is still not enough to reliably estimate height. The main reason is that inside the narrow corridor, most laser points come from side walls instead of the ground and ceiling. As a result, a larger drift in height is observed when the robot revisits the origin and loop-closure is harder to detect.

The second test is carried out around the patio on the CMU campus. The VLP-16 is mounted on a motor with the rotating speed set to $30rpm$. Since this is a larger area, the distance map used for localization has a coarser resolution of $0.3m$ and is constrained within a $40m \times 40m \times 40m$ bounding box.

In the last test, the robot maps a narrow hallway with 1.1m width at minimum. To ensure enough laser points are collected, the robot is manually carried and moved slowly ($\approx$ 0.5m/s). This time only small drifts in height is observed before closing the loop. This is because by rotating the LiDAR, the robot obtains wider FOV, which could significantly improve the mapping performance.

3.6 Conclusion and Discussion

In this work, we present a robust localization approach fusing IMU and laser range data into an ESKF framework. The algorithm is tested in an online setting and it is robust in various

environments with different characteristics and scales. Additionally, built on top of Cartograph SLAM, we extend the localization algorithm to a full SLAM pipeline and have shown the results of using different setups of LiDAR sensors.

Finally, we offer a discussion of several insights and issues that emerged from this work:

- On the linearization of non-linear systems to propagate uncertainty. In both ESKF and EKF, the uncertainty estimation is achieved by linearizing the original system around the current state, which inevitably introduces inaccuracy to the system model. However, in practice, as long as the robot moves at a moderate speed and the sensing frequency is adequately high, the linearization point is not far away from the actual state and the approximation remains valid.
- How would localization fail? Generally speaking, the observed information determines whether the estimation of unknowns is successful or not. A more technical term *observability* [70][84] has been used to evaluate whether the measurement has enough information content to uniquely and tightly constrain the unknowns. In the experiment, we showed the localization accuracy could be improved by simply letting the LiDAR observe more points. This fact brings up an interesting question: how can we quantitatively predict whether the localization would fail or not? We will take a closer look at this problem in Chapter 4.
- On the accumulated drifts upon loop-closure. Loop-closure effectively corrects the accumulated drifts by averaging out the total error across all the frames involved in the loop. However, averaging does not imply the exclusion of errors. In fact, loop-closure happens in the back-end, where it is difficult to evaluate every constraint built from the front-end to eliminate the source of drifts. Alternatively, directly formulating the error function based on raw sensor observations could bridge the front and the back ends, improving the overall accuracy at the cost of increased computations. In Chapter 5, we will try to formulate a batch optimization problem that directly depends on raw sensing data such that outliers can be evaluated and rejected to guarantee the total consistency.

Chapter 4

Degeneracy in State Estimation

In the previous chapter, we presented an algorithm that combines LiDAR and IMU for real-time localization. Although the algorithm is shown robust to the perturbance of sensor readings and aggressive motions, it could fail easily in environments with *geometric degeneracy*. One typical example of a degenerated environment is a long straight tunnel. The primary reason for localization failure is the ambiguity due to the lack of geometric features. In this chapter, we look at the robust localization problem in degenerated environments and discuss what we can learn from such failures.

This chapter is structured as follows: Section 4.1 defines the problem of localizability prediction. Section 4.2 reviews works that are related to this chapter. Section 4.3 describes the developed localizability model in detail. In Section 4.4, we present a sensor fusion algorithm that combines LiDAR, IMU together with an Ultra-WideBand (UWB) radio for robust localization in tunnels. Experimental results are reported in Section 4.5 and conclusions are drawn in Section 4.6.

4.1 The Localizability Problem

Since a LiDAR sensor captures the geometry information by scanning the environment, it is likely to be affected in geometrically degenerated cases. For example, a robot navigating through a long straight tunnel (as seen in Fig. 4.1 top-left) will not be able to determine its location along the tunnel since the measurements are identical everywhere. We can understand this situation with an analogy to a sliding block inside a pipe (see Fig. 4.1 bottom-left). The block is obtained by connecting the endpoints of the laser scan. The contact forces, analogous to the surface normals, prohibit motions towards the sides of the pipe. However, since there is no friction to restrain the object, its motion along the pipe becomes under-constrained.

To identify the geometric degeneration in general environments, a mathematical model to predict the localizability is needed. Therefore, the problem to be addressed in this chapter can be stated as:

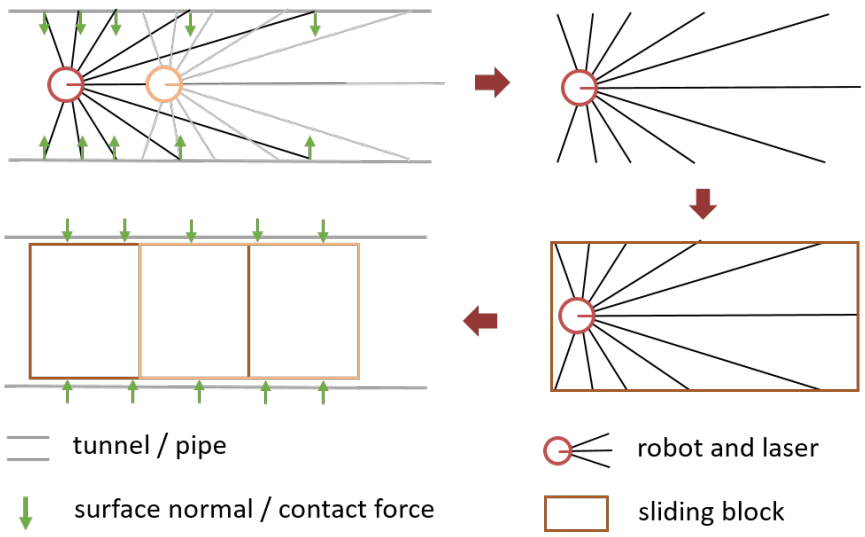


Figure 4.1: An analogy between robot navigation inside a tunnel and a sliding block inside a pipe, where the measured surface normals correspond to contact forces.

Given the prior map and the sensor measurements, how can we qualitatively and quantitatively determine whether the data have enough information to estimate the robot motion?

The term *localizability* shares the same meaning of *observability* [64][84] in general estimation problems. In this work, we choose to use localizability to specifically describe the quality of sensor measurements in terms of information content that can be used for localization. The localizability can be defined for both the prior map and the sensor measurements. If defined for prior maps, the localizability can be used to predict whether a robot is safe to navigate through a specific region. For example, one may analyze the map in an offline manner before the real deployment of robots. The localizability of sensor measurements, on the other hand, describes whether the current sensor readings can privide enough constraints for motion estimation. In this case, one may use the localizability to predict failures in an online manner.

4.2 Related Work

Modeling sensing degeneracy is a research topic closely related to uncertainties, which is critical to the robustness of a system. Therefore, in this section, we review the efforts made to model and predict uncertainties in estimation problems. More specifically, we discuss prior works on LiDAR sensing degeneracy in detail, followed by a brief review of works using cameras.

LiDAR sensing degeneracy: For LiDAR-based approaches, perhaps the earliest attempt is the work from Roy and Thrun [121] known as *coastal navigation*. Their model is formulated in a probabilistic framework but needs approximations to compute the uncertainty efficiently. Instead of modeling the laser uncertainty directly, Diosi and Kleeman [39] compute the uncertainty of

line segments extracted from laser scans. Since ICP is a critical component of LiDAR-based state estimation and mapping algorithms, there are works studying the uncertainty of ICP algorithms. Bonnabel et al. [19] provide an analytical derivation of the covariance matrix for ICP-based registration problems. Gelfand et al. [51] show that sampling points that better constrain the motion estimation would effectively make the registration process more stable. Censi [26] derives a lower bound of the uncertainty matrix that 2D ICP algorithms can achieve using Cremar-Rao's bound, which directly inspires the development of our method. His later work extends the idea to pose tracking [27] and localization [28]. Zhang et al. [157][155] use a *degeneracy factor* to characterize geometric degeneration and improve the accuracy of ego-motion estimation in the context of solving an optimization problem. More recently, LiDAR-inertial observability-aware systems [138][3] are developed and successfully applied to environments with geometric degeneracy.

Visual sensing degeneracy: In the case of cameras, available approaches typically try to estimate the uncertainty of extracted features. For example, Eudes and Lhuillier [43] model the error propagation from image pixels to reconstructed 3D points in the settings of a bundle adjustment problem. Yang et al. [152] offer an analysis of the degenerate motion for visual odometry using line features. Similar studies on line degeneration are carried out in [135] [160]. Different from geometric analysis, Vega et al. [148] propose a learning approach to predict uncertainty matrix based on the system's experience. This approach is demonstrated using cameras and LiDARs in the simulation. More recently, as deep learning techniques are gaining more attention, neural network empowered methods [88] are developed to predict the covariance matrix for state estimation.

Our Localizability model presented in this chapter shares a similar idea with [27] and [157] in that the sensitivity of measurements w.r.t. parameters is used to identify degeneration. But we formulate the sensitivity from a constraint set and use a physically meaningful metric to evaluate the localizability.

4.3 The Localizability Model

The goal of modeling the degeneration of geometry is to develop theoretical tools to identify degeneration in given maps and also gain insights on designing reliable sensing systems. In other words, given the prior map, we would like to answer whether the current measurement from a specific sensor contains enough information to estimate the robot state.

4.3.1 Localizability of the LiDAR

The LiDAR sensor measures the ranges along an array of laser beams. Therefore, each beam returns an endpoint indicating where the laser hit the surface of an object. The collection of all endpoints will form a laser point cloud that shows a sparse sample of the environment geometry. Based on the scanned point cloud, LiDAR-based motion estimation is carried out by aligning

the shapes of two point clouds. In the case of localization, the problem becomes estimating the sensor position and orientation such that the laser scan aligns with a prior map.

In this section, we represent the LiDAR-based localization problem as solving a set of constraint equations:

$$\mathcal{C}(\mathbf{x}, \mathbf{R}, \rho_i) = \mathbf{n}_i^T(\mathbf{x} + \mathbf{R}\mathbf{r}_i\rho_i) + d_i = 0 \tag{4.1}$$

where $(\mathbf{x}, \mathbf{R}) \in (\mathbb{R}^3, SO(3))$ denotes the robot position and orientation, and $i \in \{1, 2, \cdots, m\}$ is the point index in the laser scan. $(\mathbf{n}_i, d_i) \in (\mathbb{R}^3, \mathbb{R})$ encodes the normal vector and distance of the local plane in the environment. In practice, this local plane is not readily available from sensor data but can be estimated from the neighboring points. $\mathbf{r}_i \in \mathbb{R}^3$ is the unit range vector indicating the direction of a laser beam in the robot body frame and $\rho_i \in \mathbb{R}$ is the range value. (4.1) describes a simple fact that the scanned points should align with the map when the robot is localized (illustrated in Fig. 4.2).

Now we evaluate the *strength* of the constraint by measuring the sensitivity of measurements w.r.t. the robot pose $(\mathbf{x}, \mathbf{R})$. The key observation is that if the robot pose is perturbed slightly but the resulting measurements do not change much, then the constraint is weak. Otherwise, the constraint is strong. Therefore, it is a natural idea to compute the derivative of ρ_i w.r.t. $\mathbf{x}$ and $\mathbf{R}$ as a measure of the sensitivity.

Note that (4.1) is formulated for a single end point. Taking all end points into consideration results in a set of constraints. Stacking the derivatives computed from all the constraints gives two matrices:

$$\mathbf{F} = \begin{bmatrix} -\dfrac{\mathbf{n}_1}{\mathbf{n}_1^T\mathbf{r}_1} & \cdots & -\dfrac{\mathbf{n}_m}{\mathbf{n}_m^T\mathbf{r}_m} \end{bmatrix} \tag{4.2}$$

$$\mathbf{T} = \begin{bmatrix} -\dfrac{\rho_1\mathbf{r}_1 \times \mathbf{n}_1}{\mathbf{n}_1^T\mathbf{r}_1} & \cdots & -\dfrac{\rho_m\mathbf{r}_m \times \mathbf{n}_m}{\mathbf{n}_m^T\mathbf{r}_m} \end{bmatrix} \tag{4.3}$$

Here we directly give the collection of derivatives while details for computing $\frac{\partial \mathcal{C}}{\partial \mathbf{x}}$ and $\frac{\partial \mathcal{C}}{\partial \mathbf{R}}$ can be found in Appendix A.1. We could then perform Eigenvalue Decomposition on the information matrices:

$$\mathbf{F}\mathbf{F}^T = \mathbf{U}_\mathbf{F}\mathbf{D}_\mathbf{F}\mathbf{U}_\mathbf{F}^T \tag{4.4}$$

$$\mathbf{T}\mathbf{T}^T = \mathbf{U}_\mathbf{T}\mathbf{D}_\mathbf{T}\mathbf{U}_\mathbf{T}^T \tag{4.5}$$

and any eigenvalues significantly smaller than the others indicate degeneration in the direction of the corresponding eigenvectors.

A straightforward choice of the metric to evaluate the degeneration is the eigenvalues or condition number. However, we found this metric difficult to interpret because its physical meaning is not clear. To accommodate this issue, we choose to project each row in $\mathbf{F}$ and $\mathbf{T}$ into the eigenspace

$$\mathbf{F}' = \mathbf{U}_\mathbf{F}\mathbf{F} \tag{4.6}$$

$$\mathbf{T}' = \mathbf{U}_\mathbf{T}\mathbf{T} \tag{4.7}$$

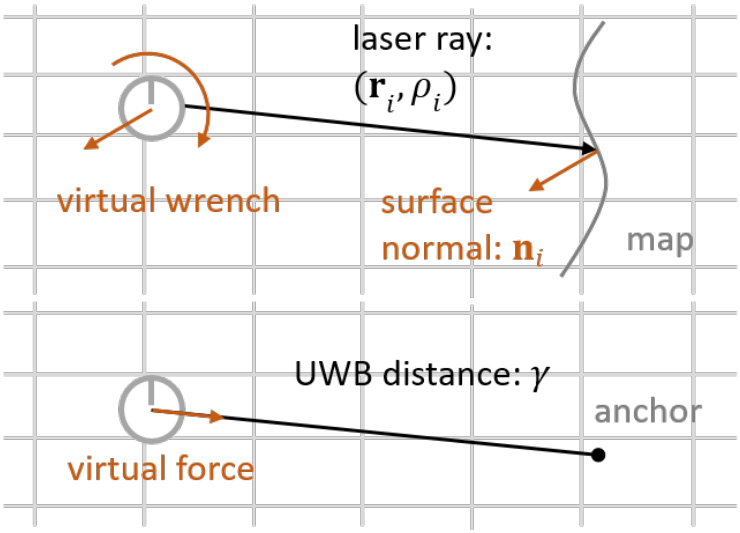


Figure 4.2: An illustration of the visual wrench restraining the robot position and orientation. *Up:* The constraint model of LiDAR end point. *Bottom:* The constraint model of UWB ranging measurement.

where $\mathbf{F}'$ and $\mathbf{T}'$ are projected derivatives over position and orientation respectively. Then we define the localizability vector $\mathbf{l}_\mathbf{F} \in \mathbb{R}^3$ and $\mathbf{l}_\mathbf{T} \in \mathbb{R}^3$ as

$$\mathbf{l}_\mathbf{F} = \sum_{i=1}^{m} \text{abs}(\mathbf{F}'_i), \quad \mathbf{l}_\mathbf{T} = \sum_{i=1}^{m} \text{abs}(\mathbf{T}'_i) \tag{4.8}$$

where $i = 1, \cdots, m$ is the index of points. (4.8) indicates that each element in the localizability vector is the sum of absolute values of each row in $\mathbf{F}'$ and $\mathbf{T}'$, respectively.

A closer look at (4.2), (4.3) and (4.8) gives a more natural and intuitive interpretation. As illustrated in Fig. 4.2, we can interpret the position constraints as forces in the direction of $\mathbf{n}_i$ (ignoring the signs) and the orientation constraints as torques in the direction of $\mathbf{r}_i \times \mathbf{n}_i$. Now the $\mathbf{F}$ and $\mathbf{T}$ are collections of wrenches (forces and torques) restraining the translation and rotation of the robot. Aligning with this picture, well-conditioned $\mathbf{F}$ and $\mathbf{T}$ indicate a *frictionless force-closure*, which is a term used in the field of manipulation mechanics to describe a solid grasp of an object. The characterization of a frictionless force-closure is to check whether the row vectors in $\mathbf{F}$ and $\mathbf{T}$ span the space of $\mathbb{R}^3$ [102]. Interestingly, this shares a similar idea of identifying degeneration using small eigenvalues. Furthermore, we can interpret the physical meaning of the localizability as the magnitude of accumulated virtual forces and torques gained from the measurements to restrain the robot pose.

4.3.2 Localizability of UWB Ranging

The Ultra-Wideband (UWB) sensor measures the distance from the anchor (attached to the environment) to the target (attached to the robot). Assuming the target is located at the origin of the

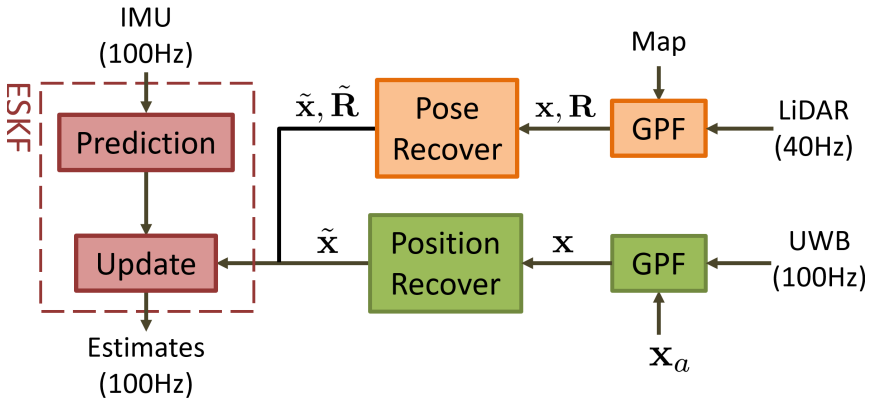


Figure 4.3: Sensor fusion system overview. The green blocks indicate pipeline to process the UWB measurements and is described in detail in this chapter. The red and orange blocks are pipelines of the ESKF and LiDAR fusion respectively.

robot body frame, we get the constraint equation

$$\mathcal{C}(\mathbf{x}, \mathbf{R}, \gamma) = ||\mathbf{x} - \mathbf{x}_a||^2 - ||\gamma||^2 = 0 \tag{4.9}$$

where $\mathbf{x}_a \in \mathbb{R}^3$ is the anchor position in the environment and $\gamma \in \mathbb{R}$ is the measured range. Following similar procedures, we obtain the force matrix $\mathbf{F}$ for the UWB

$$\mathbf{F} = \frac{\mathbf{x} - \mathbf{x}_a}{\gamma} \tag{4.10}$$

(see Appendix A.2 for more details). Again, $\mathbf{F}$ can be treated as a collection of unit forces. In fact, there is only one column in $\mathbf{F}$ and thus represents a single force in the direction from the anchor to the target. The force is later projected into the previously derived eigenspace to be compared with the LiDAR localizability. On the other hand, since the sensor does not provide any information about the orientation, the torque matrix $\mathbf{T}$ is trivially zero:

$$\mathbf{T} = \mathbf{0} \tag{4.11}$$

4.4 Probabilistic Fusion of LiDAR and UWB

The fusion of IMU, LiDAR and the UWB is based on the Error State Kalman Filter (ESKF) framework presented in Chapter 3. An ESKF is very similar to the well-known Extended Kalman Filter (EKF), except it models the system dynamics in error states, which is beneficial when linearizing the system [134].

Here we briefly revisit the process of the ESKF filtering algorithm. In the ESKF, the IMU measurements are integrated over time to predict the robot states. And the laser scans are matched to the prior map to recover a 6D pose measurement which is further used to update the prediction. Different from the LiDAR, the UWB ranges are converted to a 3D position measurement since

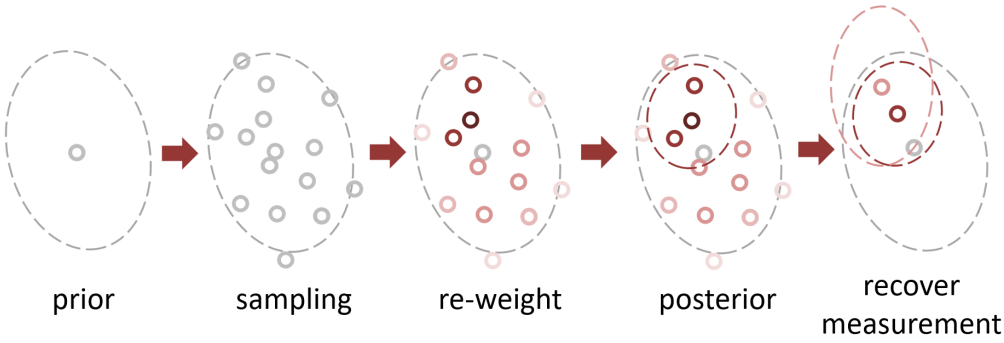


Figure 4.4: An illustration of the GPF in 2D. Grey ellipse: uncertainty of prior belief. Dark red ellipse: the uncertainty of posterior. Light red ellipse: uncertainty of the recovered position measurement. The color of a particle encodes its weight with darker color corresponds to a higher weight.

the UWB doesn't contain any orientation information. After that, the recovered position is used to update the prediction similarly. Note that the update rate from LiDAR and UWB is different and depends on the data frequency. Fig. 4.3 depicts the pipeline adapted to handle both LiDAR and UWB measurements. In this section, we elaborate on the pipeline to process UWB data. More details of handling LiDAR measurements can be found in the previous chapter.

Specifically, a Gaussian Particle Filter (GPF) is used to convert the UWB ranges to a pose measurement as in [22]. Fig. 4.4 illustrates the process of GPF using a 2D example. First, a set of particles $\{\mathbf{x}_i \in \mathbb{R}^3 | i = 1, 2, \cdots, N\}$ are drawn based on the position partition $(\bar{\mathbf{x}}, \bar{\mathbf{\Sigma}}) \in (\mathbb{R}^3, \mathbb{R}^{3\times3})$ of the full prior belief which consists of position, orientation, velocity and so on. Additionally, each particle is assigned with a weighting factor

$$w_i = \exp\left[-\left(\frac{||\mathbf{x}_i - \mathbf{x}_a|| - \gamma}{\sigma}\right)^2\right] \tag{4.12}$$

where σ is the ranging noise of the UWB and is tuned by hand in the experiments. This weighting factor measures how likely is each particle to be the true state. After that, the position posterior $(\mathbf{x}, \mathbf{\Sigma})$ is found by computing the weighted mean and covariance of the set

$$\mathbf{x} = \frac{\sum_i w_i \mathbf{x}_i}{\sum_i w_i} \tag{4.13}$$

$$\mathbf{\Sigma} = \frac{\sum_i w_i (\mathbf{x}_i - \mathbf{x})(\mathbf{x}_i - \mathbf{x})^T}{\sum_i w_i} \tag{4.14}$$

With the prior and the posterior belief in hand, we differentiate them to recover a position measurement $(\tilde{\mathbf{x}}, \tilde{\mathbf{\Sigma}}) \in (\mathbb{R}^3, \mathbb{R}^{3\times3})$

$$(\tilde{\mathbf{x}}, \tilde{\mathbf{\Sigma}}) = (\mathbf{x}, \mathbf{\Sigma}) \ominus (\bar{\mathbf{x}}, \bar{\mathbf{\Sigma}}) \tag{4.15}$$

by inversing the Kalman update step

$$\tilde{\mathbf{x}} = \mathbf{K}^{-1}(\mathbf{x} - \bar{\mathbf{x}}) + \bar{\mathbf{x}} \tag{4.16}$$

$$\tilde{\mathbf{\Sigma}} = (\mathbf{\Sigma}^{-1} - \bar{\mathbf{\Sigma}}^{-1})^{-1} \tag{4.17}$$

where $\mathbf{K}$ is the Kalman gain computed as

$$\mathbf{K} = \bar{\mathbf{\Sigma}}(\bar{\mathbf{\Sigma}} + \tilde{\mathbf{\Sigma}})^{-1} \tag{4.18}$$

Here the observation matrix is an identity and hence omitted. Finally, $(\tilde{\mathbf{x}}, \tilde{\mathbf{\Sigma}})$ is used to update the full state in the ESKF.

4.5 Experiments

4.5.1 Experiment Overview

Figure 4.5: *Left:* The Smith Hall tunnel at CMU with the UWB anchor board placed on the ground. *Right:* The customized DJI M100 quadrotor.

Experiments are carried out inside the Smith Hall tunnel at CMU (shown in Fig. 4.5). The tunnel is of size $35m \times 2.4m \times 2.5m$ $(l \times w \times h)$ with pipes on both sides. The prior map is obtained by aligning multiple local scans along the tunnel. The robot (see Fig. 4.5) is a customized DJI Matrice 100 quadrotor. It has a rotating Hokuyo UTM-30LX-EW LiDAR (40Hz, 30m range), a Microstrain IMU (100Hz), a Pozyx UWB target board (100Hz, 100m range with clear line-of-sight), and a DJI Manifold computer (2.32GHz). Note that the Hokuyo LiDAR is mounted on a continuously rotating motor (180°/sec) and the laser scan is projected into the robot body frame using the encoder angles. There are other sensors such as GPS, compass and a gimbal camera that are not used in this work.

4.5.2 Localizability in Tunnels

The LiDAR localizability is evaluated at 20 evenly sampled places along the tunnel (as shown in Fig. 4.6). Firstly, we define the map frame with x pointing along the tunnel and z pointing

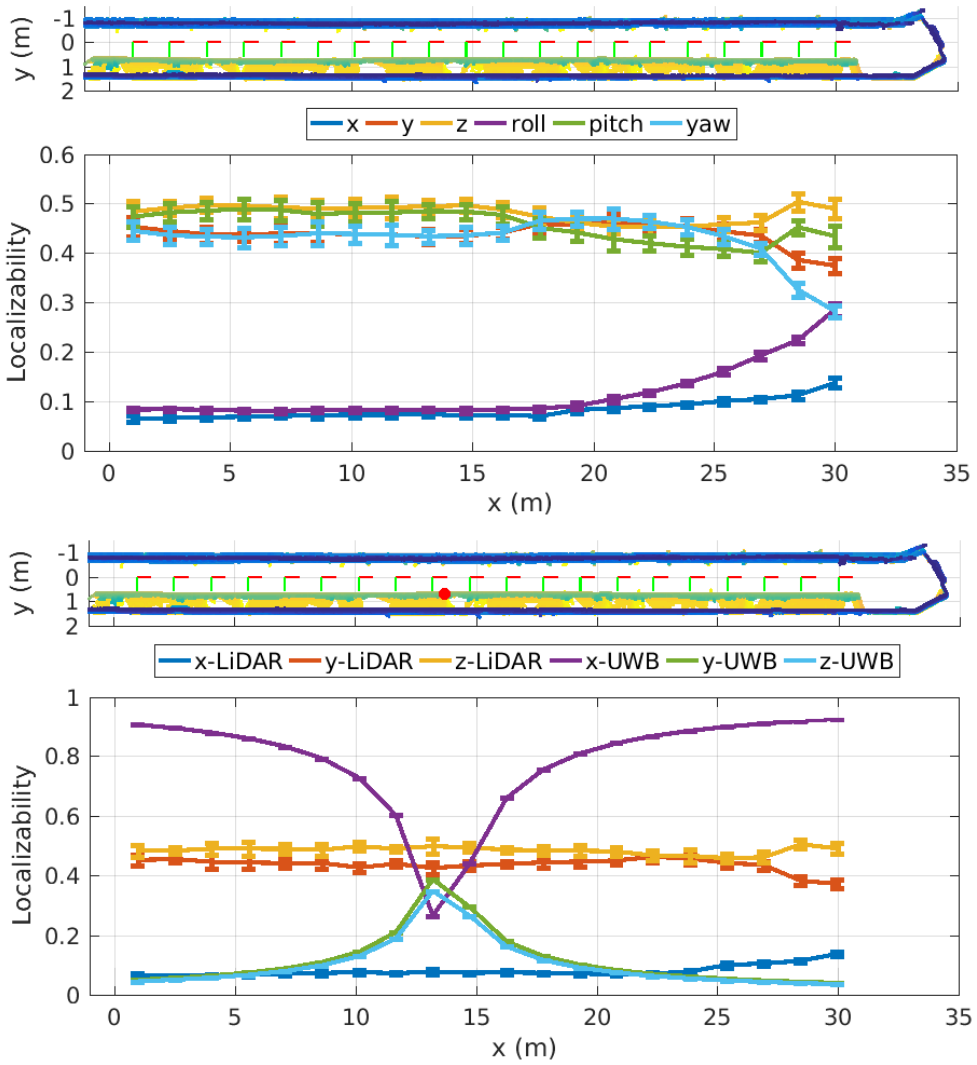


Figure 4.6: *Top:* The LiDAR localizability along the tunnel. The upper plot is a top-down view of the pre-built map where the ground and ceiling points are cropped and the color encodes the height of remaining points. The red-green (x-y) frames indicate the 20 sampled positions. *Bottom:* A comparison of position localizability of the LiDAR and the UWB. The red dot marks the position of the UWB anchor in the tunnel.

downward. Secondly, to simulate the measurements at each place, 4000 points are sampled uniformly within the range of 15 meters. We choose 4000 because that is about the amount of downsampled laser points used for localization per $180°$ rotation of the motor. And the effective range of LiDAR is decreased since distant points have nearly $90°$ reflection angle resulting in unreliable measurements. Then, at each point, a local surface is estimated by fitting a plane to its 20 nearest neighbors. Finally, using Eqn. 4.2-4.8, the localizability vector can be computed. The computation is repeated 10 times with a different set of points and results are averaged. Fig. 4.6 shows a top-down view of the sampled poses and their localizability. In order to show the

strength of the localizability clearly in the comparison, the values are scaled as

$$\mathbf{l_i} = \frac{\mathbf{l}_i}{\sum_i \mathbf{l}_i} \quad (i = 1, 2, 3) \tag{4.19}$$

It can be observed that the position and orientation localizability along the x-axis is significantly smaller than the other two dimensions. This is because the position x is ambiguous along the tunnel except at the right end where a vertical wall restrains the position. Additionally, since the tunnel has an arc ceiling and almost identical width and height, the roll angle cannot be effectively constrained by LiDAR measurements. Fortunately, the orientation can be directly measured by the IMU thus roll angle is well-constrained after sensor fusion.

It's also worth mentioning that the x-axis of the eigenspace is parallel to that of the body frame (see Fig. 4.7). That is because the x-axis in the body frame is actually the degenerated direction. However, that is not necessarily the case for y and z if there is no significant difference in their constraint strength.

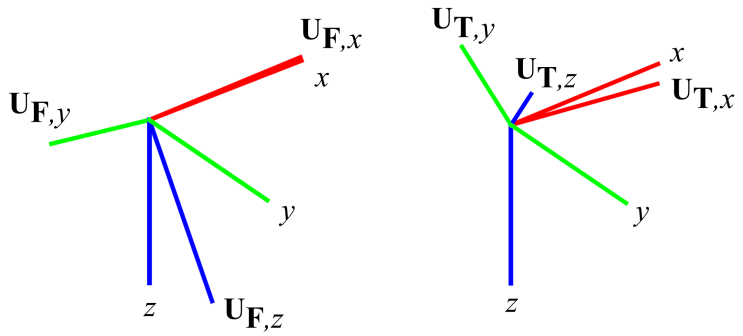


Figure 4.7: An example of the eigenspace obtained from the 1st sampled place (near the left end of the tunnel). x-,y- and z-axis define the robot body frame. *Left*: The eigenspace of $\mathbf{F}\mathbf{F}^T$. *Right*: The eigenspace of $\mathbf{T}\mathbf{T}^T$.

Considering the UWB, we project the force matrix into the fore-mentioned eigenspace, evaluate the localizability, and compare that with the LiDAR (see the bottom plot of Fig. 4.6). It is easy to see that the UWB compensates for the LiDAR localizability along the x-axis, which means fusing the two sensors will make the estimation problem well-constrained. However, we observe a decrease in localizability in x near the anchor. This is a singular point where only position y and z are measured. Theoretically, additional anchors are needed to solve this issue. In practice, the singular point does not cause failure since the time of being under-constrained is short. Once the robot passes this position, the localizability in x increases and the robot can be localized again.

4.5.3 Tunnel Localization Test

The localization test is conducted by manually flying the robot from the map origin to the other end of the tunnel with an average speed of $0.7m/s$. The UWB anchor board is placed a priori

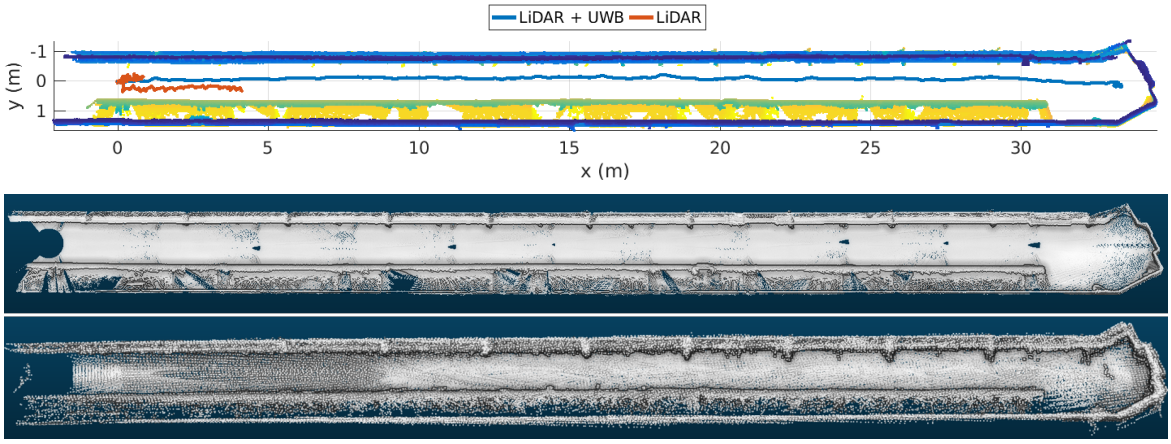


Figure 4.8: *Up*: A comparison of estimated trajectories with/without fusion of the UWB ranging data. *Middle*: The ground truth map is built by matching multiple local scans using the ICP algorithm. *Bottom*: The reconstructed map is assembled by laser scans with estimated poses.

and its location is measured in the pre-built map. In this experiment, we control the usage of the UWB ranging data and compare the localization performance. When the UWB is disabled, the localization starts to drift shortly after the robot takes off. When the UWB ranging data is fused with LiDAR, the robot is able to successfully localize itself throughout the whole flight.

Fig. 4.8 shows the estimated trajectories, the prior map and the reconstructed map. Since the localization accuracy is difficult to measure without a motion capture system, we use the reconstructed map to qualitatively evaluate estimation accuracy. Although the reconstructed map shows larger noise than the prior map, structures on the sidewall are recovered, which indicates the localization is correct.

4.6 Conclusion and Discussion

In this chapter, we present a novel geometric degeneration modeling method that encodes the sensitivity of measurements w.r.t. robot poses. We find a geometric analogy between the frictionless force-closure characterization and our localizability model, which helps to explain its physical meanings. Additionally, it is shown that the LiDAR and the UWB ranging sensor are complementary in terms of localizability and the presented fusion method is demonstrated to allow for robust localization inside real geometrically degenerated tunnels.

Additionally, there are several issues related to this work worth further discussion and potentially open directions for future research:

- Local and global ambiguity. The concept of localizability discussed in this work can be understood as the ambiguity that happens locally around the robot. Therefore, we can apply the idea of perturbation to access the strength of constraints. On the other hand, we realize a different type of ambiguity exists on a global scale. For instance, two identical

rooms may confuse the robot which tries to localize itself in the map. Our concept of localizability does not apply to this type of situation.

- The connection between localizability and uncertainty. It is shown in the experiment that the localizability matches the localization error, thus could be treated as a measure of pose uncertainties. Although the computed localizability may not be suitable for being used as uncertainty directly since it is up to a scale, the analysis still generates rich information for accessing the quality of maps and measurements for localization.
- The analogy of frictionless force-closure is an interesting finding of this work. It implies that the estimation problem in SLAM could have rich physical meanings, which can lead to the development of new understandings and methods.
- The roles played by the data points in estimation problems seem to be separable. Through the localizability analysis in the tunnel experiments, we learned that data points at different places contribute separately to the estimation problem. For example, points on the ground provide strong constraints to the height estimation but weak constraints to other dimensions. This observation brings up an interesting question: if one dimension of an estimation problem is *relatively* over-constrained, can we reduce the number of measurements for sparsity? Or the reversed question: if one dimension is under-constraint, can we increase the importance of data points relevant to that dimension? In the case of a tunnel, this suggests a similar idea as in [51], isolating a small number of critical points to constraint the position along the tunnel.
- Finally, the localizability analysis offers an insight into complementary sensing, suggesting that designing a multi-sensor system could be more efficient if the sensing capacities are well-coordinated. For example, instead of deploying many UWB beacons in the tunnel, we can use only one pair of UWB to specifically constrain the degenerated dimension.

Chapter 5

Joint Optimization for Dense Reconstruction

In previous chapters, we have discussed how to achieve real-time pose estimation using LiDAR and IMU, and proposed the localizability model to predict potential failures in degenerated environments. From this chapter, we switch our focus from online real-time estimation to offline mapping. From a practical perspective, our goal is to build a textured and accurate 3D model that can assist humans for inspection.

The rest of this chapter is organized as follows: Section 5.1 defines the dense reconstruction problem. Section 5.2 reviews prior works related to LiDAR-camera fusion and calibration. Section 5.3 describes the proposed method in detail. Experimental results are shown in Section 5.4. Conclusions and future work are discussed in Section 5.5.

5.1 The Dense Mapping Problem

Our goal is to build accurate dense 3D models by fusing multiple frames of LiDAR and camera data as shown in Fig. 6.9. The LiDAR scans 3D points on the surface of an object and the acquired data are accurate in range and robust to low-texture conditions. However, the LiDAR data contain limited information of texture (only intensities) and are sparse due to the physical spacing between internal lasers. Differently, a camera provides denser texture data but does not measure distances directly. Although a stereo system measures the depth through triangulation, it may fail in regions of low texture or repeated patterns. Those complementary properties make it attractive to fuse LiDAR and cameras for building dense textured 3D models.

The majority of proposed sensor fusion algorithms typically augment the image with LiDAR depth. Then the sparse depth image may be upsampled to get a dense estimation, or used to facilitate the stereo triangulation process. However, we observe two drawbacks of these strategies. The first one is that depth augmentation requires extrinsic sensor calibration, which, compared to the calibration of stereo cameras, is less accurate since matching structural and textural fea-

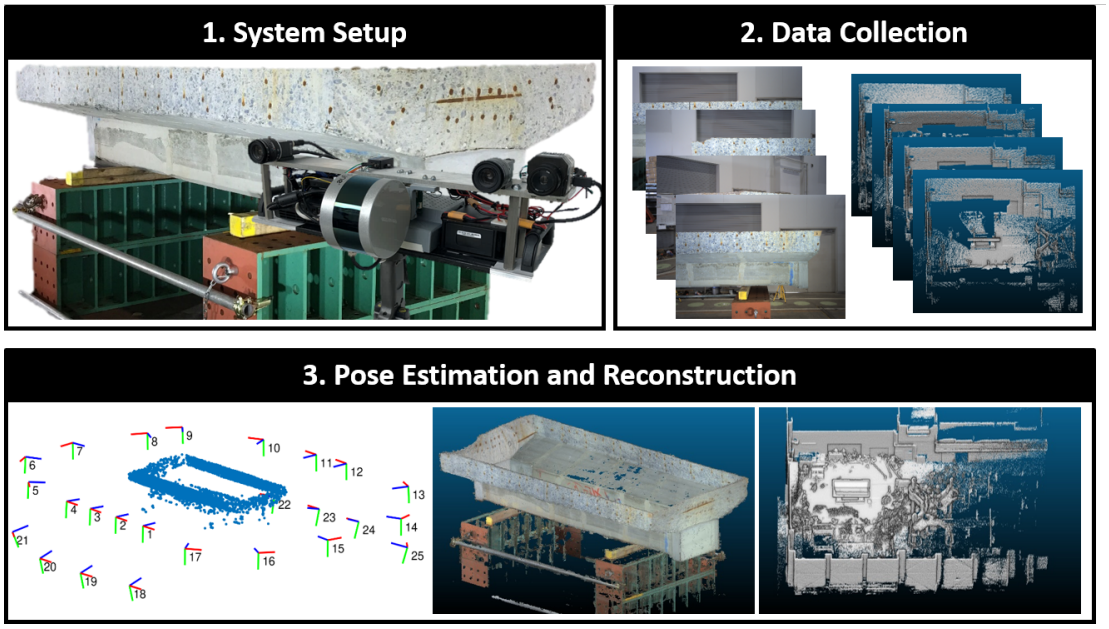


Figure 5.1: A customized LiDAR-stereo system is used to collect stereo images (only left images are visualized) and LiDAR point clouds. Our algorithm estimates the camera poses, generates a textured dense 3D model of the scanned specimen and a point cloud map of the environment.

tures can be unreliable. The second drawback is that the upsampling or LiDAR-guided stereo triangulation techniques assume the geometry is smooth locally, which becomes invalid if the actual depth is too sparse. The accuracy of fused depth map is hence decreased, which may still be useful for obstacle avoidance, but not ideal for the purpose of mapping.

Additionally, although LiDAR-camera fusion techniques have also been used for state estimation, the resulting pipelines are usually designed to separately process the raw sensing data such that the information is abstracted to reduce the computational cost. However, one drawback of this strategy is the compounding errors accumulated and propagated through different components of the pipeline. For instance, biased motion constraints due to false feature matches are passed to the back-end of SLAM and could be difficult to remove. Therefore, instead of separate components, a joint or batch strategy might make more sense in cases where efficiency can be traded off for improved accuracy.

Based on the above discussion, our problem in this chapter can be stated as follows:

> Given LiDAR and camera measurements, design algorithms that consistently combine different sensing capacities to produce dense accurate 3D maps.

In this work, we choose to use a rotating LiDAR with a wide-baseline, high-resolution stereo system to increase the density of raw data. Moreover, we aim to recover the extrinsic calibration simultaneously.

5.2 Related Work

LiDAR-camera fusion: For the purpose of motion estimation and mapping, available fusion algorithms are mostly designed for LiDAR-monocular or LiDAR-stereo systems and assume the extrinsic transform is known. For a LiDAR-monocular system, images are often augmented with the projected LiDAR depth. The fused data can then be used for multiple tasks. For example, Dolson et al. [40] upsample the range data for the purpose of safe navigation in dynamic environments. Bok et al. [18] and Vechersky et al. [147] colorize the range data using camera textures. Zhang and Singh [154] show significant improvement in the robustness and accuracy of the visual odometry if enhanced with depth. For LiDAR-stereo systems [71][95][92][32], LiDAR is typically used to guide the stereo matching algorithms since a depth prior could significantly reduce the disparity searching range and help to reject outliers. For instance, Miksik et al. [95] interpolate between LiDAR points to get a depth prior before stereo matching. Maddern and Newman [92] propose a probabilistic framework that encodes the LiDAR depth as prior knowledge and achieves real-time performance. Additionally, in the area of surveying [101][104][1], point clouds are registered based on the motion estimated using cameras. Our work in this chapter differs from these works in that the LiDAR data and camera data are not fused directly to create the 3D map. Instead, we use LiDAR data to refine the stereo reconstruction after the calibration is recovered.

LiDAR-camera calibration: For extrinsic calibration, available methods can be roughly categorized according to the usage of a target. For example, a single [161] or multiple [50] chessboards can be used as planar features to be matched between the images and point clouds. Besides, people also use specialized targets, such as a box [116], a board with shaped holes [149] or a trihedron [54], where the extracted features also include corners and edges. The usage of a target simplifies the problem but is inconvenient when a target is not available. Therefore target-free methods are developed using natural features (e.g. edges) which are usually rich in the environment. For example, Levinson and Thrun [86] make use of the discontinuities of LiDAR and camera data, and refine the initial guess through a sampling-based method. This method is successfully applied to a self-driving car to track the calibration drift. Pandey et al. [109] develop a Mutual Information (MI) based framework that considers the discontinuities of LiDAR intensities. However, the performance of this method is dependent on the quality of intensity data, which might be unreliable without calibration. Differently, [72][126][21] recover the extrinsic transform based on the ego-motion of individual sensors. These methods are closely related to the well-known hand-eye calibration problem [143] and do not rely on feature matching. However, the motion estimation and extrinsic calibration are solved separately and the sensor uncertainties are not considered. Instead, we try to construct a cost function that joins the two problems in a probabilistically consistent way and optimizes all parameters together.

5.3 Joint Estimation and Mapping

In this section, we elaborate on the technical details of the proposed joint optimization framework for fusing raw measurements from LiDAR and camera.

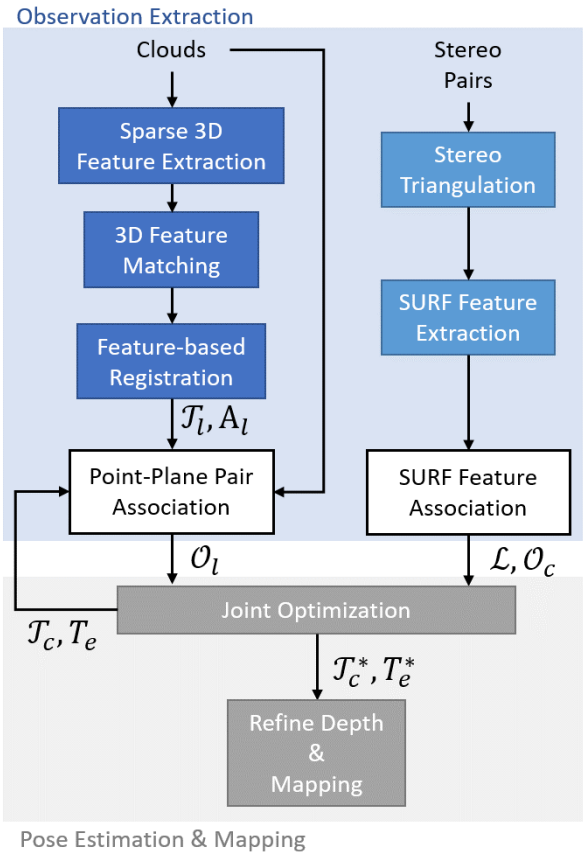


Figure 5.2: A diagram of the proposed pipeline. In the observation extraction phase (front-end), SURF features are extracted and matched across all datasets to build the landmark set $\mathcal{L}$ and the camera observations $\mathcal{O}_c$. On the other hand, point clouds are abstracted with BSC features, and roughly registered to find cloud transforms $\mathcal{T}_l$. Then point-plane pairs are found to build the LiDAR observation set $\mathcal{O}_l$. In the pose estimation and mapping phase (back-end), we solve the BA problem and the cloud registration problem simultaneously. Here the $\mathcal{O}_l$ is recomputed after each convergence based on the latest estimation $\mathcal{T}_c, \mathbf{T}_e$ and the optimization is repeated for a few iterations. Finally, local stereo reconstructions are refined using LiDAR data and assembled to build the 3D model.

5.3.1 Approach Overview

Before introducing the proposed pipeline, we clarify the definitions used in this chapter. An *image landmark* $\mathrm{l} \in \mathbb{R}^3$ is defined as a 3D point that is observed in at least two images. Then a *camera observation* is represented by a 5-tuple $\mathbf{o}_c = \{i, k, \mathbf{u}, d, w\}$, where the elements are the camera id, the landmark id, image coordinates, the depth and a weight factor of the landmark, respectively. In addition, a *LiDAR observation* is defined as a 6-tuple $\mathbf{o}_l = \{i, j, \mathbf{p}, \mathbf{q}, \mathbf{n}, w\}$ that contains the target cloud id i, the source cloud id j, a key point in the source cloud, its nearest neighbor in the target cloud, the neighbor's normal vector and a weight factor. In other words,

one LiDAR observation associates a 3D point to a local plane and the point-to-plane distance will be minimized in the later joint optimization step.

The complete pipeline of proposed method is shown in Fig. 5.2. Given the stereo images and LiDAR point clouds, we first extract and match features to prepare three sets of observations, namely the landmark set $\mathcal{L}$, the camera observation set $\mathcal{O}_c$ and the LiDAR observation set $\mathcal{O}_l$. The observations are then fed to the joint optimization block to estimate optimal camera poses $\mathcal{T}_c^*$ and sensor extrinsic transform $\mathbf{T}_e^*$. Based on the latest estimation, the LiDAR observations are recomputed and the optimization is repeated. After a number of iterations, the parameters converge to local optima. Finally, the refinement and mapping block joins the depth information from stereo images and LiDAR clouds to produce the 3D model. In the rest of this section, each component is described in detail individually.

5.3.2 Camera Observation Extraction

Given a stereo image pair, we firstly perform stereo triangulation to obtain a disparity image using Semi-Global Matching (SGM) proposed in [66]. The disparity image is represented in the left camera frame. Then SURF [12] features are extracted from the left image. Note that our algorithm itself does not require a particular type of feature to work. After that, a feature point is associated with depth value if a valid disparity value is found within a small radius (2 pixels in our implementation). Only the key points with depth are retained for further computation. The steps above are repeated for all stations to acquire multiple sets of features with depth. Once the depth association is done, a global feature association block is used to find correlations between all possible combinations of images. We adopt a simple matching method that incrementally adds new observations and landmarks to $\mathcal{O}_c$ and $\mathcal{L}$. Algorithm 1 shows the detailed procedures. Basically, we iterate through all possible combinations to match image features based on the Euclidean distance of corresponding descriptors. $\mathcal{L}$ and $\mathcal{O}_c$ will be updated accordingly if a valid match is found.

Additionally, an adjacency matrix $\mathbf{A}_c$ encoding the correlation of the images can be obtained. Since the camera FOV is narrow, it is likely that the camera pose graph is not fully connected. Therefore, additional connections have to be added to the graph, which is one of the benefits of fusing point clouds.

5.3.3 LiDAR Observation Extraction

Although many 3D local surface descriptors have been proposed (a review is given in [60]), they are less stable and not accurate compared to image feature descriptors. In fact, it is preferable to use 3D descriptors for rough registration and refine the results using slower but more accurate methods such as Iterative Closest Point (ICP) [14]. Our work follows a similar idea. Specifically, the Binary Shape Context (BSC) descriptor [41] is used to match and roughly register point clouds to compute the cloud transforms $\mathcal{T}_l$. As a 3D surface descriptor, BSC encodes the point density and distance statistics on three orthogonal projection plane around a feature

Algorithm 1: SURF Feature Association

Given feature sets $\mathcal{F}_{1:N}$ from N stations;
for $i = 1 : N$ **do**
  for $j = i + 1 : N$ **do**
    for $\mathbf{f}$ *in* $\mathcal{F}_i$ **do**
      find the best match $\mathbf{g}$ in $\mathcal{F}_j$;
      if $\mathbf{f}$ *and* $\mathbf{g}$ *NOT similar* **then**
        continue;
      end
      if $\mathbf{f}, \mathbf{g}$ *both unlabeled* **then**
        create new landmark id $k \leftarrow |\mathcal{L}|$;
        label $\mathbf{f}, \mathbf{g}$ with id k;
        add new landmark $\mathbf{l}$ with id k to $\mathcal{L}$;
        add new observations $\mathbf{o_f}, \mathbf{o_g}^1$ to $\mathcal{O}_c$;
      else if $\mathbf{f}$ *labeled,* $\mathbf{g}$ *unlabeled* **then**
        copy label from $\mathbf{f}$ to $\mathbf{g}$;
        add new observation $\mathbf{o_g}$ to $\mathcal{O}_c$;
      else if $\mathbf{f}$ *unlabeled,* $\mathbf{g}$ *labeled* **then**
        copy label from $\mathbf{g}$ to $\mathbf{f}$;
        add new observation $\mathbf{o_f}$ to $\mathcal{O}_c$;
      else
        continue;
      end
    end
  end
end
return $\mathcal{O}_c, \mathcal{L}$.

point. Furthermore, it represents the local geometry as a binary string which enables fast difference comparison on modern CPUs. Fig. 5.3-left shows an example of extracted BSC features. However, feature-based registration is of low accuracy. As shown in the right plots of Fig. 5.3, misalignment can be observed in the rough registered map. As a comparison, the refined map of higher accuracy obtained by our method is also visualized.

After the rough registration, another adjacency matrix $\mathbf{A}_l$ encoding matched cloud pairs is obtained. We use the merged adjacency matrix $\mathbf{A}_c \vee \mathbf{A}_l$ to define the final pose graph, where $\vee$ means element-wise or logic operation.

To obtain $\mathcal{O}_l$, a set of points are sampled randomly from each point cloud as the key points. Note that the key points to refine the registration are denser than the features. For each pair of connected clouds in $\mathbf{A}_l$, the one with a smaller index is defined as the target while the other one as the source. Then each key point in the source is associated with its nearest neighbor and a local normal vector in the target within a given distance threshold. Finally, all point matches are

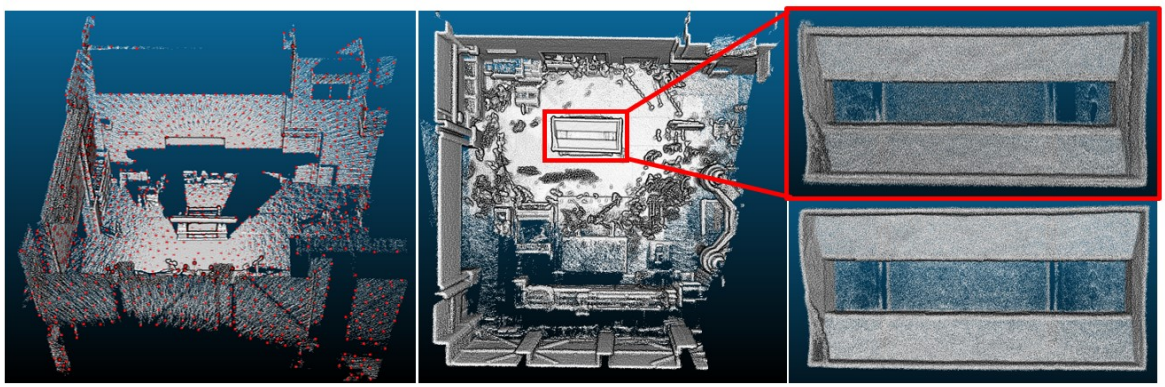


Figure 5.3: *Left:* An example of extracted BSC features (red) from a point cloud (grey). *Middle:* Registered point cloud map based on matched features. *Right:* Comparison of rough registration (top-right) and refined registration (bottom-right) in a zoomed-in window.

formatted as a LiDAR observation and stacked into $\mathcal{O}_l$.

5.3.4 Joint Optimization

Given the observations $\mathcal{O}_c$ and $\mathcal{O}_l$, we first formulate the observation likelihood as the product of two probabilities

$$P(\mathcal{O}_c, \mathcal{O}_l | \mathcal{T}, \mathcal{L}, \mathbf{T}_e) = P(\mathcal{O}_c | \mathcal{T}, \mathcal{L}) P(\mathcal{O}_l | \mathcal{T}, \mathbf{T}_e) \tag{5.1}$$

where $\mathcal{T} = \{\mathbf{T}_i | i = 1, 2, \cdots\}$ is the set of camera poses with $\mathbf{T}_1 = \mathbf{I}_4$, and $\mathbf{T}_e$ is the extrinsic transform. Assuming the observations are conditionally independent, we have

$$P(\mathcal{O}_c | \mathcal{T}, \mathcal{L}) = \prod_{\mathbf{o}_c \in \mathcal{O}_c} P(\mathbf{o}_c | \mathbf{T}_i, \mathbf{l}_k) \tag{5.2}$$

$$P(\mathcal{O}_l | \mathcal{T}, \mathbf{T}_e) = \prod_{\mathbf{o}_l \in \mathcal{O}_l} P(\mathbf{o}_l | \mathbf{T}_i, \mathbf{T}_j, \mathbf{T}_e) \tag{5.3}$$

where i, j are camera ids and k is the landmark id, which are specified by observation $\mathbf{o}_c$ or $\mathbf{o}_l$. The probability of one observation is approximated with a Gaussian distribution as

$$P(\mathbf{o}_c | \mathbf{T}_i, \mathbf{l}_k) \propto \exp\left(-\frac{1}{2} w_{\mathbf{o}_c} (E_f^2 + E_d^2)\right) \tag{5.4}$$

$$P(\mathbf{o}_l | \mathbf{T}_i, \mathbf{T}j, \mathbf{T}_e) \propto \exp\left(-\frac{1}{2} w_{\mathbf{o}_l} E_l^2\right) \tag{5.5}$$

where $w_{\mathbf{o}_c}, w_{\mathbf{o}_l}$ are the weighting factors of camera and LiDAR observations. And the residual E_f and E_d encode landmark reprojection and depth error, while E_l denotes the point-to-plane

distance error. Those residuals are defined as

$$\text{feature: } E_f = \frac{||\phi(\mathbf{l}_k|\mathbf{K}, \mathbf{T}_i) - \mathbf{u}||}{\sigma_p} \tag{5.6}$$

$$\text{depth: } E_d = \frac{\|\psi(\mathbf{l}_k|\mathbf{T}_i)\| - d}{\sigma_d} \tag{5.7}$$

$$\text{laser: } E_l = \frac{\mathbf{n}^{\mathrm{T}}\left(\psi(\mathbf{p}|\mathbf{T}_{l,ij}) - \mathbf{q}\right)}{\sigma_l} \tag{5.8}$$

Here, $\mathbf{u}$ and d are observed image coordinates and depth of landmark k. $\mathbf{T}_{l,ij} = (\mathbf{T}_e\mathbf{T}_i)^{-1}\mathbf{T}_j\mathbf{T}_e$ is the transform from target cloud i to source cloud j. Function $\phi(\cdot)$ projects a landmark onto the image i specified by input intrinsic matrix $\mathbf{K}$ and transform $\mathbf{T}_i$. Function $\psi(\cdot)$ transforms a 3D point using the input transformation. σ_p, σ_d and σ_l denote the measurement uncertainties of extracted features, stereo depths and LiDAR ranges, respectively.

Substituting (5.2)-(5.8) back into (5.1) and taking the negative log-likelihood gives the cost function

$$f(\mathcal{T}, \mathcal{L}, \mathbf{T}_e) = \frac{1}{2}\sum_{\mathbf{o}_c} w_{\mathbf{o}_c}\left(E_f^2 + E_d^2\right) + \frac{1}{2}\sum_{\mathbf{o}_l} w_{\mathbf{o}_l} E_l^2 \tag{5.9}$$

which is iteratively solved over parameters $\mathcal{T}, \mathcal{L}, \mathbf{T}_e$ using the Levenberg-Marquardt algorithm.

To filter out incorrect observations in both images and point clouds, we check the reprojection error $||\phi(\mathbf{l}_k|\mathbf{K}, \mathbf{T}_i) - \mathbf{u}||$ and depth error $\|\psi(\mathbf{l}_k|\mathbf{T}_i)\| - d$ of camera observations and check the distance error $\mathbf{n}^{\mathrm{T}}\left(\psi(\mathbf{p}|\mathbf{T}_{l,ij}) - \mathbf{q}\right)$ of LiDAR observations after the optimization converges. The observations whose errors are larger than prespecified thresholds will be marked as outliers and assigned with zero weights. The cost function (5.9) is optimized repeatedly until no more outliers can be detected. The thresholds can be tuned by hand and in the experiments we use 3 pixels, 0.01m and 0.1m respectively.

Similar to the ICP algorithm, the $\mathcal{O}_l$ is recomputed based on the latest estimation of $\mathcal{T}_c, \mathbf{T}_e$, while the $\mathcal{O}_c$ remains unchanged. Once $\mathcal{O}_l$ is updated, the outlier detection and optimization steps are repeated as mentioned above. The $\mathcal{O}_l$ only needs to be recomputed a few times (4 times in our experiments) to achieve good accuracy.

Additionally, the strategy of specifying the uncertainty parameters is as follows. Based on the stereo configuration, the triangulation depth error e_d is related to the stereo matching error e_p by a scale factor as in $e_d = (d^2/bf)e_p$, where b is the baseline, f is the focal length and d is the depth. Assuming the uncertainties of feature matching and stereo matching are equivalent, we have $\sigma_d = (d^2/bf)\sigma_p$. Therefore, we can now set σ_p to be the identity (i.e. 1) and set σ_d by multiplying the scale factor. On the other hand, the value of σ_l is tuned by hand so that the total cost of camera and LiDAR observations are roughly at the same magnitude. In the experiments, setting $\sigma_p = 1$, $\sigma_d = 5 \times 10^3$ and σ_l between $[0.02, 0.1]$ can generate sufficiently good results.

5.3.5 Fused Mapping

With the camera poses estimated, building a final 3D model could be simply registering all stereo

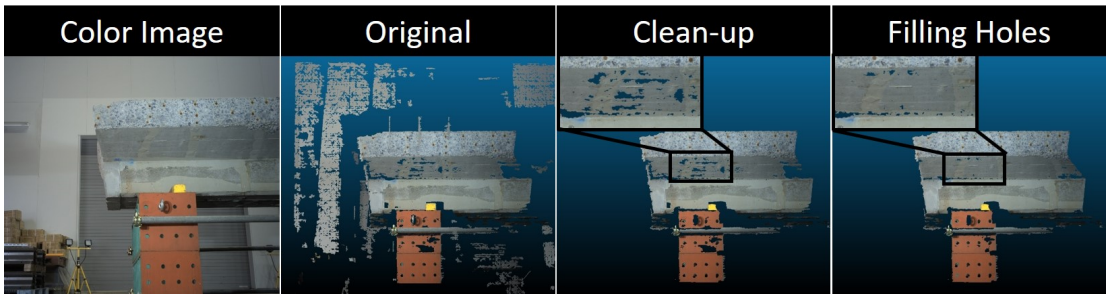


Figure 5.4: An example of refining the stereo depth. The outliers are first filtered out by limiting its difference to the LiDAR depth within a maximum range threshold. Then the holes are filled with the surrounding LiDAR depth only if the local surface has a near-zero curvature.

point clouds together. However, the stereo depth maps typically contain outliers and holes due to triangulation failure. In order to refine the stereo depth maps, we further perform a simple but effective two-fold fusion of LiDAR and camera data for each frame or station. In the first fold, the stereo depth is compared with the projected LiDAR depth and will be removed if there is a significant difference. In the second fold, LiDAR depth is selectively used to fill holes in the stereo depth. Particularly, we only use the regions that are locally flat (such that the local smoothness assumption is valid), and well observed (avoiding degenerated view angle). The curvature of the local surface is used to measure the flatness. And the normal vector is used to compute the view angle. Fig. 5.4 shows an example of refining the stereo point cloud. It can be observed that holes lying on a flat surface can be filled successfully, while the missing points close to the edges are not treated to avoid introducing new outliers.

5.3.6 Conditions of Uniqueness

The proposed approach relies on the ego-motion of individual sensors to recover the extrinsic transform $\mathbf{T}_e$, making it possible that $\mathbf{T}_e$ is not fully observable if the motion degenerates. It turns out to be the same problem encountered in hand-eye calibration, where the extrinsic transform between a gripper and a camera is estimated from two motion sequences. Here we discuss conditions for a fully observable $\mathbf{T}_e$ by borrowing knowledge from the hand-eye calibration, whose classical formulation is given by

$$\mathbf{T}_c\mathbf{T}_e = \mathbf{T}_e\mathbf{T}_h \tag{5.10}$$

where $\mathbf{T}_h, \mathbf{T}_c$ represent the relative motion of the hand and the camera w.r.t. their own original frames. Incorporating multiple stations will result in a set of (5.10) and then $\mathbf{T}_e$ can be solved. According to [143], the following two conditions must be satisfied to guarantee a unique solution of $\mathbf{T}_e$:

1. At least 2 motion pairs $(\mathbf{T}_c, \mathbf{T}_h)$ are observed. Equivalently, at least 3 stations are needed, with one of them to be the base station.
2. The rotation axes of $\mathbf{T}_c$ are not colinear for different motion pairs.

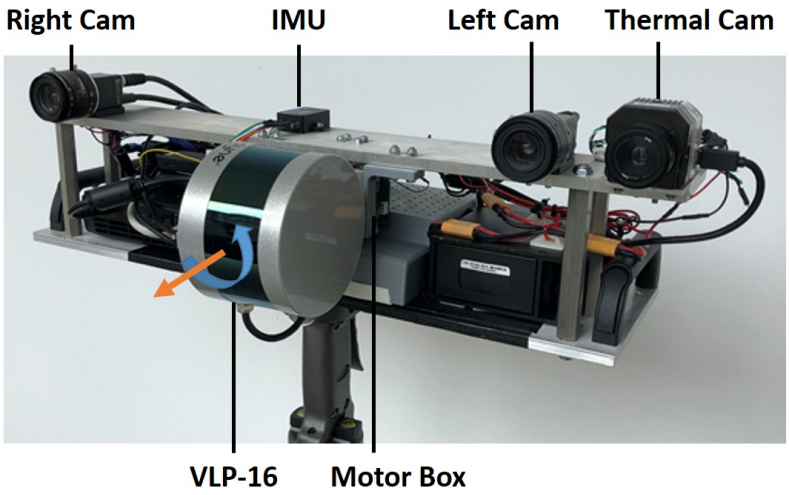


Figure 5.5: The sensor pod developed for data collection.

In our case, the robot hand frame is substituted by the LiDAR frame. Therefore, the configuration of each station must also satisfy the above conditions of uniqueness. This provides formal guidance to collect data effectively. From our experience of deploying the developed system, an operator without adequate background knowledge in computer vision, particularly in structure from motion, is likely to miss the second condition and only rotates the sensor about the vertical axis, which will make the extrinsic calibration unobservable.

5.4 Experiments

5.4.1 The Sensor Pod

To collect data for experiments, we developed a sensor pod (as shown in Fig. 5.5) which has a pair of stereo cameras (global shutter, resolution 4112×3008, baseline 38cm), a Velodyne Puck (VLP-16), an IMU and a thermal camera. This work only uses the stereo image pairs and LiDAR clouds for reconstruction. Particularly, the VLP-16 is mounted on a continuously rotating (180° per second) motor to increase the sensor FOV.

The calibration between the involved sensors are performed separately. We use the OpenCV library [20] to obtain camera intrinsic and extrinsic parameters. The transform between the motor and the LiDAR frame is obtained by placing the sensor pod in a conference room, and carefully tuning the transform until the accumulated points on walls and ceiling form thin surfaces in the fixed motor base frame. From now on, we use the term *LiDAR frame* to denote the fixed motor base frame instead of the actual rotating Velodyne frame, and assume all point clouds have been transformed into the LiDAR frame.

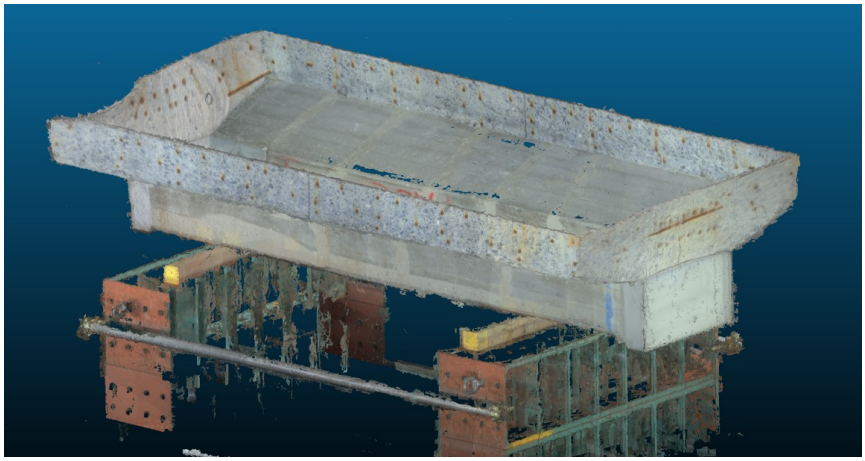


Figure 5.6: Built point cloud model of the T-shaped specimen.

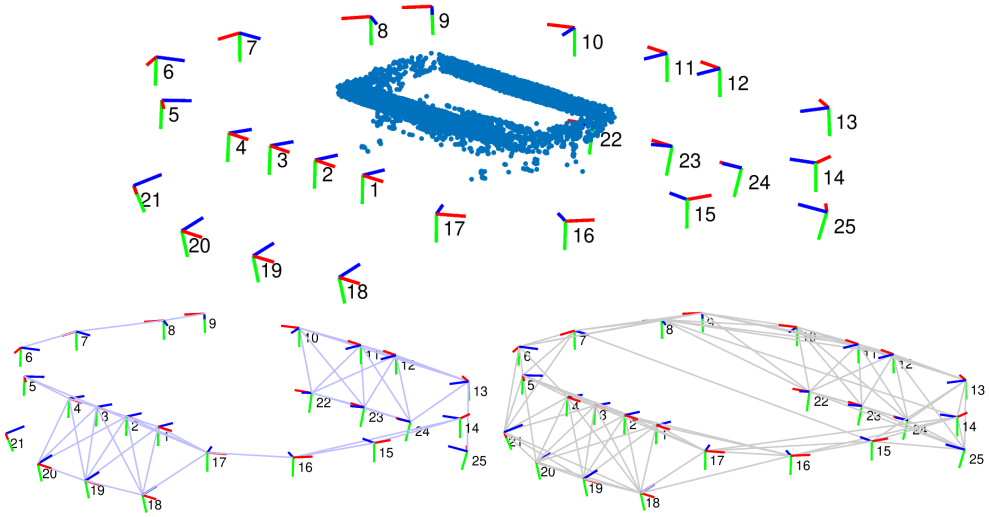


Figure 5.7: *Top:* Estimated camera poses (numbered in the order of capture) and visual landmarks (blue points). We follow the convention to define camera frame z (blue) forward, y (green) downward. *Bottom:* Pose graph connections from images (blue) and poing clouds (gray)

5.4.2 Reconstruction Tests

The first reconstruction test is carried out at the Shimizu Institute of Technology in Tokyo to scan a T-shaped concrete specimen that is under structural tests. In total, 25 stations of data are collected around the specimen at a distance of about 2.5 meters. Each station contains a stereo image pair, a point cloud that accumulates scans for 20 seconds and contains approximately 1.6 million points. For station 1-17, the sensor pod is placed on a tripod and pointed to the specimen. Station 18-25 are collected with the sensor pod on the ground, tilted up to capture the bottom of the specimen. Fig. 5.6 shows the reconstructed model and Fig. 5.7 visualizes the camera poses and landmarks. In the lower plots of Fig. 5.7, correlations found between images (blue lines) and point clouds (grey lines) are visualized. Since the cameras have narrow FOV (48° horizontal), it

Figure 5.8: From top to bottom, the results of three tests are visualized: a squared pillar (top), a cylinder pillar (middle) and a bridge pillar (bottom). From left to right, we visualize the camera poses and landmarks (blue points), a sample of the image data, complete LiDAR point cloud, overlaid LiDAR and stereo point cloud, dense stereo point cloud.

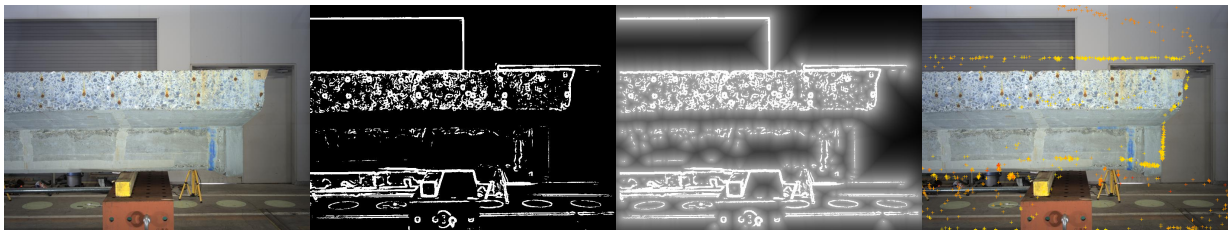
Table 5.1: Dataset and Model Statistics

Datasets	Stations (Frames)	# of LiDAR points ($\times 10^6$)	# of stereo points ($\times 10^6$)	Error (mm)
T-shaped	25	32.4	78.4	N/A
squared	29	39.1	210.3	2.7
cylinder	54	66.5	111.7	N/A
bridge	32	38.6	168.7	3.9

is likely that adjacent images don't have enough overlap, which makes the pose graph not fully connected. Fortunately, LiDAR clouds have much wider FOV and therefore guarantees a fully connected graph.

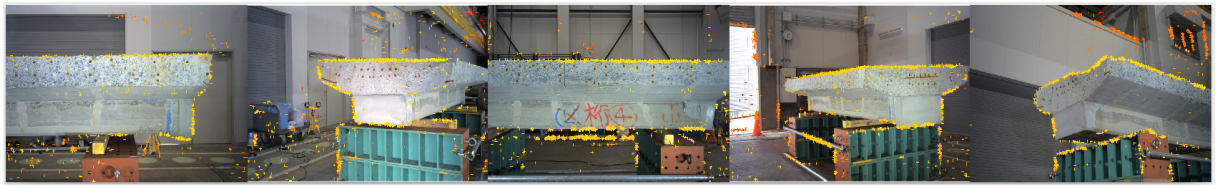
As to the computation statistics, we provide a rough measure of the processing time of the major components. On a standard desktop (i7-3770 CPU, 3.40GHz$\times$8), it takes less than 2min to remove vignetting effects and triangulate a stereo pair (40-50min for the whole dataset). The feature-based cloud registration takes about 15min in total and the joint pose estimation and map refinement can be finished in about 15min and 20min respectively.

In addition to the T-shaped specimen, we tested our algorithm in different environments, where the shapes of reconstructed objects vary from simple squared and cylinder pillars to more complex bridge pillars (see Fig. 5.8). Table 5.1 summarizes the model statistics. The averaged error is obtained by comparing to a ground truth model and more details are provided in Section 5.4.



(a) Raw image (b) Edges (c) Edge score (d) Initialization

(e) Edge alignment after calibration using [86]



(f) Edge alignement after joint optimization

Figure 5.9: (a)-(d) The key steps of [86]. (e)-(f) Comparison of extrinsic calibration results from [86] (e) and ours (f). The color of projected cloud edge points encodes the correlation score: yellow means high while red means low.

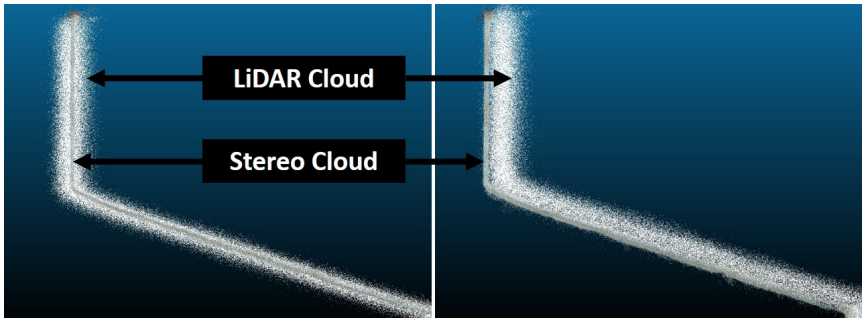


Figure 5.10: Cutaway view of the overlaid LiDAR clouds (white) and stereo clouds (textured). *Left:* Jointly optimized. *Right:* Calibrated using [86].

5.4.3 LiDAR-Camera Calibration

In this section, we evaluate the accuracy of the recovered extrinsic transform. As a comparison, we implemented a target-free calibration method [86] which uses discontinuities in images and point clouds to iteratively refine an initial guess. The key steps of this method are shown in Fig. 5.9a-d. Basically, the initial guess is perturbed in each dimension (x, y, z, roll, pitch, yaw) separately and then moved towards the direction that increases the correlation between image

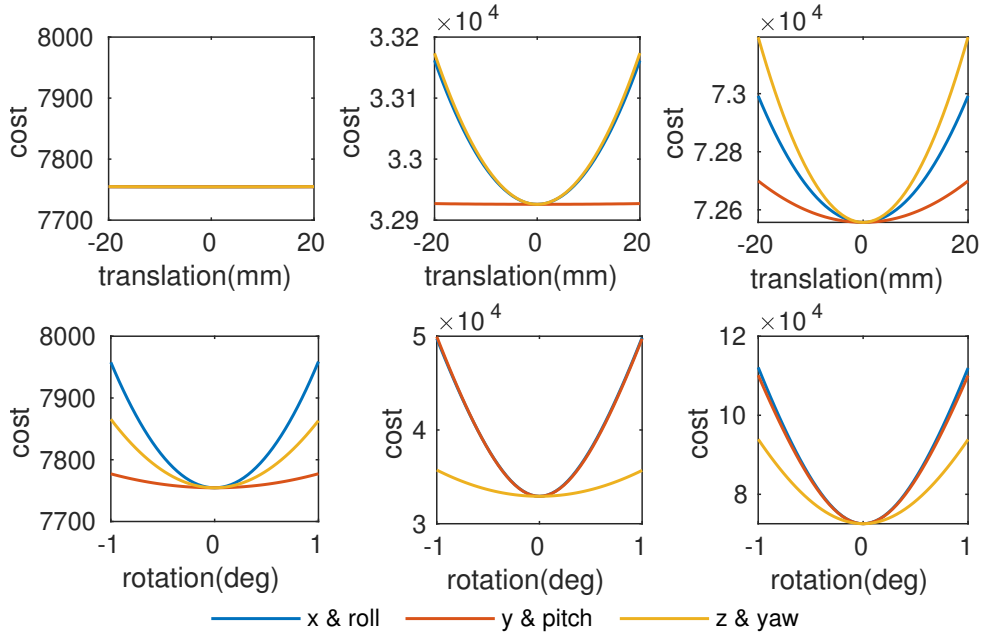


Figure 5.11: Changes of cost values w.r.t. perturbed extrinsic transform. From left to right, the three columns show the cost changes in three tests: with rotation fixed, with rotation about one axis, and with rotation about two axes. Within each test, translation (top plots) and rotation (bottom plots) perturbations are visualized separately.

edges and projected cloud edges. Eventually, a locally optimal solution can be found if any further changes will decrease the edge correlation.

Since it is difficult to get ground truth calibration, we choose to compare the extrinsic parameters computed from two methods. The extracted point cloud edges are projected on to the image plane and the projection is visualized in Fig. 5.9e and 5.9f. However, the edges are both well aligned and no obvious difference can be identified. We then compare the overlay of LiDAR clouds and stereo clouds (see Fig. 5.10). It can be observed that with our results, the models are aligned consistently while there exists an offset if calibrated using [86]. Further investigation shows that the offset happens along the camera's optical axis, in which direction the motion will generate less flow on the image. As a result, the total correlation score becomes less sensitive to the motion of the LiDAR along the optical axis. This observation suggests that calibration methods using direct feature alignment, including target-based and target-free, may require wide angle lenses.

5.4.4 Observability of Extrinsic Transform

The uniqueness conditions stated in Section 5.3 basically requires the sensor pod to change its position and orientation for different stations. In this section, we aim at providing more intuition behind the formal statements. Specifically, the conditions are experimentally demonstrated by perturbing the extrinsic parameters around their optimal values. Three tests are designed to

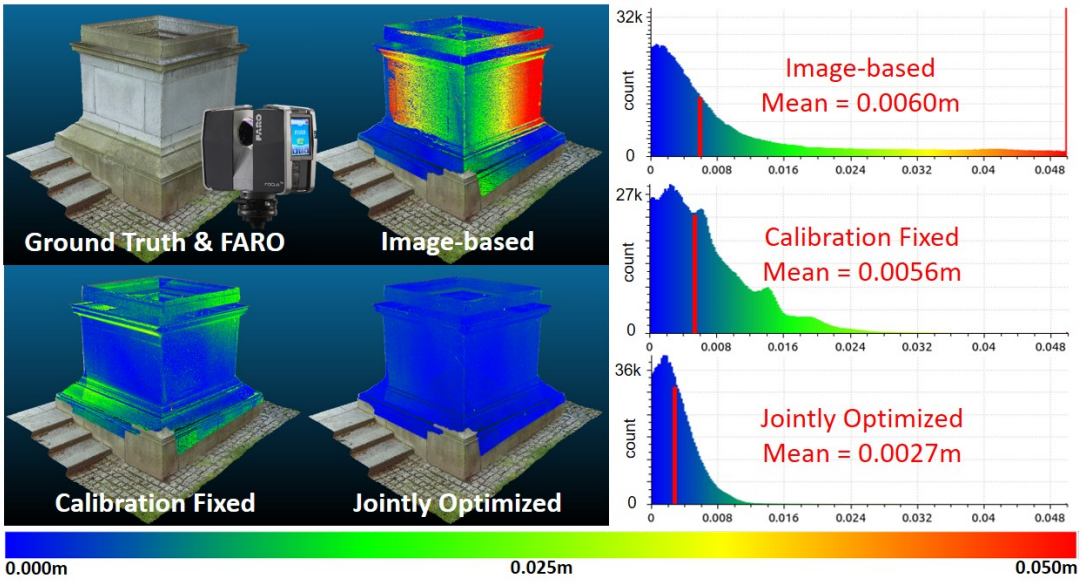


Figure 5.12: Comparing the reconstructed models with the ground truth model built by the FARO scanner. On the left are visualizations of the ground truth model and the distance map of reconstructed models, where the color encodes the distance error between two point clouds. On the right are the distance histograms corresponding to each comparison and the averaged errors are marked by the red vertical bar.

clarify the situations of degeneration.

Rotation is fixed: In this case, the sensor pod is placed at 3 different positions but keeps its orientation unchanged. Specifically, station 1-3 are used for optimization. The total cost after the perturbation is visualized in the left 2 plots of Fig. 5.11. It can be seen that perturbing the translation won't affect the cost value at all, meaning unobservable. Besides, since the 3 frames are almost collinear, the pitch angle is also under-constrained (flat orange curve).

Rotation about one axis: In this case, stations 1-17 are used, where the sensor pod is placed around the T-shaped specimen and all rotations are about the camera's y-axis. As shown in the middle plots of Fig. 5.11, position y is under-constrained.

Rotation about two axes: For reference, we show the perturbed cost with all 25 available datasets in the right plots of Fig. 5.11. In this case, the rotations can be about x- or y-axis. As expected, the extrinsic transform is well constrained.

5.4.5 Model Accuracy Evaluation

Since the ground truth data are not available during the test in Tokyo, we evaluate the reconstruction accuracy on the squared concrete pillar instead. A FARO FOCUS^{3D} scanner (see Fig. 5.12) with $\pm$3mm range precision is used to obtain the ground truth. The comparison is performed by measuring the point to plane distance between the reconstructed model and the ground truth af-

ter precise ICP registration. Furthermore, we compare the results of three models reconstructed using: (1) stereo images only (standard stereo BA), (2) both LiDAR and stereo data but extrinsic calibration is pre-calibrated using [86], and (3) both LiDAR and stereo data with extrinsic calibration being adjusted jointly (proposed in this work). Comparisons (1) and (2) share the same cost function in (3). However, in comparison (1) LiDAR observations are set to have zero weights and $\mathbf{T}_e$ is fixed, and in comparison (2) only $\mathbf{T}_e$ is fixed during optimization.

The error maps and histograms are visualized in Fig. 5.12. It can be observed that fusing LiDAR data helps to reduce the model error from 6mm to 2.7mm, which already lies in the precision range of the ground truth. In fact, due to the limited number of matches between some image frames, the pure image-based model does not align well, resulting in multiple layers of the surface. Compared with the pre-calibrated case, jointly optimizing the calibration improves the overall model accuracy and we also benefit from the convenience of self-calibration. Additionally, since our model is reconstructed from multiple sets of data and each station is collected close to the wall (2-3 meters), it measures about 70 points/cm^2, which is much denser than the ground truth (10-15 points/cm^2). The evaluation results are obtained using the CloudCompare[2] software.

5.5 Conclusion and Discussion

In this work, we present a joint optimization approach to fuse LiDAR and camera for pose estimation and dense reconstruction. It is shown to be able to build dense 3D models and recover camera-LiDAR extrinsic transform accurately. Besides, the accuracy of the reconstructed model is evaluated by comparing it to a ground truth model and it shows our method can achieve accuracy similar to a survey scanner.

This study shows that the joint motion estimation effectively avoids unreliable direct matching between different domains. However, it does not suggest that the LiDAR and camera data should not be correlated in the front end. Therefore, how to further leverage the given LiDAR and camera data in an early fusion fashion becomes an interesting question to explore, which will be the focus of Chapter 6.

Another lesson learned from this work is understanding sensing models from the perspective of observability w.r.t. the unknowns. For example, the discussion of extrinsic calibration unveils the correlation between the calibration accuracy and the amount of optical flow caused by motion along the optical axis. This idea is similar to the concept of localizability in Chapter 4 in that it uses the *strength* of constraints to evaluate whether an unknown variable is well-observed or tightly constrained.

Finally, we recognize that the performance of this work in large-scale reconstruction tasks remains unclear. The reported results support our argument for building locally dense and accurate models, but leave large-scale reconstruction undiscussed. The fact is, to ensure robust performance for larger environments (e.g., a real bridge or tunnel), one needs to consider many new challenges relating to system stability, which will be further discussed in the next chapter.

[2]CloudCompare: `https://www.cloudcompare.org/`

Chapter 6

LiDAR-enhanced Structure from Motion

In Chapter 5, we discussed dense reconstruction through a joint optimization based on raw LiDAR and camera measurements. Although the reconstructed models offer high density and accuracy, the proposed pipeline can be fragile in more complicated real-world scenarios. For example, false visual feature matches could corrupt the final dense model. Therefore, in this chapter, we investigate the techniques to robustify the dense mapping pipeline. More specifically, we switch our thoughts from multi-sensor fusion to Structure from Motion (SfM), and try to cast the dense mapping problem into the SfM framework. One reason is that the techniques developed to improve robustness for large-scale SfM problems are readily applicable in our task. Additionally, the previously presented joint optimization framework could be integrated to SfM conveniently as the final batch optimization step.

The remaining of this chapter is structure as follows: In Section 6.1, we define the problem to be addressed. In Section 6.2, we review the literature on SfM and the combination of LiDAR and cameras for dense reconstruction. Then Section 6.3 describes the proposed pipeline in detail. Experimental results are shown in Section 6.4. Conclusions are discussed in Section 6.5.

6.1 The Dense Reconstruction Problem

As introduced in the previous chapter, high-resolution, narrow Field-Of-View (FOV) cameras are used to capture rich texture details. This poses new challenges to standard SfM algorithms. First of all, most available global or incremental SfM pipelines are based on a single camera, therefore, not recovering scale directly. More importantly, due to the limited FOV, the overlapped area between adjacent images is reduced, which will result in a pose graph that is only locally connected and jeopardize the accuracy of estimated motions. Finally, although there exist methods that adopt SfM to depth sensors, such as RGB-D cameras and structured light devices, they are not quite suitable for large-scale outdoor applications where long-range LiDAR sensors are used. In the specific settings of inspection, we state the dense reconstruction problem as follows:

Given measurements from the LiDAR and stereo cameras, develop robust algorithms to build locally dense and globally consistent maps in large-scale real-world scenarios.

In this work, we propose a novel pipeline that extends the traditional SfM algorithm to work with stereo cameras and LiDAR sensors. This work is based on the simple idea that LiDAR's long-ranging capacity can be used to restrain the relative motions between images. More specifically, we first implemented a stereo SfM pipeline that computes the motions of the camera and estimates the 3D positions of visual features (the structures). Then the LiDAR point clouds and the visual features are joined in a single optimization function, solved iteratively to refine the camera motions and structures. In our pipeline, the LiDAR data enhance the SfM algorithm in two folds: 1) LiDAR point clouds are used to detect and reject invalid image matches, making the stereo SfM pipeline more robust to visual ambiguities; 2) LiDAR point clouds are combined with visual features in a joint optimization framework to reduce the motion drifts. With the two folds of enhancement, our pipeline can achieve more consistent and accurate motion estimation than the state-of-the-art SfM algorithms.

The contribution of this work can be summarized as follows: 1) We adapt the global SfM techniques to a stereo camera system to initialize the camera motions in true scale. 2) LiDAR data are used to verify image matches, further robustifying the pipeline. 3) We extend our previously proposed joint optimization pipeline (in Chapter 5) by considering the shared structures of the stereo camera and LiDAR, which improves the accuracy and consistency of built models.

6.2 Related Work

Structure from Motion: SfM has been an active research area over the past several decades. Early works on self-calibration and reconstruction [96][13][45][36][114] lay the foundation for the development of general purpose SfM tools. One of the earliest open-source tools Bundler, also known as PhotoTourism [133], shows the ability to process a vast amount of online photos to build large-scale 3D models. Since then, significant advancements have been achieved in works such as Theia [137], VisualSFM [150], OpenMVG [100][99] and more recent COLMAP [127]. A comprehensive comparison of available SfM approaches is given in [15] which concludes that COLMAP achieved state-of-the-art accuracy, robustness and completeness. Although many approaches exist, the majority are targeted on the general-purpose reconstruction, which does not take the special conditions of inspection into consideration. For instance, an inspection camera may use long focal length and narrow FOV lens, limiting the overlap between images. This will cause the camera pose graph only locally connected and thus is more likely subject to drifts.

Stereo State Estimation: Stereo pair is more popular in the robotics community for robot state estimation, i.e. stereo visual odometry (stereo-VO). As one of the most fundamental components of robot autonomy, stereo-VO algorithms typically adopt optimization-based approaches such as [90] [85] [145], or filter-based approaches as in [112] and [136]. In addition to pure vision-based methods, people have also been fusing the depth information from LiDAR for robust and

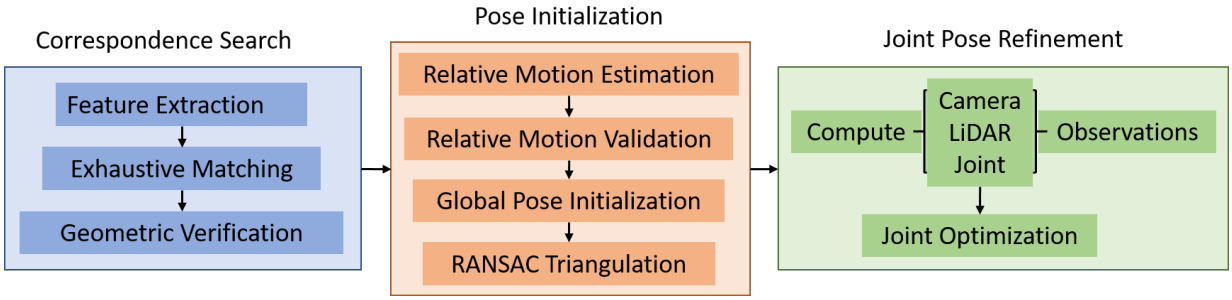


Figure 6.1: The LiDAR-enhanced stereo SfM pipeline.

low-drift state estimation. A monocular camera combined with the LiDAR, as in [154] [55] [132], is demonstrated to be able to provide real-time state estimation with true scale. In [10] and [131], a stereo pair is used to robustify the state estimation. These applications typically use wide-angle cameras and process incoming visual data incrementally for real-time performance. Differently, our work focuses on achieving higher map accuracy through batch optimization and higher robustness by LiDAR-aided visual outlier rejection.

6.3 LiDAR-enhanced SfM

The proposed pipeline takes a set of stereo image pairs and associated LiDAR point clouds as inputs and generates 3D models of the covered environment in the format of triangulated visual points and a merged LiDAR point cloud. Fig. 6.1 shows the procedures of our LiDAR-enhanced SfM pipeline, which is described in detail in this section.

6.3.1 Correspondence Search

Given the stereo image pairs, computing the correspondences includes feature extraction, matching and geometric verification. Firstly, we rely on the OpenMVG library to extract SIFT [89] features from images. Then the features are matched exhaustively using the provided cascade hashing method [31]. Finally, The found matches between two images are verified by checking the two-view epipolar geometry. Specifically, the fundamental matrix $\mathbf{F}$ is estimated using RANSAC and then used to check the epipolar error of matched features. Only geometrically consistent features are kept for further computation.

6.3.2 Relative Motion Estimation

Since the stereo pair is pre-calibrated, we treat a pair of left and right images as an independent unit, called a station. The pose of a station is defined as the pose of the left camera. To estimate the relative motion, a standard stereo method relies on feature points that are observed by all the four images in two stations, while we observe that many points are only shared by three

or even two images. Ignoring those points could lose important information to estimate the camera motion, especially in the case that images have limited overlap. Therefore, we choose to explicitly handle different cases of shared views between two stations.

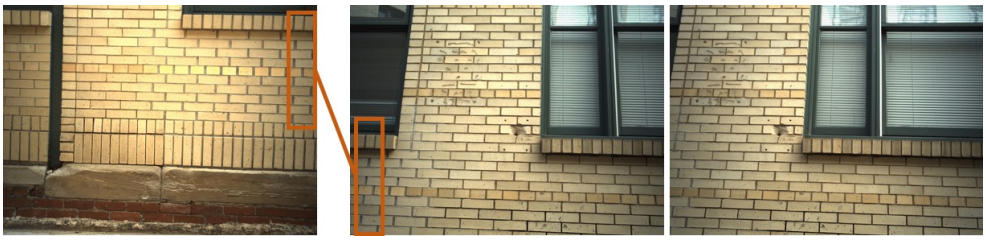


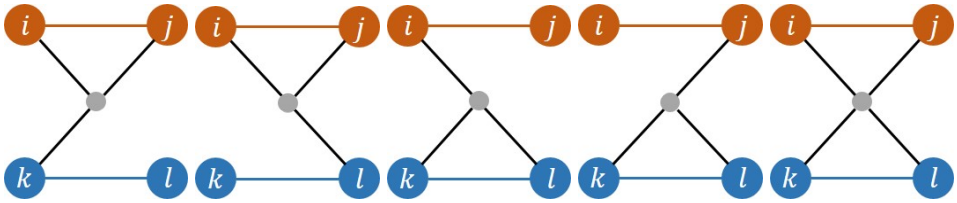
Figure 6.2: An example of regions for 2-view features. *Left:* The right image of one station; *Middle-Right:* The left and right images of the neighbor station. The shared small regions are close to the boundries and marked by red boxes.

Specifically, we consider feature points that are shared by at least 3 views to ensure metric reconstruction. Although points with only 2 views could help to estimate the rotation and the direction of translation, they are ignored here since those points typically come from small overlapped regions as shown in Fig. 6.2. On the other hand, for 3-view or 4-view features, potential cases are shown in Fig. 6.3. It is also possible that multiple types of shared features exist between two stations. To simplify the problem, we choose the type with most correspondences to solve the relative motion. In the case of 3-view, the points are first triangulated with the stereo pair and then the RANSAC+P3P algorithm is used to solve the transform. In the case of 4-view, we follow the standard treatment that first triangulates the points in the two stations and then applies RANSAC+PCA registration algorithm to find the relative motion. In both cases, a non-linear optimization procedure is used to refine the computed poses and triangulations by minimizing the reprojection error of inliers. Finally, all poses are transformed to represent the relative motion between left cameras.

6.3.3 Relative Motion Validation

Once the relative motions are found, a pose graph can be built with the nodes representing the station poses and edges representing the relative motions. The global poses can then be solved by averaging the relative motions on the pose graph. However, it is likely that invalid edges exist due to visual ambiguities in the environment (see Fig. 6.4a and 6.4b) and directly averaging relative motions could give incorrect global poses. Therefore, a two-step edge validation scheme is designed to remove outliers.

In the first step, we check the overlap of LiDAR point clouds for all station pairs and reject inconsistent ones. Specifically, two occupancy grids, namely the source grid $\mathcal{G}_s$ and target grid $\mathcal{G}_t$, are constructed from the corresponding point clouds. Each cell in the grids takes the value 1 if occupied, 0 if free, and -1 if unknown. Then for each occupied cell in $\mathcal{G}_s$, we transform the its center $\mathbf{c} \in \mathbb{R}^3$ to the frame of $\mathcal{G}_t$ using the estimated relative motion $(\mathbf{R}, \mathbf{t}) \in (SO(3), \mathbb{R}^3)$, and



(a) *i-j-k* (b) *i-j-l* (c) *i-k-l* (d) *j-k-l* (e) *i-j-k-l*

Figure 6.3: Examples of shared features (grey dot) between two stations (red and blue circle pairs). The colored bar indicates known stereo calibration. (a)-(d) 3-view; (e) 4-view.

check the cell state at that location. In order to quantify the consistency, we define a consistency ratio as

$$r_c = \frac{\sum_{\mathbf{c}\in\{\mathcal{G}_s(\mathbf{c})=1\}} I\left(\mathcal{G}_t(\mathbf{Rc}+\mathbf{t})=1\right)}{\sum_{\mathbf{c}\in\{\mathcal{G}_s(\mathbf{c})=1\}} I\left(\mathcal{G}_t(\mathbf{Rc}+\mathbf{t})\neq -1\right)} \tag{6.1}$$

where $I(\cdot)$ is the indicator function that equals 1 if the argument is true, or 0 otherwise. The consistency ratio measures how much the $\mathcal{G}_t$ agrees with $\mathcal{G}_s$. In addition, we perform cross-checking by switching the target and source grids and recompute the consistency ratio. A relative motion is treated as valid only if both the ratios are larger than 0.6 (see Fig. 6.4c for an example of inconsistent grids). Note this is a loose threshold since we encourage more pairs to remain for completeness and rely on further checks to reject remaining outliers. Fig. 6.5a shows a 2D example of checking the consistency of two occupancy grids. In the implementation, the Octomap library [68] is used to efficiently iterate through occupied cells and look up the occupancy at a query location.

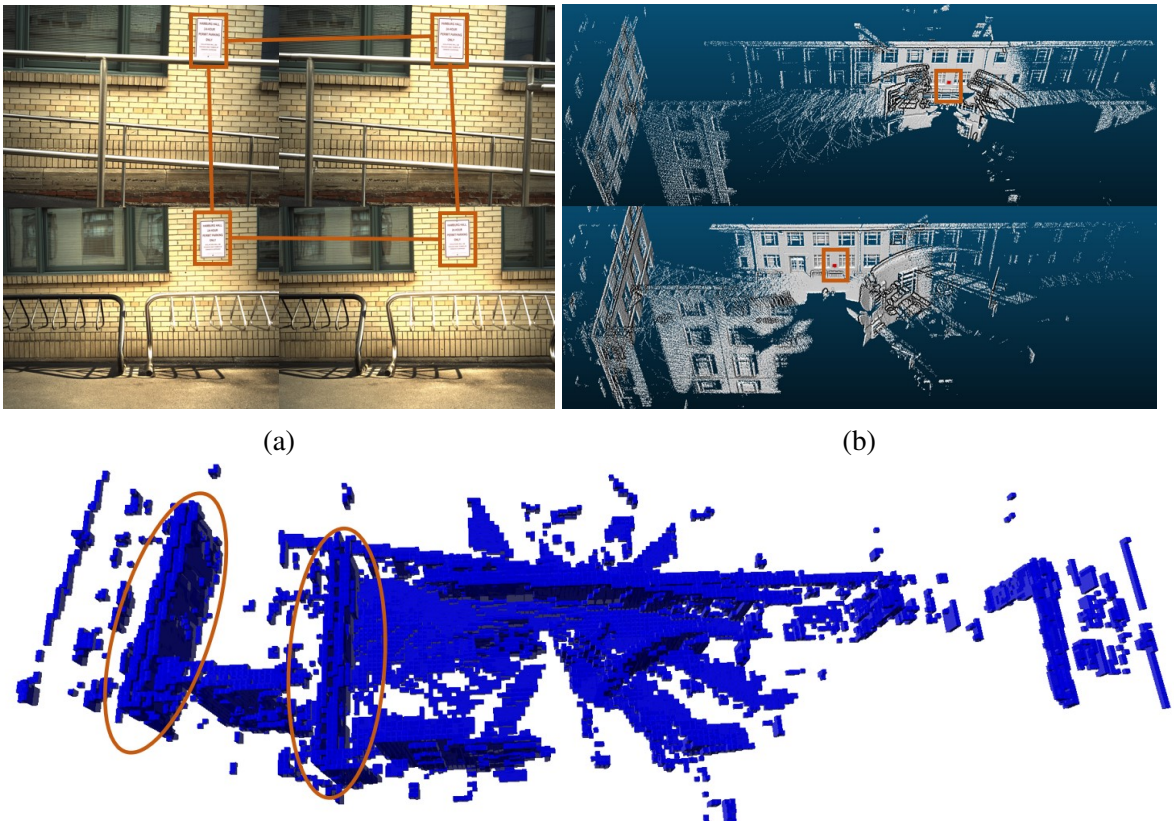
As the second step of validation, we check the cycle consistency as in [98]. Specifically, we check small cycles of length 3, namely triplets. Here small cycles are preferable since they are easy to find and check. And more importantly, small cycles capture local information hence will not be affected by accumulated drifts. Note that the triplet here is a collection of 3 stations instead of images. As shown in Fig. 6.5b, we composite the relative motions $\mathbf{T}_{ki}\mathbf{T}_{jk}\mathbf{T}_{ij} \in SE(3)$ and check the resulting angle and translation. 2° and 0.1m are used as thresholds to label a triplet check as pass or not. For each relative motion, we keep track of its success rate r_s defined as

$$r_s = \frac{\text{\# of passed checks}}{\text{\# of involved checks}} \tag{6.2}$$

and those with r_s lower than 0.6 are treated as invalid. This threshold will be increased if invalid motions still exist.

6.3.4 Global Pose Initialization

Given the set of valid relative motions $\mathcal{E} = \{\mathbf{T}_{ij} \in SE(3)\}$, initializing the global poses $\mathcal{T} = \{\mathbf{T}_i \in SE(3)\}$ can be easily achieved by solving the pose graph $(\mathcal{T}, \mathcal{E})$ with relative motions being the constrains. We first construct a maximum spanning tree (MST) from $\mathcal{E}$ where each



(a) (b)

(c)

Figure 6.4: An example of invalid relative motion due to visual ambiguities. (a) Two pairs of images are matched incorrectly due to the identical parking sign. (b) The corresponding point clouds from the two stations with the signs marked by red boxes. (c) The merged occupancy grids showing incorrect alignment (red ellipses). The consistency ratio, in this case, is 0.56 while it is typically over 0.7 for valid relative motions.

edge is weighted by the number of feature correspondences. Then $\mathcal{T}$ can be constructed using edges from the MST. Finally, all pairs in $\mathcal{E}$ are considered, and the following cost function is used to optimize the poses in a global frame:

$$\mathcal{T} = \arg\min \sum_{\mathbf{T}_{ij} \in \mathcal{E}} || \log(\mathbf{T}_{ij}^{-1} \mathbf{T}_i^{-1} \mathbf{T}_j) ||^2 \tag{6.3}$$

where the function $\log(\cdot) : SE(3) \rightarrow \mathbb{R}^6$ computes the twist vector of a rigid body transform.

6.3.5 RANSAC Triangulation

We adopt the robust triangulation approach proposed in [127] which uses RANSAC for each 3D feature point to find the best views for triangulation. For each track, which is a collection of observations of one feature in different camera views, two views are sampled randomly and

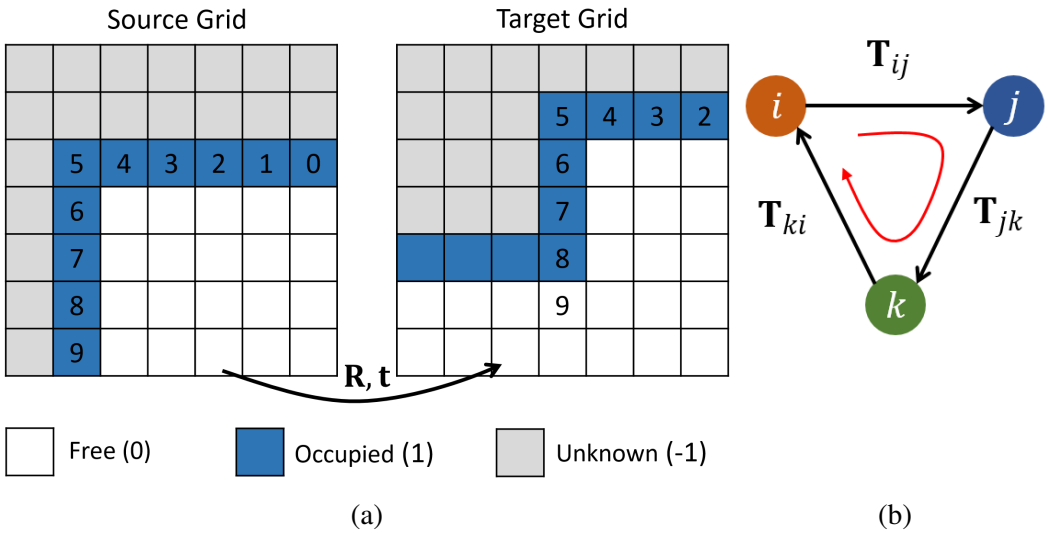


Figure 6.5: (a) A 2D example of checking the consistency of occupancy grids. The occupied cells (labeled as 0-9, painted in blue) of the source grid are transformed into the target grid using $(\mathbf{R}, \mathbf{t})$. Cells 2-8 are confirmed by the target grid while cell 9 is inconsistent. Therefore the consistency ratio is computed as $7/8$. (b) Transform-based cycle consistency checking.

the DLT method [62] is used to triangulate the point. By projecting the point onto other views and selecting the ones with a small reprojection error, inlier views can be found. This process is repeated for a number of times and the largest set of inlier views (minimum 3 views are required) is retained. Finally, inlier views are used to refine the position of a feature point in the global frame by minimizing the reprojection error.

6.3.6 Joint Pose Refinement

The pose refinement for vision-based SfM algorithms is typically achieved by Bundle Adjustment (BA). However, due to several system reasons, such as inaccurate feature locations, calibration inaccuracy, correspondence outliers and so on, the pose estimation could have significant drifts over long distances, especially when loop closures can not be found effectively. To address this issue, we consider making use of LiDAR's long-ranging capacity to constrain the camera motion. In Chapter 5, an optimization pipeline that jointly minimizes over the camera and LiDAR observations is proposed. In this work, we further introduce the joint observations that access the shared structures of the camera and the LiDAR.

Camera observations: Camera observations are defined as the feature coordinates on images and are directly obtained as inliers from the triangulation step. We define the camera observation error $\mathbf{e}_c$ as the feature reprojection error, namely

$$\mathbf{e}_c = \pi(\mathbf{x}; \mathbf{T}_i, \mathbf{P}) - \mathbf{u} \tag{6.4}$$

where function π projects feature point $\mathbf{x} \in \mathbb{R}^3$ onto the image plane, $\mathbf{P} \in \mathbb{R}^{3\times4}$ is the camera projection matrix to be chosen depending on $\mathbf{x}$ is observed by the left or right camera, $\mathbf{u} \in \mathbb{R}^2$ is the observed image coordinates. We use the symbol $\mathcal{C}$ to denote the camera observation set and $\mathcal{S}$ as the set of reconstructed feature points in 3D.

LiDAR observations: A LiDAR observation is defined as a matched pair of a key point $\mathbf{p} \in \mathbb{R}^3$ in station i and a local patch $(\mathbf{y}, \mathbf{n}) \in (\mathbb{R}^3, S(2))$ in station j. Here $\mathbf{y}$ is the nearest neighbor of $\mathbf{p}$ in station j and $\mathbf{n}$ is the normal vector of the local patch. The LiDAR observation error $\mathbf{e}_l$ is defined as the point to patch distance:

$$\mathbf{e}_l = \mathbf{n}^T \left(\tau(\mathbf{p}; \mathbf{T}_e^{-1}\mathbf{T}_j^{-1}\mathbf{T}_i\mathbf{T}_e) - \mathbf{y}\right) \tag{6.5}$$

where function $\tau(\cdot)$ transforms a point based on the given transformation matrix, $\mathbf{T}_e$ is the extrinsic transform of LiDAR w.r.t. the left camera. Our algorithm relies on a reasonable initial guess of $\mathbf{T}_e$, but will also adjust it along with the poses and structures in the optimization step. Finally, we sample 5000 key points from each point cloud and search for the corresponding local patches from stations that are within a 5-meter range. The constructed LiDAR observation set is denoted by symbol $\mathcal{L}$.

Joint observations: Similar to a LiDAR observation, a joint observation is also defined as a point-patch pair, except the point here comes from $\mathcal{S}$ instead of a LiDAR point cloud. For each feature point $\mathbf{x} \in \mathcal{S}$, we iterate through its views to find the point clouds potentially observing the same structure. Then for each point cloud, $\mathbf{x}$ is matched to the nearest local patch $(\mathbf{y}, \mathbf{n})$ which shares the same definition as in the LiDAR observations. The joint observation error $\mathbf{e}_j$ is defined as

$$\mathbf{e}_j = \mathbf{n}^T \left(\tau(\mathbf{x}; \mathbf{T}_e^{-1}\mathbf{T}_i^{-1}) - \mathbf{y}\right), \tag{6.6}$$

and the symbol $\mathcal{J}$ is used to denote the set of extracted joint observations.

Joint Optimization: The final joint optimization is done by minimizing the previously defined observation errors in a single cost function:

$$\arg\min \lambda_c \sum_{\mathcal{C}} \mathbf{e}_c^2 + \lambda_l \sum_{\mathcal{L}} \mathbf{e}_l^2 + \lambda_j \sum_{\mathcal{J}} \mathbf{e}_j^2 \tag{6.7}$$

to adjust camera poses $\mathcal{T}$, structures $\mathcal{S}$ and camera-LiDAR extrinsic transform $\mathbf{T}_e$ simultaneously. Here each component is weighted by a hand-tuned factor. In practice, we set λ_c to be 1 and tune the other two such that the cost of individual components are roughly of the same magnitude. Additionally, the error terms in (6.7) are wrapped with robust the Huber loss function and Ceres Solver [2] is used to solve the optimization problem.

Once the optimization procedure converges, observations are filtered by the errors and those with error larger than a threshold are removed from the observation set. 4 pixels, 0.1m and 0.1m are used as thresholds for the camera, LiDAR and joint observations respectively. The cost function (6.7) is solved again once outliers are removed.

Noticing that the LiDAR and joint observations are found through nearest neighbor search, which is highly dependent on the accuracy of the initialization. Therefore, the LiDAR and joint observations are recomputed once a new estimation of $\mathcal{T}$ and $\mathbf{T}_e$ is obtained. This process is repeated for

Figure 6.6: Top right is the CMU Smith Hall dataset. In the second row, from left to right are a concrete square pillar, a cylinder pillar, a T-shaped specimen, and a bridge pillar.

several iterations until the cost function converges. In our experience, 4-6 iterations are typically enough to generate well-aligned models.

6.4 Experiments

6.4.1 Sensor Pod and Datasets

A sensor pod is developed to have multiple onboard sensors, including two Ximea color camera (12 megapixel, global shutter), and a Velodyne Puck LiDAR (VLP-16) mounted on a continuously rotating motor. The scanned points from the VLP-16 are transformed into a fixed base frame using the motor angle measured by an encoder. Additionally, we assume the stereo camera and the LiDAR-motor system are both pre-calibrated. The LiDAR scanned points are transformed from the rotating LiDAR frame to a fixed motor frame, which is then referred to as the LiDAR frame. The extrinsic transform from the left camera to the fixed LiDAR frame is jointly optimized as $\mathbf{T}_e$ in our pipeline.

In this work, we collect our own datasets instead of using public SfM datasets since most of them are not suitable for our specific case that requires stereo cameras and a LiDAR. As shown in Fig. 6.6, our datasets vary from simple concrete structures to a large scale building. The Smith Hall dataset contains 276 stations (552 images and 276 LiDAR point clouds) and is most challenging due to repeated patterns and limited overlap between images. The rest datasets in the second row are of smaller scale and contain 29, 54, 25, 32 stations respectively.

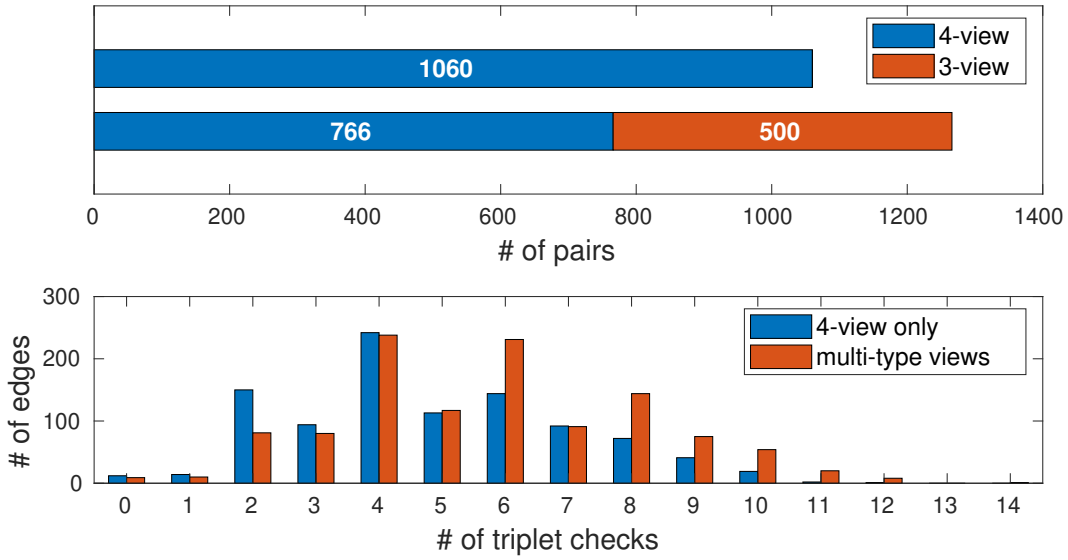


Figure 6.7: *Top:* Visualization of the number of the pairs solved from 4-view and 3-view points. *Bottom:* Histograms of the number of edges for different number of triplet checks.

6.4.2 Relative Motion Estimation

In this experiment, we show that relaxing the 4-view requirement of a feature point allows recovering more valid relative motions, which is beneficial for cycle consistency checking and model completeness. Here the Smith Hall dataset is used. As shown in the upper plot of Fig. 6.7, 500 edges are computed thanks to the 3-view feature points. Compared with the case of only 4-view features, 206 (19.4%) more relative motions are recovered. A number of 4-view cases are replaced by 3-view due to more features are available. The lower plot of Fig. 6.7 shows a histogram of the number of edges w.r.t. the number of involved triplets. We observe that the edges tend to be involved in more triplets, implying that more frequent checks can be conducted. The total number of triplet checks for this dataset is therefore increased from 4.8K to 6.7K. Finally, the initialized global poses are visualized in Fig. 6.8. Because more relative motions are recovered, our algorithm can connect more stations (276 vs. 273), which helps to pick up the loop closure around the top right corner.

6.4.3 Relative Motion Validation

In this section, we compare the outlier rejection performance of the proposed grid-based check (GC, with r_c threshold being 0.6) and success rate check (SR), to rotation cycle check (RC) used by OpenMVG [98] and transform (rotation and translation) cycle check (TC). Again, the smith hall dataset is used.

Firstly, we obtain the ground truth label by checking the difference of each relative motion to the

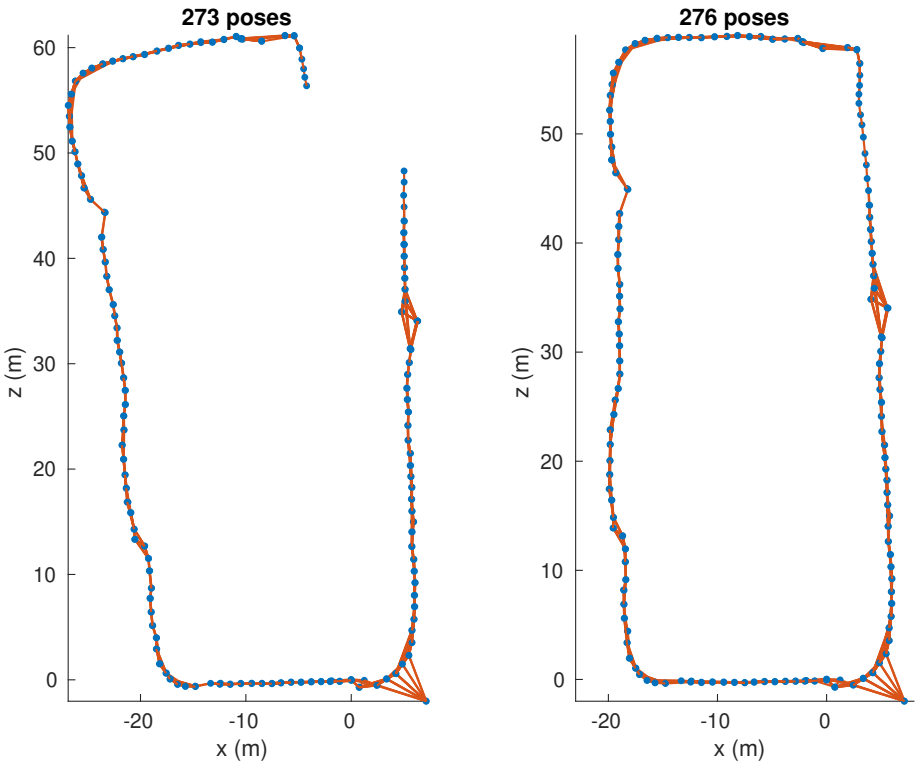


Figure 6.8: *Left:* Initialized pose graph with 4-view features. *Right:* Initialized pose graph with multi-type views.

global poses produced by our pipeline, which is of reasonable accuracy as shown in Fig. 6.9. Then those pairs with an angular error greater than 2° or translational error greater than 0.1m are labeled as outliers. After that, the validation methods discussed above are used to check the given relative motions. Specifically, a confusion matrix is computed for each case and is shown in Table 6.1. We use *recall* to measure the ratio of picked true outliers among all true outliers and *precision* to denote the fraction of true outliers among the predicted instances. We observe that a number of inconsistent relative motions could pass the RC or TC checks, corrupting the initialization of global poses. On the other hand, GC and SC checks apply more strict passing rules, hence more outliers can be rejected effectively. However, it is also observed that many valid motions fail to pass the SR check. The number is regulated by the threshold of r_s. In reconstruction experiments, we tune the r_s threshold to ensure effective rejection of outliers and meanwhile to keep as many relative motions as possible.

6.4.4 Joint Observations

For simplicity, we use a smaller scale dataset, namely the bridge pillar dataset, to show the benefits of modeling joint observations in the joint optimization. As shown in Fig. 6.10, the structures estimated from images are overlaid with the merged LiDAR point cloud. From the zoom-in view,

Table 6.1: A comparison of validation strategies.

	Confusion Matrix	Recall	Precision
RC	[1151, 0; 91, 24]	20.87%	100.00%
TC	[1151, 0; 88, 27]	23.48%	100.00%
GC	[1149, 2; 10, 105]	91.30%	98.13%
SR(0.9)	[1035, 116; 5, 110]	95.65%	48.67%
SR(0.6)	[1139, 12; 40, 75]	65.22%	86.21%
GC+SR(0.9)	[1033, 118; 3, 112]	97.39%	48.70%
GC+SR(0.6)	[1137, 14; 6, 109]	94.78%	88.62%

it is easy to see that including the joint observations leads to better alignment of the two models, indicating more accurate $\mathbf{T}_e$. This observation can be explained by analyzing the observability of $\mathbf{T}_e$ discussed in [143]. Basically, if no joint observations are used, $\mathbf{T}_e$ is constrained by the trajectories of two sensors. The solution may not be unique if the two trajectories are degraded. An extreme case is the straight-line trajectories. In this case, the translation of $\mathbf{T}_e$ could take arbitrary values, while still satisfies the rigid body transform constraints between two trajectories. On the other hand, joint observations are directly shared by the camera and the LiDAR, hence better constrains $\mathbf{T}_e$.

6.4.5 Reconstruction Tests

Sparse reconstructions: The proposed pipeline is mainly tested with our own collected datasets. The first row of Fig. 6.11 shows the reconstruction of small concrete structures. The second row (Fig. 6.11e) compares the Smith Hall reconstruction results using COLMAP, OpenMVG, and our pipeline. We also compare the model with and without joint optimization using our method. From the comparison, we observe that both COLMAP and OpenMVG fail to handle the visual ambiguities caused by the parking signs (shown in Fig. 6.5) and the limited overlapped images. Therefore the generated models are either inconsistent or incomplete. Our pipeline rejects the invalid motions effectively and allows for building a more consistent model. For quantitative analysis, we first register the models to a ground truth map acquired by a FARO terrestrial scanner, then compute the distances of model points to ground truth. The errors are encoded in the color of each point and the histograms are computed to show the error distribution. We measured the best average error of 0.02m from our jointly optimized model. Comparing the joint and stereo models shows that LiDAR helps to remove drifts within long distances loop-closure. We encourage readers to watch the supplementary video for more visualizations.

Dense reconstruction of a bridge tunnel: The second reconstruction test is conducted inside the tunnel of the Pearl Harbor Memorial Bridge, also known as the Q-bridge (see Fig. 6.12). The dataset contains 2007 stereo image pairs that captures the side walls and the ceiling of the tunnel.

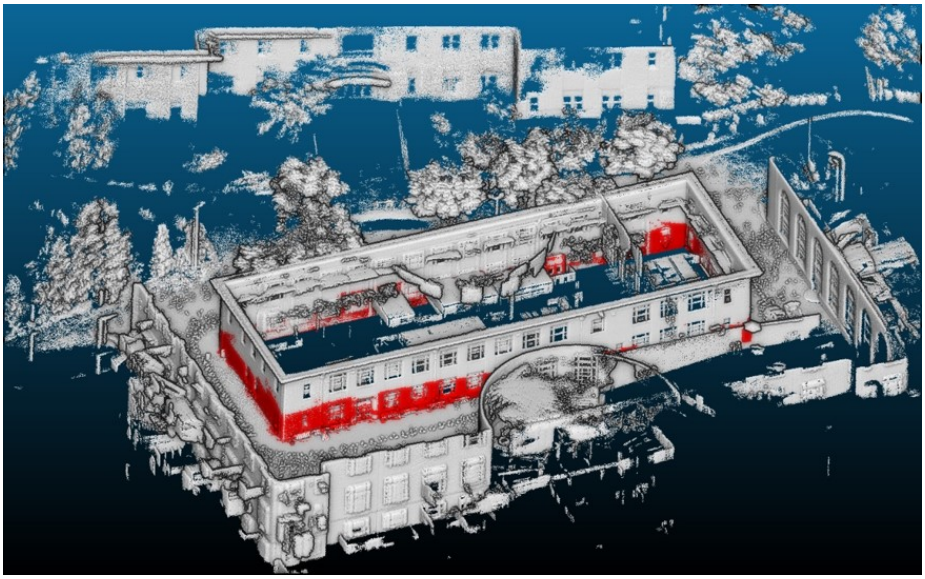


Figure 6.9: Reconstructed point cloud model (grey) of the CMU Smith Hall overlaid with visual feature points (red).

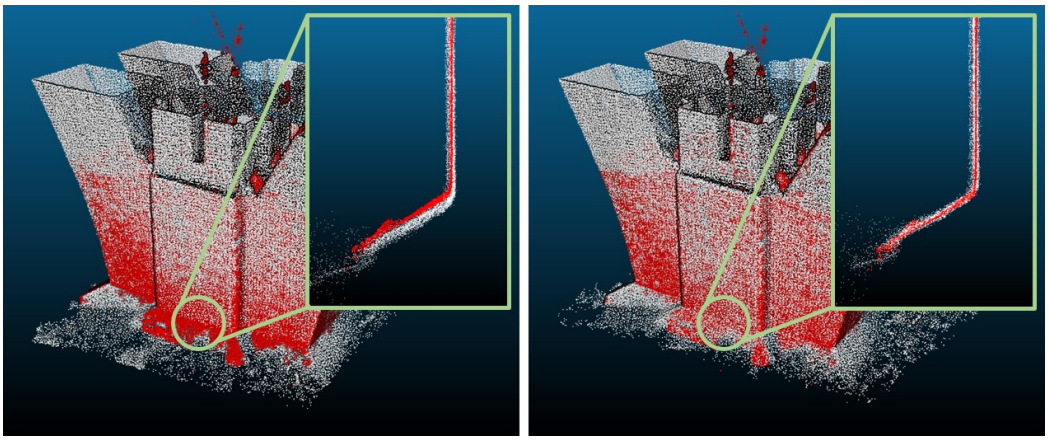
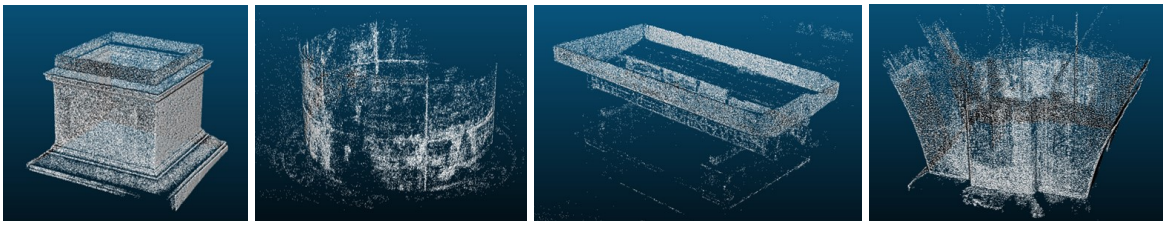


Figure 6.10: Overlay of LiDAR point cloud (grey) with reconstructed visual features (red). *Left:* Without joint observations. *Right:* With joint observations.

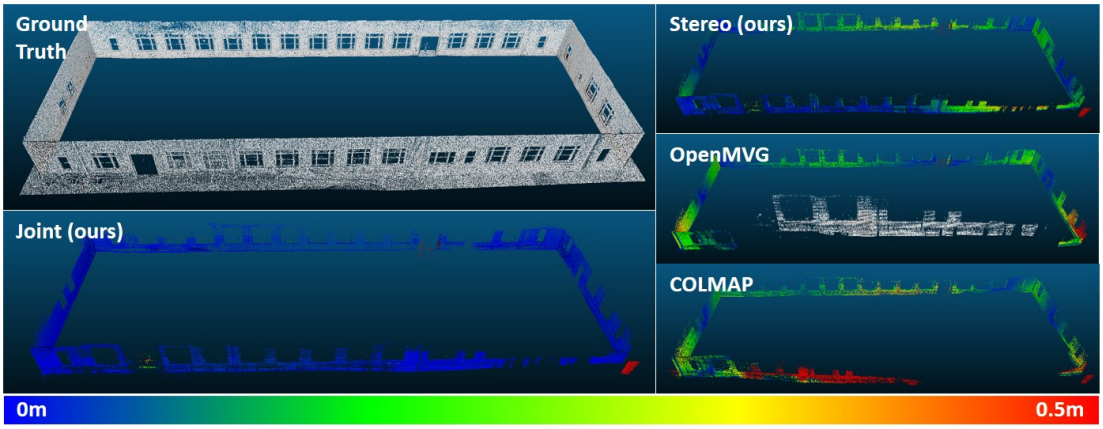
To process the collected data, we use a workstation with 32 computing cores (Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz) and 128GB RAM.

We first perform sparse reconstruction using all available images. The sparse reconstruction results are visualized in Fig. 6.12b. In this experiment, we again compared the performance with and without the LiDAR data. It can be observed that the reconstruction is corrupted with severe drifts for one side of the wall, which is corrected by jointly optimize over the LiDAR measurements.

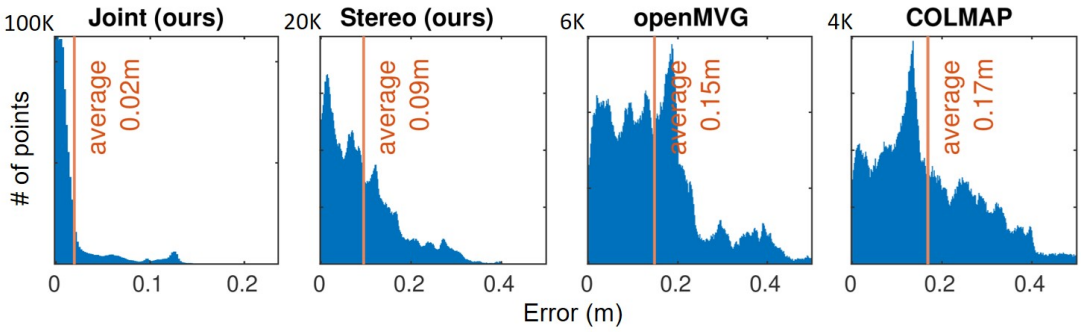
Then the reconstructed sparse model are partitioned into $1m$ cubic cells (see Fig. 6.12c) and involved images are collected for densification using OpenMVS [29]. Once all dense local maps



(a) Square (b) Cylinder (c) T-shaped (d) Bridge Pillar



(e) The Smith Hall model accuracy evaluation.



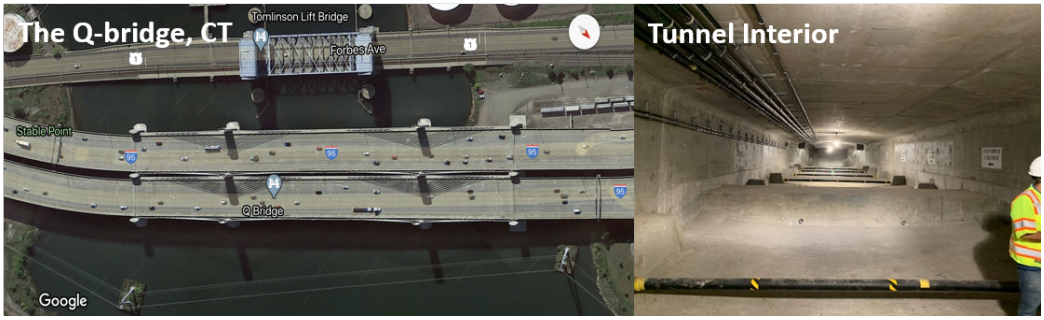
(f) Error histogram with average bar annotated.

Figure 6.11: Reconstruction results on collected datasets. (a) - (d) are reconstructed sparse models of small structures, while (e) shows and compares the Smith Hall models using different methods. The error is encoded in the colors of points. In (f) the histograms of errors are visualized.

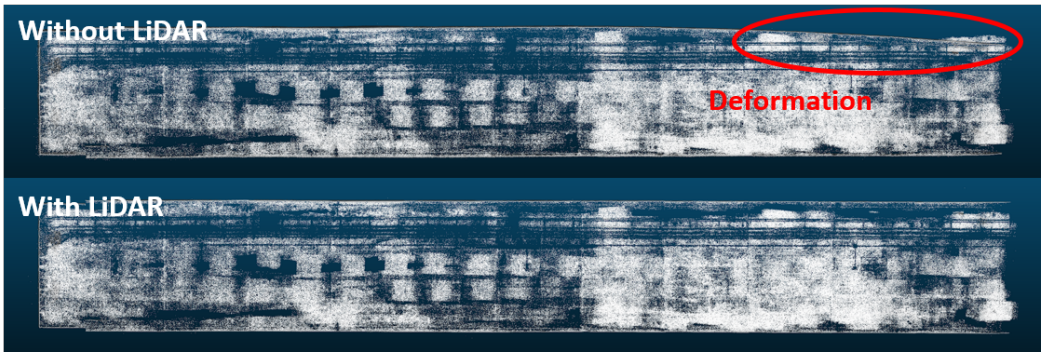
are processed, they are then merged for visualization. The Potree visualizer[1] is used to render the final bridge model as shown in Fig. 6.12d. In Table 6.2, we report the dense model statistics.

Projecting cracks on 3D model: In this test, we build dense 3D model for a concrete specimen with cracks. The cracks are extracted from images using the deep network architecture proposed in [111] and then projected onto a mesh model to find the 3D coordinates of pixels labeled as

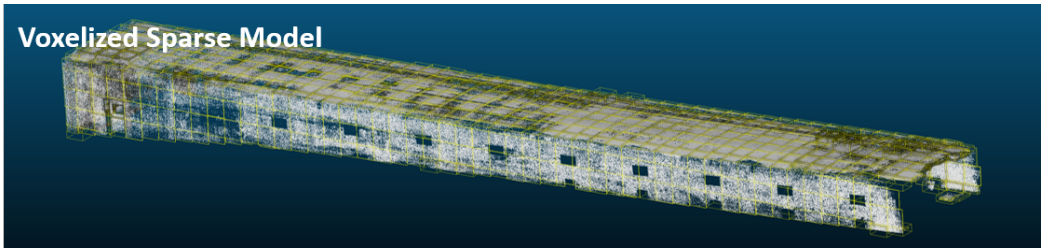
[1]Potree: `https://potree.github.io/`



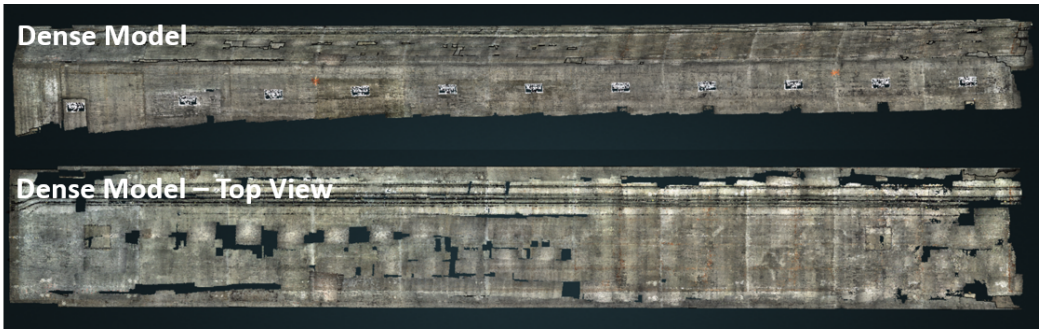
(a) The Q-bridge tunnel



(b) Comparison of sparse models



(c) Voxelization of the sparse model



(d) Dense model rendered in Potree

Figure 6.12: Reconstruction of the Q-bridge tunnel.

cracks. Then 3D crack points are visualized together with the dense model (see Fig. 6.13) providing an accurate reference for human inspectors to locate and measure the defects.

Table 6.2: Model statistics of the Q-bridge tunnel.

Tunnel Size	$65m \times 7m \times 4m$ $(L \times W \times H)$
NO. of Images	4014
Image Resolution	4112×3008 pixels
NO. of Sparse Features	1.7×10^6
Voxels Size	$1m$
NO. of Voxels	763
Dense Model Points	2×10^9
Model Density	$\approx 150/cm^2$

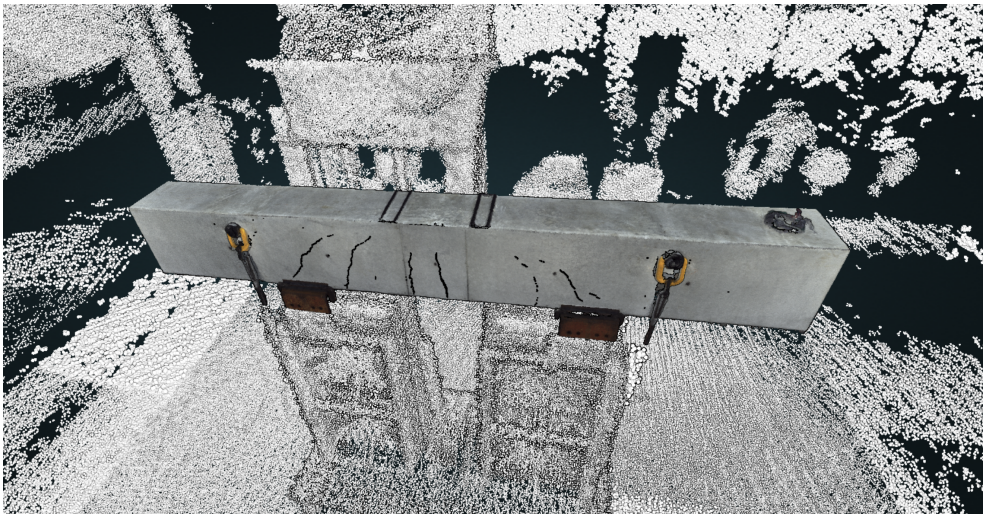


Figure 6.13: 3D visualization of cracks (black) detected on a concrete beam. LiDAR point clouds (grey) are visualized to provide context information of the environment.

6.5 Conclusion and Discussion

This work proposes a LiDAR-enhanced stereo SfM pipeline that uses LiDAR information to enhance the robustness, accuracy, consistency and completeness of the stereo SfM. Experiment results show that the proposed method is effective in finding more valid motion pairs and eliminating visual ambiguities. Additionally, we show that incorporating the joint observations of the camera and the LiDAR helps to fully constrain the extrinsic transform. Finally, the LiDAR-enhanced SfM pipeline can produce more consistent reconstruction results than the state-of-the-art SfM methods.

Learned lessons:

- The consistency validation strategy is effective in rejecting outlier feature matches as well

as relative motions. Given a set of noisy observations that are corrupted with outliers, this strategy tries to generate a large number of tests, each of which only involves a small number of samples so that the test can be carried out efficiently. Then based on the testing results, a pre-defined metric can be used to make the decision on whether each observation is an outlier or not. Particularly, this strategy is used in the famous RANSAC method, where each test is formulated as a model fitting problem to the consensus set.

- In this work, LiDAR and camera measurements are fused to improve the robustness. Specifically, the fusion first happens in the front-end, where LiDAR range data provides extra structural information to help reject visual ambiguities. Then LiDAR and camera measurements are bridged by the shared structure observation in the final optimization stage. From a sensor fusion perspective, our strategy falls into the *intermediate fusion* category. This is because sensor data are preprocessed to be fused in a shared domain (i.e., the Euclidean space), instead of being directly aligned as in early fusion and merged after separate estimations as in late fusion.
- The strategy of splitting a large problem into smaller ones is shown to be effective in dealing with large-scale datasets since it significantly reduces memory consumption. Besides, it also points to a potential future work for partitioned sparse reconstruction similar to the idea of submap-based BA [106] [94].
- Last but not least, it is learned that dense, large models can be extremely difficult to process and visualize. For instance, even for a structure as simple as the specimen in Fig. 6.13, a high-end graphics card is necessary to render such an amount of points. Therefore, for visualization, it might make more sense to use meshes to reduce the complexity of geometry and only maintain a dense texture map. However, in cases where accurate measurements have to be performed, a full resolution point cloud will become useful to provide more accurate and detailed information.

Chapter 7

Abstracted Mapping using Geometric Primitives

As discussed at the end of Chapter 6, high-resolution 3D models can cause huge overhead for downstream operations. Therefore, as the last piece of our technical contribution, we shift our focus to lightweight abstracted map representations. Here, the term *abstracted* stands for the abstraction process of using *geometric shape priors* to approximate the original map. However, rather than pursuing a precise approximation as the vast majority of methods in the field of structural modeling, we decide to cast the problem into a general setting of SLAM, and focus on the questions: how could geometric primitives be represented to reflect the nature of geometry? How can they be integrated into SLAM? And how would the sensing of abstracted shapes affect the estimation problem?

This chapter is organized as follows: Section 7.1 discusses the problem of using geometric primitives for map abstraction. Section 7.2 gives a review of the methods for different levels of landmark representation. Section 7.3 covers the fundamentals of quadrics and Section 7.4 details the SLAM formulation with quadrics factors. In Section 7.5, experiments in simulation and real world are presented. Finally, conclusions are drawn in Section 7.6.

7.1 The Representation of Landmarks

Geometric primitives such as points, lines, and planes have been widely used in SLAM to represent the 3D environment thanks to their simplicity. Many state-of-the-art SLAM systems utilize one [48] [75] or a combination [53] [139] [103] of those primitives to formulate the back-end optimization, estimating the states of the robot and landmarks simultaneously. Despite the simplicity, however, those primitives have limitations in representing more complex shapes in the environment, e.g. curved surfaces.

Recently, high-level landmarks embedded with semantic labels have been shown to significantly improve the performance of SLAM [30], localization [128], and place recognition [49]. To

include semantic information into the optimization framework of graph-SLAM, abstract shapes, such as cuboids [151] or ellipsoids [107], have been used to represent the geometry of objects. However, those works mainly focus on the shape representation of objects, hence not suitable for large surfaces.

In this work, we propose to use quadrics as a high-level representation of geometric primitives. Quadrics, as a general algebraic representation of second-order surfaces, are able to represent 17 types of shapes [7]. Because of the flexibility, quadrics have a long history being used for 3D surface representation in the field of computer vision [73] and computer graphics [113]. It is more recently that quadrics are introduced to SLAM. We can roughly break down the ongoing research of quadrics in SLAM into two categories: Firstly, ellipsoid, as a special type of quadrics with a closed shape, is used to approximate the shape and pose of objects [107]. Secondly, the representation of low-level landmarks, namely points, lines and planes, can be unified using quadrics, leading to a compact formulation of graph-SLAM with heterogeneous landmarks [103].

Our work aligns with the above two directions and extends prior work in two folds: Firstly, since quadrics have the power to represent various shapes, some of which are quite frequently seen in man-made environments (e.g. cylinders and cones), we can potentially include more types of primitives in SLAM and still keep a unified and concise formulation. Secondly, it is noticed that quadrics can be symmetric and degenerated, which could cause ambiguity in SLAM. However, those properties are not readily available from quadrics representation. Therefore, we are particularly interested in finding out how the quadrics representation implicitly encodes the geometric properties, and hope the insights would lead us to a geometrically meaningful formulation of quadrics SLAM. Therefore, the central problem of this chapter can be summarized as:

> For SLAM using high-level 3D shapes, how to develop a unified representation that generalizes to multiple types of shapes and remains geometrically meaningful?

More specifically, the main contribution of this work can be summarized as:

- A unified representation of high-level geometric primitives using quadrics is proposed. A wider spectrum of shapes is included, while previous works only consider points, lines, planes, or ellipsoids for SLAM.
- A new decomposed representation of quadrics is proposed. The decomposed representation is geometrically meaningful in that it explicitly models the degeneration and symmetry of quadrics.
- A novel decomposed quadrics factor is systematically formulated based on geometric error metrics.
- Experiments in simulation and the real world are conducted to show the proposed quadrics-based back-end framework is robust, efficient and lightweight compared to the baselines.

7.2 Related Work

Low-level landmark representation: Points are the most popular landmark representation in state-of-the-art SLAM systems [35][81]. They typically provide a sparse representation in the case of cameras, or dense representation when range sensors are used. Differently, lines (edges) and planes, that are often referred to as high-level landmarks, have been shown to improve the robustness and accuracy of SLAM systems [158]. Representation for lines include a point plus a direction [80], Plücker coordinates [163] and a pair of endpoints [115]. Planes are usually represented with a normal and a distance [139] as a non-minimal representation. Differently, Kaess [75] proposes using quaternion as a minimal representation of planes and formulates plane factors in a graph-SLAM problem. As another minimal representation, Geneva et al. [52] choose to use the closest point on a plane to the origin as the representation of planes. Although geometrically meaningful, each type of landmark requires a special factor implementation in a graph-SLAM problem, which increases the complexity of system development. To accommodate this issue, efforts are taken to unify the representation of low-level landmarks. *SPmap* [24] is perhaps the earliest attempt to develop a generic framework for geometric entities in SLAM and shows how 2D line-segments representation can be unified together with the robot poses. Closely related to our work, Nardi et al. [103] and Aloise et al. [6] introduce the concept of *matchables* as a unified representation of points, lines and planes. Differently, our work extends the usage of quadrics to higher-order surfaces such as cylinders and cones and studies the degeneracy of those shapes.

High-level landmark representation: As discussed at the end of Chapter 2, high-level representations use parameterized shapes that can be embedded with semantics to build compact yet descriptive maps. In this work, we focus on the geometry underneath the semantic information. Aligning with this line of research, Salas et al. [124] use pre-defined mesh models to represent detected objects which is difficult to generalize to unobserved objects. After that, more general shape representations are used. Yang et al. [151] fit *cuboids* as bounding boxes to describe objects. Papadakis et al. [110] extract predefined *spheres* while Nicholson et al. [107] propose to use *ellipsoids* to approximate size, position and orientation of objects. Tschopp et al. [144] demonstrate that *superquadrics* have the advantage of flexibility and meaningful parameterization. However, those methods are mainly designed for object shape representation, thus are not quite suitable to represent large surfaces such as a cylindrical pillar. This work, therefore, focuses on the representation of high-level surfaces in the environment.

7.3 Quadrics Basics

7.3.1 Quadrics Representation

Quadrics are defined implicitly by the zero-contour of a two-degree algebraic function:

$$Ax^2 + By^2 + Cz^2 + 2Dxy + 2Eyz + 2Fxz + 2Gx + 2Hy + 2Iz + J = 0 \qquad (7.1)$$

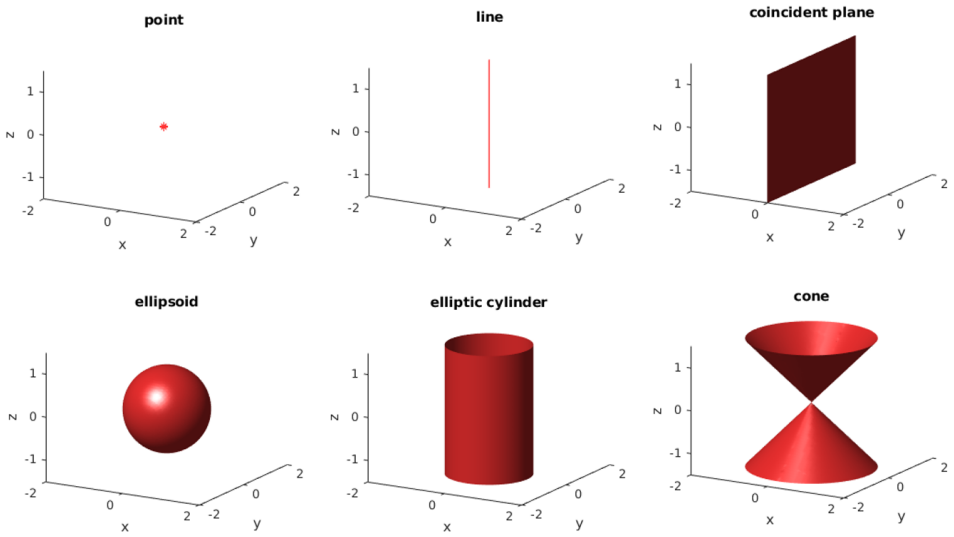


Figure 7.1: Examples of geometric primitives that can be represented as quadrics.

that contains 10 parameters but only 9 degrees of freedom due to the homogeneous parameterization. The shape function (7.1) has a compact matrix form:

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} = 0 \tag{7.2}$$

where

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \mathbf{Q} = \begin{bmatrix} A & D & F & G \\ D & B & E & H \\ F & E & C & I \\ G & H & I & J \end{bmatrix}$$

Despite the 17 subtypes of quadrics, we consider four shapes, namely coincident planes, ellipsoids, elliptic cylinders and elliptic cones, that appear most frequently in man-made structured environments. Additionally, we also consider points and lines as degenerated ellipsoids and cylinders respectively. The shape priors are shown in Fig. 7.1.

7.3.2 Quadrics extraction

Although quadrics and primitive extraction is not the focus of this work, those techniques are used in the experiments for a demonstration of quadrics SLAM. Therefore, we offer a brief discussion of the primitive extraction methods. There exist several options to extract general quadrics from a given point cloud or mesh model. As one of the most popular segmentation algorithms, region growing (RG) [82] method can be used to grow a segment by including surrounding regions with similar curvature. Once the segmentation is finished, shape parameters can be found by fitting quadrics to the points of each region. Differently, hierarchical strategies can segment the given shape in a top-down manner [79] where the model is split recursively until a good quadrics fit is found; or the bottom-up manner [9], where small regions are joined if they

Table 7.1: Quadrics Representation of Primitives

Primitives	Canonical $\mathbf{C}$	Scale $\mathbf{S}$	$\mathbf{I^s}$
Point	$\text{diag}([1\ 1\ 1\ 0])$	$\text{diag}([1\ 1\ 1\ 1])$	$[0\ 0\ 0]$
Line	$\text{diag}([1\ 1\ 0\ 0])$	$\text{diag}([1\ 1\ 1\ 1])$	$[0\ 0\ 0]$
Plane	$\text{diag}([1\ 0\ 0\ 0])$	$\text{diag}([1\ 1\ 1\ 1])$	$[0\ 0\ 0]$
Cylinder	$\text{diag}([1\ 1\ 0\ -1])$	$\text{diag}\left(\left[\frac{1}{a}\ \frac{1}{b}\ 1\ 1\right]\right)$	$[1\ 1\ 0]$
Cone	$\text{diag}([1\ 1\ -1\ 0])$	$\text{diag}\left(\left[\frac{1}{a}\ \frac{1}{b}\ 1\ 1\right]\right)$	$[1\ 1\ 0]$
Ellipsoid	$\text{diag}([1\ 1\ 1\ -1])$	$\text{diag}\left(\left[\frac{1}{a}\ \frac{1}{b}\ \frac{1}{c}\ 1\right]\right)$	$[1\ 1\ 1]$

belong to the same quadrics. Those methods have demonstrated successful application on noise-free meshes. To handle noisy data, Birdal et al. [16] propose a voting procedure when fitting with minimal point sets, which demonstrates improved robustness to noises and pushes the usage of quadrics closer to real-world data. In this work, we found that free-form quadrics fitting can be quite sensitive to laser scans which are sparse, noisy and incomplete (due to occlusions). Therefore, a type-aware primitive segmentation strategy is adopted. Specifically, the RANSAC-based primitive segmentation method proposed in [125] is used to extract specific types of primitives (e.g., cylinders, spheres and cones). Those primitives are then represented in the general quadrics form and will be used later to construct a quadrics factor graph.

7.3.3 Quadrics Composition

A quadrics $\mathbf{Q}$ contains three pieces of information: *type* (e.g. plane, cylinder etc.), *size* (e.g. radius of sphere and cylinders), and *pose* in 3D space, which corresponds to three matrices.

Canonical matrix $\mathbf{C}$: The *canonical form* of quadrics is obtained by aligning quadrics axes to the coordinate axes. In the canonical form, $\mathbf{Q}$ is reduced to the *canonical matrix* $\mathbf{C}$. For quadrics discussed in this paper, $\mathbf{C}$ is always a diagonal matrix, whose pattern uniquely determines the type. Table 7.1 summarizes the canonical matrices of the considered quadrics in this paper.

Scale matrix $\mathbf{S}$: The canonical matrix $\mathbf{C}$ represents quadrics of unit length. For example, $\mathbf{C} = \text{diag}(1, 1, 1, -1)$ defines a unit sphere. Then a diagonal scale matrix $\mathbf{S}$ can be used to scale the unit-length quadrics. However, except ellipsoids, the other quadrics in Table 7.1 are degenerated, meaning scaling in some directions will not affect the geometric shape. For example, a plane can not be scaled at all. Therefore, we use $\mathbf{I^s} \in \{0, 1\}^3$ to indicate the directions that are not scale-degenerated.

Transformation matrix $\mathbf{T}$: Let $\mathbf{T} \in SE(3)$ be the transform matrix between two frames. Then a given $\mathbf{Q}$ in one frame can be transformed to the other frame by:

$$\mathbf{Q}' = \mathbf{T}^{-T}\mathbf{Q}\mathbf{T}^{-1} \tag{7.3}$$

Composition: Any quadrics $\mathbf{Q}$ can be constructed by the composition of the three matrices:

$$\mathbf{Q} = \mathbf{T}^{-T}\mathbf{S}^{T}\mathbf{C}\mathbf{S}\mathbf{T}^{-1} \tag{7.4}$$

In preparation for the later discussions, we explicitly rewrite (7.4) as:

$$\begin{aligned}\mathbf{Q} &= \begin{bmatrix}\mathbf{R}^T & -\mathbf{R}^T\mathbf{t}\\ \mathbf{0} & 1\end{bmatrix}^T \begin{bmatrix}\mathbf{D} & \mathbf{0}\\ \mathbf{0} & d\end{bmatrix} \begin{bmatrix}\mathbf{R}^T & -\mathbf{R}^T\mathbf{t}\\ \mathbf{0} & 1\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{R}\mathbf{D}\mathbf{R}^T & -\mathbf{R}\mathbf{D}\mathbf{R}^T\mathbf{t}\\ * & \mathbf{t}^T\mathbf{R}\mathbf{D}\mathbf{R}^T\mathbf{t}+d\end{bmatrix} = \begin{bmatrix}\mathbf{E} & \mathbf{l}\\ \mathbf{l}^T & k\end{bmatrix}\end{aligned} \tag{7.5}$$

where $\mathbf{D}$ and d are the diagonal blocks of $\mathbf{S}^T\mathbf{C}\mathbf{S}$, while $\mathbf{E}$, $\mathbf{l}$, k are corresponding blocks of the resulting $\mathbf{Q}$.

7.3.4 Quadrics Decomposition

Although quadrics is powerful in representing a wide spectrum of geometric shapes, its algebraic nature makes the connection between parameters and the geometry unclear. In order to accommodate this issue, one can decompose a quadrics $\mathbf{Q}$ into the three matrices to uncover the underlying geometry.

Type identification: In practice, one may be interested in identifying ellipsoids, cylinders and cones from a given $\mathbf{Q}$. Quadrics Shape Map (QSM) [4] can be used to determine the types of quadrics by analyzing the distribution of the eigenvalues of $\mathbf{E}$. In this work, we assume the types are known from fitting specific primitives to point clouds.

Scale identification: To recover the scale information, we first normalize the given $\mathbf{Q}$ to remove the scale ambiguity:

$$\mathbf{Q} = \left|\frac{\prod \lambda_i^{\mathbf{E}}}{\prod \lambda_i^{\mathbf{Q}}}\right| \mathbf{Q} \tag{7.6}$$

where $\lambda_i^{\mathbf{E}}$ and $\lambda_i^{\mathbf{Q}}$ are nonzero eigenvalues of $\mathbf{E}$ and $\mathbf{Q}$ respectively. Specially, for cones, $\mathbf{Q}$ is normalized by the negative eigenvalue of $\mathbf{E}$. Then the scale parameters can be recovered by:

$$\begin{bmatrix}a\\ b\\ c\end{bmatrix} = \sqrt{\left|\begin{bmatrix}1/\lambda_1^{\mathbf{E}}\\ 1/\lambda_2^{\mathbf{E}}\\ 1/\lambda_3^{\mathbf{E}}\end{bmatrix}\right|} \tag{7.7}$$

assuming $a \leq b \leq c$. In degenerated cases, certain eigenvalues will be zeros. Then the scale along those directions becomes undefined as specified in $\mathbf{I}^{\mathbf{s}}$.

Pose identification: Isolating pose information from the given $\mathbf{Q}$ is to find $\mathbf{R}$ and $\mathbf{t}$ that represent the transform between the observation frame and the quadrics canonical frame, or *local frame*. According to (7.5), the rotation can be found from eigenvalue decomposition of $\mathbf{E} = \mathbf{V}\mathbf{D}\mathbf{V}^T$, while recovering $\mathbf{t}$ involves solving $\mathbf{E}\mathbf{t} + \mathbf{l} = \mathbf{0}$. However, recovering $\mathbf{R}$ and $\mathbf{t}$ needs to consider several special situations:

Table 7.2: Degeneration characterized by eigenvalues

eig($\mathbf{E}$)	Rotation	Translation	Example
$\lambda_1 \neq \lambda_2 \neq \lambda_3$	non-degenerate $\mathbf{I}^{\mathbf{R}} = [1, 1, 1]$	non-degenerate $\mathbf{I}^{\mathbf{t}} = [1, 1, 1]$	ellipsoid
$\lambda_1 \neq \lambda_2 = \lambda_3$	$\mathbf{v}_1$ degenerate $\mathbf{I}^{\mathbf{R}} = [1, 0, 0]$	non-degenerate $\mathbf{I}^{\mathbf{t}} = [1, 1, 1]$	ellipsoid (2 equal axes)
$\lambda_1 = \lambda_2 = \lambda_3$	degenerate $\mathbf{I}^{\mathbf{R}} = [0, 0, 0]$	non-degenerate $\mathbf{I}^{\mathbf{t}} = [1, 1, 1]$	sphere, point
$\lambda_1 = 0$ $\lambda_2 \neq \lambda_3 \neq 0$	non-degenerate $\mathbf{I}^{\mathbf{R}} = [1, 1, 1]$	$\mathbf{v}_1$ degenerate $\mathbf{I}^{\mathbf{t}} = [0, 1, 1]$	elliptic cylinder
$\lambda_1 = 0$ $\lambda_2 = \lambda_3 \neq 0$	$\mathbf{v}_1$ degenerate $\mathbf{I}^{\mathbf{R}} = [1, 0, 0]$	$\mathbf{v}_1$ degenerate $\mathbf{I}^{\mathbf{t}} = [0, 1, 1]$	circular cylinder, line
$\lambda_1 \neq 0$ $\lambda_2 = \lambda_3 = 0$	$\mathbf{v}_1$ degenerate $\mathbf{I}^{\mathbf{R}} = [1, 0, 0]$	$\mathbf{v}_{2,3}$ degenerate $\mathbf{I}^{\mathbf{t}} = [0, 1, 1]$	plane

- $\mathbf{V}$ is not necessarily a valid rotation matrix. The direction of eigenvector $\mathbf{v}$ can be identical or opposite to the column of $\mathbf{R}$, due to the symmetry of quadrics.
- When $\mathbf{E}$ has *nonzero eigenvalues* only, $\mathbf{t}$ can be directly recovered as $\mathbf{t} = -\mathbf{E}^{-1}\mathbf{l}$
- When $\mathbf{E}$ has *zero eigenvalues*, $\mathbf{t}$ is only partially constrained.
- When $\mathbf{E}$ has two *equal eigenvalues*, $\mathbf{Q}$ becomes revolution quadrics, where the rotation around one axis becomes degenerated.

The above situations are caused by the degeneration and symmetry of quadrics. To systematically handle these issues, we model the pose of quadrics from the perspective of *constraints*, which will be elaborated on in the next section. In Table 7.2, we summarize all possible situations of degeneration with examples.

7.4 Quadrics in Factor Graphs

7.4.1 Pose-Quadrics Constraints

To constrain the rotation, we choose to align the columns of $\mathbf{R}$ (noted as $\mathbf{r}_i$) to corresponding non-degenerate eigenvectors $\mathbf{v}_i$. An rotation activation vector $\mathbf{I}^{\mathbf{R}} \in \{0, 1\}^3$ is defined to mark the direction to be enforced (see Table 7.2). Further more, to consistently handle the situations where $\mathbf{v}_i$ is opposite to $\mathbf{r}_i$, cross product is used to measure the *unsigned direction alignment*

error:

$$\mathcal{C}(\mathbf{r}_i) = \mathbf{v}_i \times \mathbf{r}_i = \mathbf{0},\ (\text{for } \mathbf{I}_i^{\mathbf{R}} = 1) \tag{7.8}$$

As to translation, the constraint equation is:

$$\mathcal{C}(\mathbf{t}) = \mathbf{E}\mathbf{t} + \mathbf{l} = \mathbf{V}\mathbf{D}\mathbf{V}^T\mathbf{t} + \mathbf{l} = 0 \tag{7.9}$$

Similarly, translation degeneration indicator $\mathbf{I}^{\mathbf{t}} \in \{0,1\}^3$ can be defined and we can further decompose the equation and enforce the constraints explicitly:

$$\mathcal{C}(\mathbf{t}) = \lambda_i \mathbf{v}_i^T\mathbf{t} + \mathbf{v}_i^T\mathbf{l} = 0,\ (\text{for } \mathbf{I}_i^{\mathbf{t}} = 1) \tag{7.10}$$

Equation (7.10) provides an geometric interpretation of translation constraints. One such equation defines a constraining plane with normal vector $\mathbf{v}_i$ and distance $\mathbf{v}_i^T\mathbf{l}/\lambda_i$. Therefore, $\mathbf{t}$ is constrained to a point, line or plane due to the intersection of 3, 2 or 1 such constraining planes, respectively. Finally, the scale constraints can be found by directly comparing to the eigenvalues:

$$\mathcal{C}(\mathbf{s}) = s_i^2 - \lambda_i = 0,\ (\text{for } \mathbf{I}_i^{\mathbf{s}} = 1) \tag{7.11}$$

Equation (7.8) - (7.11) translate the observation of $\mathbf{Q}$ into a set of constraints parameterized by the tuple $(\mathbf{I}^{\mathbf{R}}, \mathbf{I}^{\mathbf{t}}, \mathbf{I}^{\mathbf{s}}, \mathbf{V}, \mathbf{D}, \mathbf{l})$ where the geometric properties are explicitly represented.

7.4.2 Error Function

Given the robot pose $(\mathbf{R}_r, \mathbf{t}_r)$ and the quadrics in the world frame $(\mathbf{R}_q, \mathbf{t}_q, \mathbf{s}_q)$, the error function of observed quadrics in the robot body frame is defined as the residual vector of a constraint set:

$$\mathbf{e} = \begin{pmatrix} \mathbf{e}_{\mathbf{R}} \\ \mathbf{e}_{\mathbf{t}} \\ \mathbf{e}_{\mathbf{s}} \end{pmatrix} = \begin{pmatrix} \text{diag}(\mathbf{I}^{\mathbf{R}})\,(\mathbf{V} \otimes \mathbf{\Delta}_{\mathbf{R}})^T \\ \text{diag}(\mathbf{I}^{\mathbf{t}})(\mathbf{D}\mathbf{V}^T\mathbf{\Delta}_{\mathbf{t}} + \mathbf{V}^T\mathbf{l}) \\ \text{diag}(\mathbf{I}^{\mathbf{s}})(\mathbf{s}_q^2 - \mathbf{\Lambda}) \end{pmatrix} \tag{7.12}$$

where $\otimes$ means column-wise cross product. $\mathbf{\Delta}_{\mathbf{R}} = \mathbf{R}_r^T\mathbf{R}_q$ and $\mathbf{\Delta}_{\mathbf{t}} = \mathbf{R}_r^T(\mathbf{t}_q - \mathbf{t}_r)$ are the rotation and translation of quadrics pose transformed into the robot frame. $\mathbf{\Lambda} = [\lambda_1, \lambda_2, \lambda_3]$ is the vector of eigenvalues stored in $\mathbf{D}$. Here, $\mathbf{e}_{\mathbf{R}}$ is a 3×3 matrix and will be vectorized before being stacked into the error vector.

7.4.3 Observation Uncertainty and Weighting

One direct benefit of using decomposed constraint representation is that it allows easy incorporation of uncertainties, or weights, to measure the *strength* of constraints (7.8)-(7.11). We adopt a simple approach to compute the weight of a shape as $\tanh(\lambda N)$, where N is the number of points and λ is a scaling coefficient to be tuned by hand. The adopted strategy reduces the weights of small shapes that tend to have higher uncertainty in fitted parameters.

Algorithm 2: LM Algorithm for Quadrics Factor Graph

1: **Input:** Initial states $\mathbf{X}_0 \in \mathbb{R}^{(9M+6N)\times 1}$ of N poses, M quadrics landmarks, and K observations $\{\mathbf{Q}_k\}$
2: **Output:** Optimized states $\mathbf{X}^*$
3: Decomposition: $\mathbf{Q}_k \rightarrow (\mathbf{I}_k^{\mathbf{R}}, \mathbf{I}_k^{\mathbf{t}}, \mathbf{I}_k^{\mathbf{s}}, \mathbf{V}_k, \mathbf{D}_k, \mathbf{l}_k)$
4: Initialization: $\mathbf{X} \leftarrow \mathbf{X}_0$
5: **while** not converged **do**
6: **for** each observation **do**
7: Pose Jacobian: $\mathbf{J}_r = \left[\frac{\partial \mathbf{e}}{\partial \mathbf{R}_r}, \frac{\partial \mathbf{e}}{\partial \mathbf{t}_r}\right]$
8: Quadrics Jacobian: $\mathbf{J}_q = \left[\frac{\partial \mathbf{e}}{\partial \mathbf{R}_q}, \frac{\partial \mathbf{e}}{\partial \mathbf{t}_q}, \frac{\partial \mathbf{e}}{\partial \mathbf{s}_q}\right]$
9: Evaluate observation error: $\mathbf{e} = \left[\mathbf{e}^{\mathbf{R}}\,; \mathbf{e}^{\mathbf{t}}\,; \mathbf{e}^{\mathbf{s}}\right]$
10: Update b: $\mathbf{b} \leftarrow \mathbf{b} + \left[\cdots\ \mathbf{J}_r^T\mathbf{\Omega}\mathbf{e}\ \cdots\ \mathbf{J}_q^T\mathbf{\Omega}\mathbf{e}\ \cdots\right]$
11: Update $\mathbf{H}$: $\mathbf{H} \leftarrow \mathbf{H} + \begin{bmatrix} \vdots & \vdots \\ \cdots\ \mathbf{J}_r^T\mathbf{\Omega}\mathbf{J}_r\ \cdots & \mathbf{J}_r^T\mathbf{\Omega}\mathbf{J}_q\ \cdots \\ \vdots & \vdots \\ \cdots\ \mathbf{J}_q^T\mathbf{\Omega}\mathbf{J}_r\ \cdots & \mathbf{J}_q^T\mathbf{\Omega}\mathbf{J}_q\ \cdots \\ \vdots & \vdots \end{bmatrix}$
12: **end for**
13: Compute LM update: $\mathbf{\Delta} = -(\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{b}$
14: Apply update: $\mathbf{X} \leftarrow \mathbf{X} \boxplus \mathbf{\Delta}$
15: **end while**
16: Return $\mathbf{X}^* = \mathbf{X}$

* λ in line 13 is the LM damper updated in each iteration [37].

7.4.4 Solving the Factor Graph

Given a graph with quadrics, the cost function is constructed by accumulating the errors of each observation:

$$f = \sum \mathbf{e}^T\mathbf{\Omega}\mathbf{e} \tag{7.13}$$

where $\mathbf{\Omega} = \text{diag}(\mathbf{\Sigma}_{\boldsymbol{\theta}_q}^{-1}, \mathbf{\Sigma}_{\mathbf{t}_q}^{-1}, \mathbf{\Sigma}_{\mathbf{s}_q}^{-1})$ is the information matrix characterizing the weight of each component. In Algorithm 2, we report the basic steps of Levenberg–Marquardt (LM) method [97] for graph optimization. Sparsity is preserved by line 10-11, where only relative blocks of $\mathbf{b}$ and $\mathbf{H}$ are updated. Sparse Cholesky factorization is applied to solve line 13. We refer the readers to [35][57] for more information about the sparse structure of factor graphs. The derivation of Jacobians for quadrics factors can be found in Appendix B.

7.4.5 Baseline Parameterizations

In this section, we discuss two baseline parameterizations as comparisons to the proposed decomposed representation.

Full parameterization: A naive method to formulate the quadrics observation error is to use the full paramterization, namely the 10-D quadrics vector $\mathbf{q}$:

$$\mathbf{q} = \begin{bmatrix} A & B & C & D & E & F & G & H & I & J \end{bmatrix}^T \tag{7.14}$$

Then the observation error is evaluated by first transforming the $\mathbf{q}$ into robot body frame and then compute the difference with observation $\bar{\mathbf{q}}$:

$$\mathbf{e} = \bar{\mathbf{q}} - \left(\mathbf{T}_r^T \left(\mathbf{q} \right)^\wedge \mathbf{T}_r \right)^\vee \in \mathbb{R}^{10 \times 1} \tag{7.15}$$

where operator $(\cdot)^\vee$ and $(\cdot)^\wedge$ compute the quadrics vector and matrix respectively. About the full representation:

- The observation model has a simpler expression and easy to implement;
- The metric is algebraic error instead of geometric error, which could introduce bias to estimation [5].
- It is difficult to interpret the uncertainties of $\mathbf{q}$.

Regularized full parameterization: Inspired by [107], we implement another baseline where the structure of quadrics are explicitly modeled:

$$\mathbf{e} = \bar{\mathbf{q}} - \left(\mathbf{T}_r^T \mathbf{Q} \mathbf{T}_r \right)^\vee \in \mathbb{R}^{10 \times 1} \tag{7.16}$$

In here, $\mathbf{Q}$ is constructed as in (7.4) from the quadrics states $(\mathbf{R}_q, \mathbf{t}_q, \mathbf{s}_q)$. Equation (7.16) explicitly models rotation, translation and scale of quadrics, but still computes the algebraic error. Compared to (7.15), the type of quadrics is now treated as prior knowledge and therefore the estimation is regularized. From now on, we use decomposed (`D`), full (`F`) and regularized-full (`RF`) parameterization to denote the proposed, baseline 1 and baseline 2 representation respectively.

7.5 Experiments

7.5.1 Geometric Error vs. Algebraic Error

In this section, we compare the geometric and algebraic error formulations in simulation to investigate their convergence behaviors when solving the quadrics factor graph.

Synthetic environment: The synthetic environment, as shown in Fig. 7.3, is a Manhattan-like world that contains 15 quadric landmarks of different types. The quadrics are randomly generated in a bounded space ($6m \times 6m \times 1m$). The simulated robot trajectory is shown as the red curve which contains 50 frames whose x-axis points to the origin. For each frame, the robot will sense the surrounding environment and the nearest $K = 10$ quadrics are observed.

Noise simulation: To observe how different formulations react to various levels of noise, 2 types of perturbation are simulated. Firstly, the robot poses $\{\mathbf{R}_r, \mathbf{t}_r\}$ and quadrics parameters $\{\mathbf{R}_q, \mathbf{t}_q, \mathbf{s}_q\}$ are perturbed by adding a small Gaussian offset with $\sigma_{\mathbf{x}_0} = (\sigma_{\boldsymbol{\theta}_r}, \sigma_{\mathbf{t}_r}, \sigma_{\boldsymbol{\theta}_q}, \sigma_{\mathbf{t}_q}, \sigma_{\mathbf{s}_q})$.

Table 7.3: Perturbation Configurations

	Initialization Noise $\sigma_{\mathbf{x}_0}$ $(\sigma_{\boldsymbol{\theta}_r}, \sigma_{\mathbf{t}_r}, \sigma_{\boldsymbol{\theta}_q}, \sigma_{\mathbf{t}_q}, \sigma_{\mathbf{s}_q})$	Observation Noise $\sigma_{\bar{\mathbf{q}}}$ $(\sigma_{\bar{\boldsymbol{\theta}}_q}, \sigma_{\bar{\mathbf{t}}_q}, \sigma_{\bar{\mathbf{s}}_q})$
Low (L)	$(1°, 0.1, 1°, 0.1, 0.01)$	$(1°, 0.1, 0.01)$
Medium (M)	$(5°, 0.5, 5°, 0.5, 0.02)$	$(2°, 0.2, 0.02)$
High (H)	$(50°, 5.0, 50°, 5.0, 0.05)$	$(5°, 0.5, 0.05)$

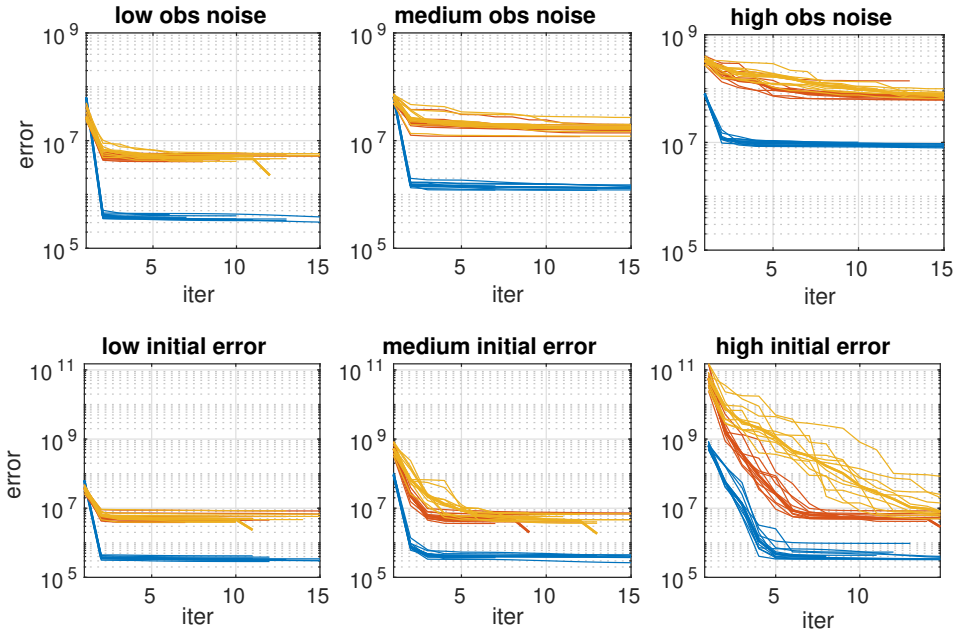


Figure 7.2: The convergence plot with increasing initialization and observation error. Each configuration is repeated 10 times. Color codes: decomposed (blue), full (red), regularized-full parameterization (orange)

This generates the initial guess for graph optimization. Secondly, each quadrics observation is perturbed in terms of rotation, translation and scale according to Gaussian noise $\sigma_{\bar{\mathbf{q}}} = (\sigma_{\bar{\boldsymbol{\theta}}_q}, \sigma_{\bar{\mathbf{t}}_q}, \sigma_{\bar{\mathbf{s}}_q})$. This gives a set of noisy observations $\{\bar{\mathbf{Q}}\}$. Table 7.3 defines 3 levels of noise: low (L), medium (M) and high (H), for each type of perturbation.

Solving the factor graph: To directly observe the behavior of the proposed quadrics factor, we choose to construct the factor graph *only* containing pose-quadrics factors and a prior factor of the first robot pose.

The convergence behavior under various noise levels is reported in Fig. 7.2. The upper plot shows the error-iteration curves of increasing observation noise. In this test, the initialization noise is at a low level. In the lower plot, we report the convergence behavior under increasing initialization noise. In this test, the observation noise is kept at a low level. We observe that the decomposed representation has a faster convergence rate, especially at a high noise level.

Besides, the curves of decomposed parameterization also tend to have fewer variations, which indicates that the cost function using geometric error has better convexity.

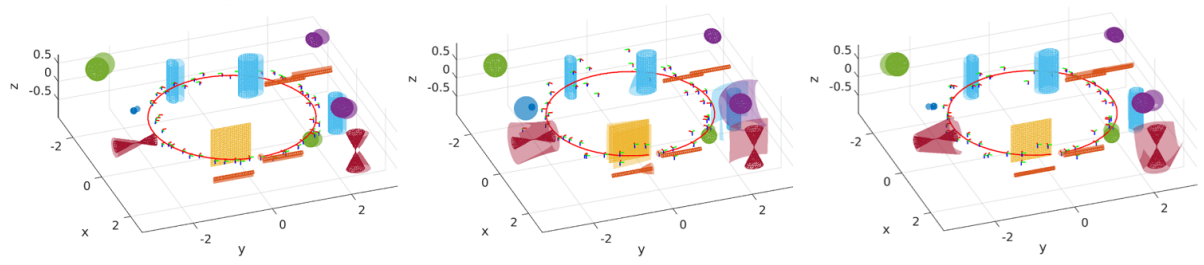


Figure 7.3: Optimization results using decomposed (Left), full (Middle), and regularized-full (Right) parameterizations. Ground truth trajectories and quadrics are visualized as solid curves and meshes respectively. Optimized poses and quadrics are drawn as frames and transparent surfaces respectively. Types of simulated shapes include points, lines, planes, ellipsoids, spheres, cylinders, and cones.

We then qualitatively evaluate the converging basin for different parameterizations. In Fig. 7.3, we compare the optimized robot poses and quadrics to the ground truth under high initialization noise. It is observed that the optimized quadrics and robot poses stay closer to the ground truth when using decomposed parameterization, indicating a wider converging basin. Additionally, the optimized quadrics with full parameterization will change the type to compensate for noises, while shapes using the other two parameterizations are well regularized.

For the above 6 noise configurations, we also compare the final optimized states to the ground truth. For trajectories, we compute the absolute trajectory error (ATE). For quadrics, we directly compare the quadrics vector. In the case of decomposed representation, the quadrics vector is reconstructed using (7.4). Table 7.4 shows the trajectory and quadrics errors in 6 noise configurations. Note that the errors are averaged across 10 tests sharing the same noise configurations. It is observed that decomposed parameterization consistently has smaller translation and quadrics errors. Although the regularized parameterization performs better in most cases in terms of rotation, the difference is small.

Table 7.4: Trajecotry and Quadrics Errors

$\sigma_{\bar{\mathbf{q}}}$-$\sigma_{\mathbf{x}_0}$	Rotation (rad)			Translation (m)			Quadrics		
	D	F	RF	D	F	RF	D	F	RF
L-L	0.055	0.049	**0.048**	**0.152**	0.218	0.213	**0.102**	0.143	0.138
M-L	0.125	0.112	**0.110**	**0.310**	0.548	0.515	**0.211**	0.362	0.312
H-L	0.309	**0.306**	0.335	**0.803**	2.070	1.854	**0.614**	1.050	0.950
L-L	0.059	0.060	**0.058**	**0.163**	0.222	0.211	**0.106**	0.141	0.135
L-M	0.057	0.053	**0.051**	**0.157**	0.259	0.252	**0.104**	0.173	0.169
L-H	**0.058**	0.062	0.264	**0.180**	0.265	0.856	**0.121**	0.198	0.527

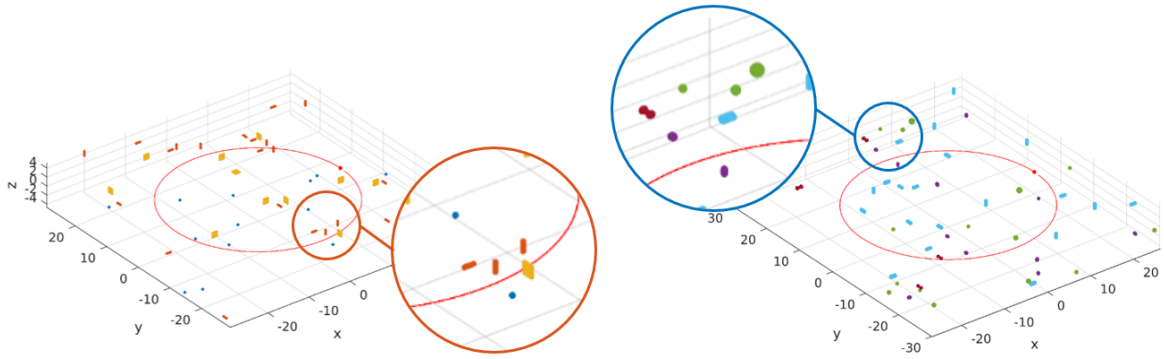


Figure 7.4: Illustration of simulated maps. *Left:* Low-level map consists of points, lines and planes. *Right:* High-level maps contains ellipsoids, spheres, cylinders, and cones. Similar as before, the red curve indicates the predefined robot path.

Discussion on the translation: Through the experiments, we found that the estimation accuracy of full and regularized-full parameterization is quite sensitive to the translation and scale perturbation of observation $\bar{\mathbf{Q}}$. Even a small perturbation would cause the final result to converge to a local minimum (see large errors of `F` and `RF` in Table 7.4). This can be explained by their correlation in $\mathbf{Q}$. From (7.3), we can see that $\mathbf{t}$ and $\mathbf{D}$ are multiplied in the relationship equation $-\mathbf{R}\mathbf{D}\mathbf{R}^T\mathbf{t} = \mathbf{l}$. In the case of small quadrics, small size noise will result in dramatic changes in the values of $\mathbf{D}$ due to the inverse relationship. Then during optimization, $\mathbf{t}$ tends to compensate for the amplified effects of scale noise thus leading to inaccurate estimation.

7.5.2 High-level Shapes vs. Low-level Shapes

Low-level primitives, such as points, lines, and planes, have been widely used in modern SLAM systems and have demonstrated great success thanks to their simplicity and richness, which questions the significance of using high-level shapes (e.g. ellipsoids, cylinders and cones) for SLAM. Therefore, in this section, we investigate the different aspects of using low-level and high-level shapes for graph-SLAM optimization.

Simulation tests: Two Manhattan-like worlds are created to contain only low-level or high-level shapes (see Fig. 7.4). Similar as the previous experiment, we let the robot follow a predefined circular trajectory that contains 100 frames (shown as the red curve) and perceive the surrounding shapes. The number of observed shapes per frame is configured to be 5, 10 or 15. Once all the observations are obtained, the quadrics factor graph is constructed and optimized.

The optimization results are reported in Fig. 7.5 and we can make the following observations:

- In the first row, the adjacency matrix stores the number of co-observed shapes between two frames and therefore the color shows the "density" of connections. Since the number of observations is controlled, we can observe that the adjacency matrices of low-level and high-level tests are similar within each configurations. This is to make sure the following comparison is fair in terms of number of observations.

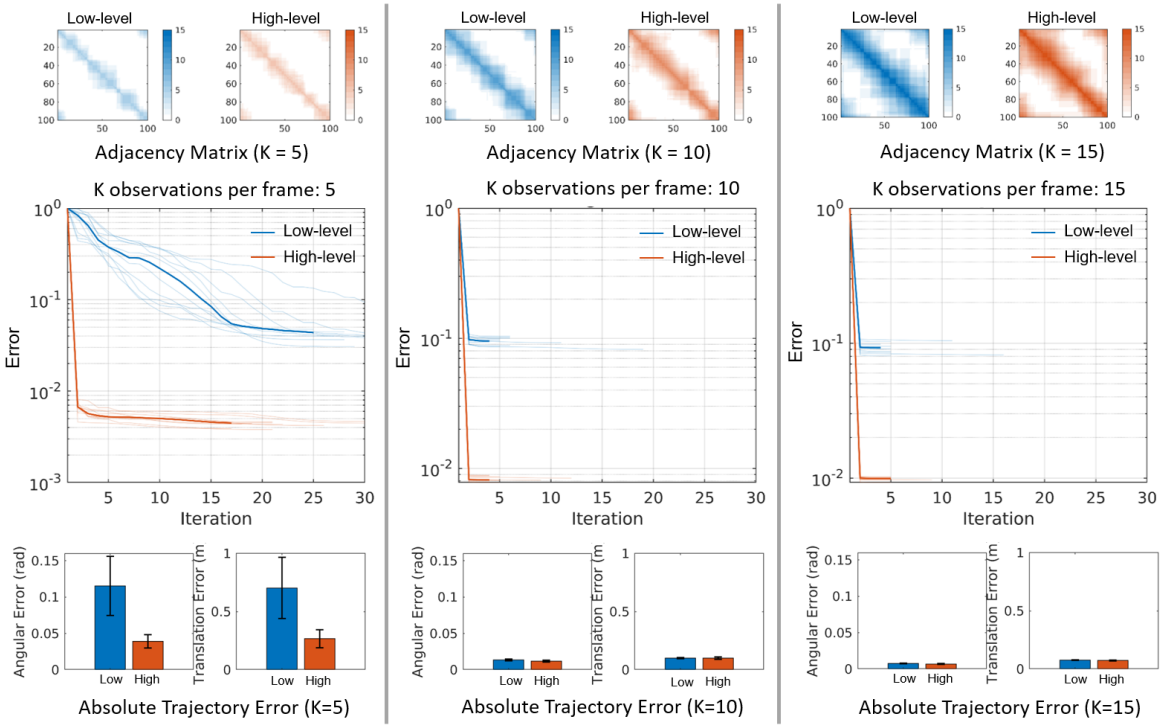


Figure 7.5: Comparison of convergence using low-level (blue) and high-level (red) shapes. The first row shows the adjacency matrix of the sampled pose graph and the value of each element indicate the number of shared shape observations. The second row compares the convergence rate when robot observes different number of low-level or high-level shapes. The third row shows the absolute trajectory error in rotation and translation under different configurations.

- The second row of Fig. 7.5 visualizes the convergence rate of different configurations. Note that the errors are normalized by their initial values. It can be seen that when the observation is limited (in the case of $K = 5$), using high-level shapes leads to a much faster convergence rate. However, when the number of observations is increased (in the cases of $K = 10$ and $K = 15$), the convergence rates are similar. This can be explained by the fact that high-level shapes provide more constraints per observation, which benefits the optimization when the total observations are limited. And when the number of observations is increased, both low-level and high-level based optimization have collected enough constraints, making the convergence rate similar.
- The third row compares the ATE of estimated trajectory. In the case of $K = 5$, we observe significant improvements in the accuracy of robot orientation and position. However, the difference becomes less obvious as K increases. This observation is again related to the constraints provided by high-level shapes in the case of limited robot observations.

Real-world tests: Experiments in the real-world are conducted following a similar manner as the simulation. An indoor cafe area where there exist multiple circular and elliptic cylinders is selected as the testing environment. In this experiment, we first create an accurate global

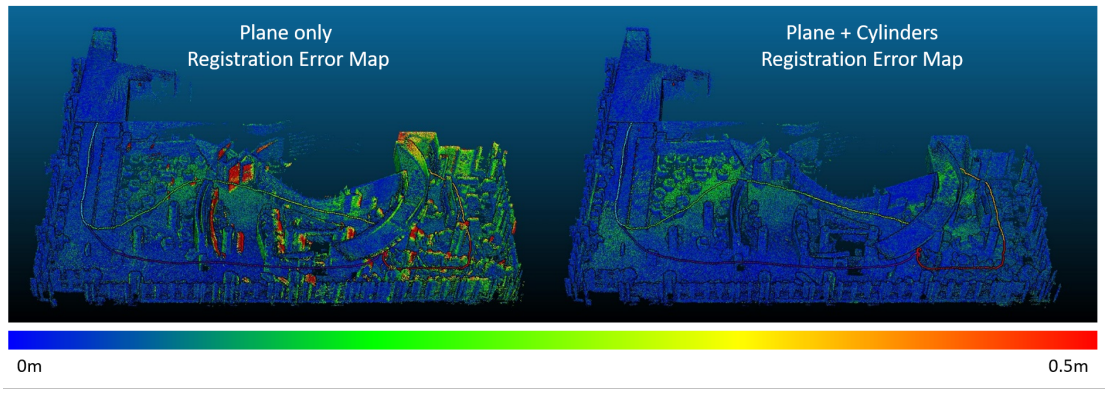


Figure 7.6: Illustration of primitive-based registration.

point cloud map as the ground truth model using a fine-tuned LiDAR odometry algorithm [159]. Then a set of submaps are created sequentially by accumulating the laser scans within a 20-second time window. The upper-left plot of Fig. 7.6 shows the global map and the trajectory segments for each submap. After that, we extract planes (blue) and cylinders (red) from each submap. The segmented submaps are shown from a top-down viewpoint and extracted cylinders are highlighted. Given the submaps represented by the extracted shapes, we setup the experiment as joint registration of all submaps from a perturbed initial guess. To compare low-level and high-level shapes, two configurations are used: one that only considers planes in the environment, while the other combines both planes and cylinders. The two registration problems are solved using the same quadrics factor graph framework as in the simulation.

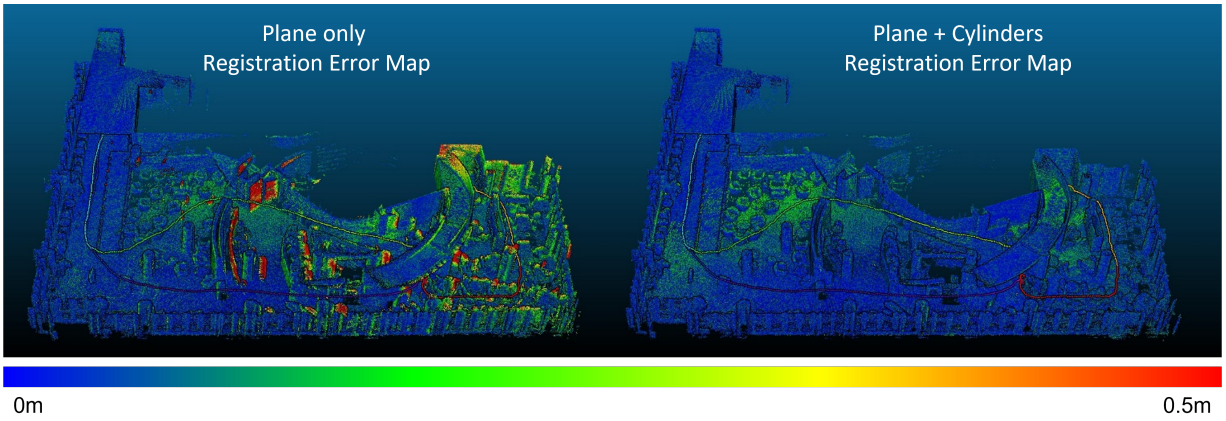


Figure 7.7: Comparison of registration error map obtained using plane-only (Left) and plane-plus-cylinder configurations (Right).

The selected environment can be treated as a Manhattan world because the planes are mostly horizontal and vertical. However, there exist a special situation (as shown in submap 3 and 4) where only one vertical wall is observed by both frames. This makes the plane-only registration under-constrained in the direction of the intersecting line of the vertical plane and the ground plane. When cylinders are also considered, the central elliptic cylinder is observed by both frames, re-

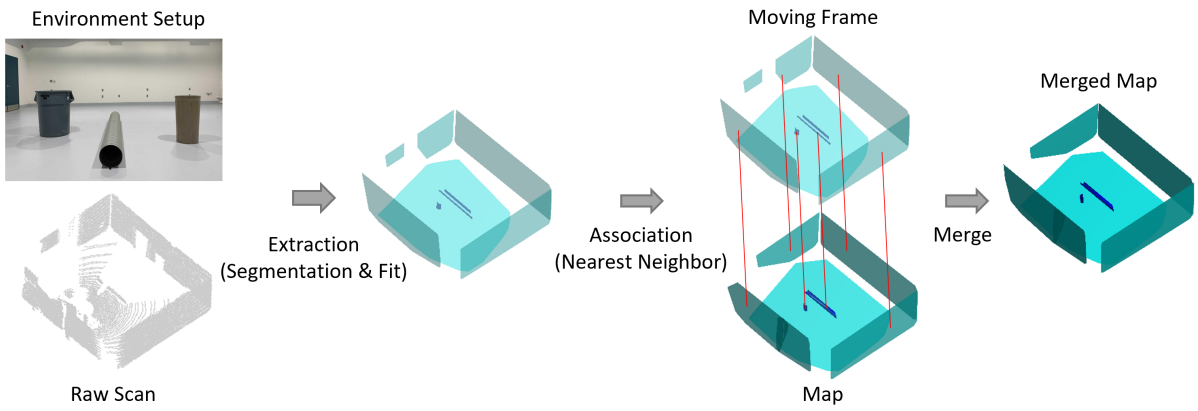


Figure 7.8: An illustration of the front-end of the primitive-based SLAM.

straining the motion in the horizontal plane. Finally, the registration results are compared to the ground truth map and the error map is visualized in Fig. 7.7, from which it is easy to see that combining planes and cylinders can generate more consistent maps.

7.5.3 Primitive-based SLAM

Although the main focus of this work is the discussion of how high-level shapes affect the back-end of graph-SLAM, we provide a simple implementation of the front-end to demonstrate the primitive-based SLAM pipeline in the real world.

For data collection, we use an Ouster OS1 LiDAR[1] to map a room with cylinders and planes (see Fig. 7.8). The front-end is implemented on top of a LiDAR odometry pipeline [159]. Firstly, shapes are extracted from selected laser scans (one scan per second) using the RANSAC method proposed in [125]. Then quadrics are associated incrementally by computing the Taubin distance [140] of shape points to existing quadrics in the map. If the averaged distance is smaller than a threshold, then two quadrics are matched. Otherwise, a new quadric is created and added to the map. Once all scans are processed, we obtain a list of quadrics, each of which has a list of frame views. The quadrics set is further pruned to only keep shapes with more than 6 views. Finally, a graph consist of robot poses and quadrics is obtained.

In the back-end stage, the graph is optimized using the LM algorithm presented in Algorithm 2. Mapping results are compared qualitatively with the LiDAR odometry and reported in Fig. 7.10. From the shown point clouds, we can see that the proposed quadrics-based back-end refines the alignment of point clouds. Additionally, the quadrics estimation is regularized and optimized. As shown in the zoomed-in views, the central axis (shown as the blue z-axis) of cylinders lies closer to the shape center in the optimized map, while the initial estimation is slightly off due to inaccurate shape fitting from partial observations. Finally, although the visualization uses point clouds, the optimization only involves 9 quadrics (shown in Fig. 7.9 plus a hidden ceiling plane),

[1]Ouster product website: `https://ouster.com/products/scanning-lidar/os1-sensor`

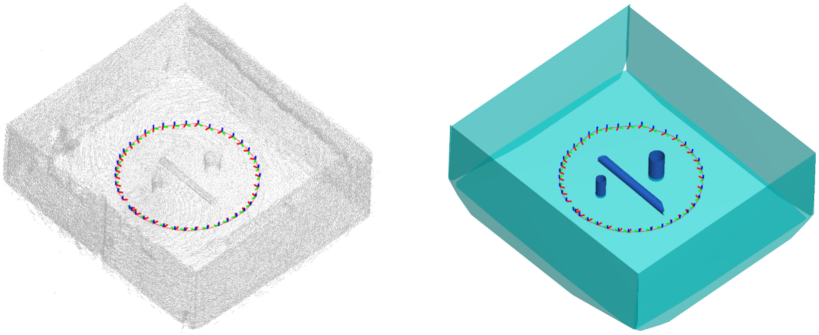


Figure 7.9: *Left:* Map represented by dense low-level points (179840 points). *Right:* Map represented by compact high-level shapes (6 planes, 3 cylinders).

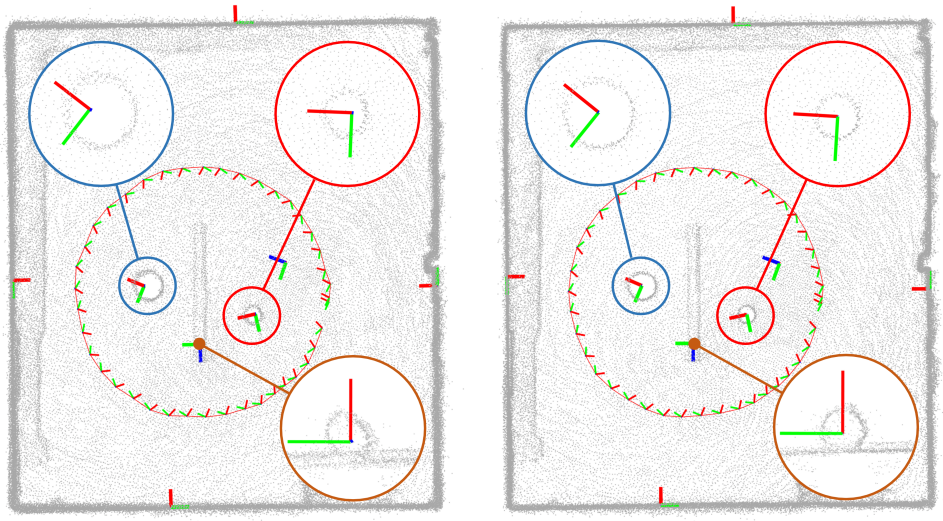


Figure 7.10: Qualitative comparison of point cloud map before (left) and after (right) quadrics graph optimization.

making the framework lightweight.

7.6 Conclusion and Discussion

In this work, we unify the geometric primitive representation using quadrics, which generalizes to a wider spectrum of shapes. Additionally, we provide a decomposed representation of quadrics that explicitly discloses the geometric properties of shapes such as degeneration and symmetry. Then based on the decomposition, we show that the observation of quadrics can be translated into constraints to the robot poses, and thus the formulation of quadrics factors in graph-SLAM is developed. In simulation experiments, we show that the decomposed quadrics factors utilize

shape priors and optimize a geometric error, making it more stable and efficient than the algebraic baselines. And using both simulated and real-world data, we show that high-level shapes generally provide more constraints than low-level primitives, which can improve the estimation accuracy. Finally, in a simple real-world environment, we demonstrate that a primitive-based SLAM pipeline using the proposed quadrics factor graph could produce accurate and lightweight maps.

Learned lessons:

- Algebraic metric vs. geometric metric. It is learned that the geometric error is more stable than the algebraic error for estimation problems in SLAM. This is because the variables in SLAM mostly describe the geometric properties of robot poses or the environment. In this case, using algebraic error as the cost function could be sensitive to noises.
- High-level shapes, although not as rich as low-level shapes in general environments, can provide more constraints via each observation. Therefore, in practice, a hybrid or combined approach of both low-level and high-level primitives might make more sense to take advantage of both sides.

Several unsolved questions that limit the usage of quadrics-based SLAM in the real world could potentially be the directions of future work. Firstly, to make use of quadrics in a practical SLAM pipeline, the front-end still remains challenging. Instead of a simplistic front-end for the proof-of-concept, a practical one would need to solve quadrics extraction fastly and accurately. Secondly, it is unclear how to estimate the covariance matrix of quadrics fitting from partially observed data in a principled way that models the anisotropic nature of uncertainty. Finally, since high-level shapes have been shown to significantly reduce the number of landmarks in the map while still capturing the overall layout, detecting loop-closure in lightweight maps would be another interesting direction to explore.

Chapter 8

Conclusions

This thesis argues that degeneracy prediction, sensor fusion and data abstraction are three key ingredients to allow for robust, accurate and informative robot-based inspection. This statement may seem to be targeted specifically for inspection. However, through the development of this thesis, we wish to provide insights that are generally applicable to robotics applications. Therefore, this chapter aims to summarize the presented works and discuss the learned lessons.

Specifically, this chapter is organized as follows: in Section 8.1, we revisit the core challenges and summarize how they are addressed through a series of works under the guidance of the central idea. Then we provide a high-level discussion of the lessons learned from this work. Possible directions for future research is presented in Section 8.3. Finally, we offer concluding remarks in Section 8.4.

8.1 Summary and Contributions

At the beginning of this thesis, to achieve the goal of robust state estimation and mapping for robot-based inspection, we proposed to investigate three core challenges. Now we briefly revisit those problems and discuss how they are addressed in the presented works:

Challenge 1: Geometric degeneracy in structured environments.
Challenge 1 poses the question: how can the robot robustly localize itself in geometrically degenerated environments? We address this question in two steps. Firstly, we resolve the general localization problem in **Chapter 3**. A LiDAR-inertial localization algorithm is presented and shown to be reliable to various environments and aggressive motions. Secondly, in **Chapter 4**, we resolve the robust localization problem in degenerated environments. An intuitive localizability model is proposed to characterize the degeneracy of given sensor measurements. After analyzing localizability for multiple sensors, we proposed a LiDAR-UWB-Inertial localization algorithm that achieves robust localization in a real degenerated tunnel.

Challenge 2: Large-scale meets high-density.
The hardness of **Challenge 2** lies in that robot mapping for inspection needs to simultaneously

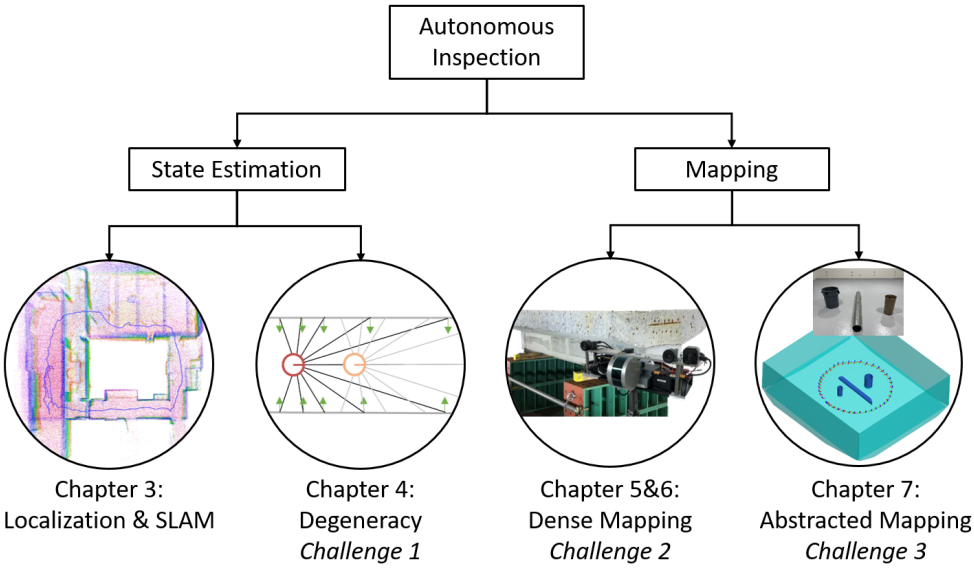


Figure 8.1: The structure of challenges and corresponding chapters.

address the large-scale and the high-resolution problem, which stresses the capacities of state-of-the-art mapping or reconstruction methods. Again, we approach this problem in two steps. Firstly, in **Chapter 5**, we present a joint formulation that relates the raw measurements from LiDAR and cameras. It is shown that the joint optimization successfully leverages LiDAR's long-ranging capacity and the camera's high resolution, and produces dense accurate 3D models. In this step, however, we have not demonstrated large-scale mapping capacity in the real world. Therefore, in **Chapter 6**, we further explore the fusion techniques of LiDAR and camera and offer a series of robustification techniques such that the joint optimization framework can build locally dense and globally consistent models in complex real-world scenarios (e.g., bridge tunnels).

Challenge 3: Redundancy in structured maps.
Challenge 3 raises the question of representing the environment with geometric primitives. Rather than pursuing a precise approximation, our focus in **Chapter 7** is set to looking for suitable representations for the purpose of light-weight SLAM and examine how the abstracted observation model affects the SLAM formulation. Our contribution is a novel decomposed quadrics representation that not only unifies the parameterization of multiple types of geometric primitives, but also discloses the underlying geometric properties. Modeling the observation of abstracted shapes leads to an intuitive interpretation of how abstract 3D shapes constrain the estimation of robot states and also yields a robust formulation of graph optimization based on geometric metrics.

8.2 Lessons Learned

In this section, we offer a high-level discussion of several lessons learned from the subjects of this thesis.

On the sensing degeneracy
Inspired by the intuition that translates the robot sensing model to the force constraints of a rigid object, the proposed localizability model characterizes the environment degeneracy with rich geometric meanings. What we learned from this part of the work is to use the geometric picture of an estimation problem to uncover the underlying structure. We find this thinking quite helpful in understanding other issues in this work, such as the inaccuracies of LiDAR-camera calibration and the degeneration of abstracted shapes.

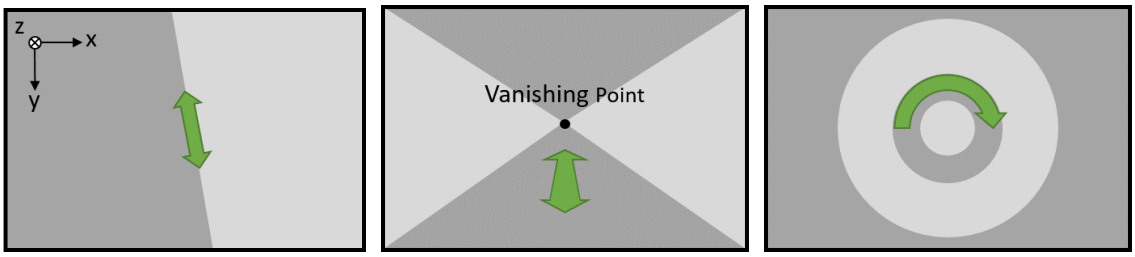
On the joint optimization.
Joint optimization effectively combines the advantages of LiDAR and camera to build dense accurate 3D models, which is a claim that has been mentioned frequently in this thesis. However, batch optimization is never the only block to create the dense mapping system. In fact, to make the joint optimization work, one has to go through a series of preparation steps such as data association, outlier rejection, variable initialization, etc. These pre-processing steps are crucial to ensure the robustness of the whole system and therefore should not be overlooked.

On quadrics and high-level maps
The idea of shape abstraction seems to match how humans interpret the physical world naturally. Using geometric primitives to parameterize the objects in the environment produces highly compact maps and provides strong constraints to estimating robot poses. However, we also learned that high-level maps can be limited in several aspects. Firstly, the shape assumption makes this strategy quite picky about the environment. In less-structured cases, low-level representations or a hybrid representation might make more sense. Secondly, the accuracy of an abstracted back-end heavily depends on the performance of shape extraction. The misalignment of high-level shapes tends to cause more significant errors in the final estimation as the map is now simplified only to contain a few such primitives. Finally, although the uncertainty of high-level shapes is straightforward to interpret (as shown by decomposed quadrics in Chapter 7), accurately characterizing uncertainty in the parameter space remains difficult. The isotropic approximation used by this work is a simple approach but not ideal to precisely capture the effects of observation acted on the final estimation.

8.3 Future Directions

Localizability-aware SLAM and data fusion

Robust SLAM has become one of the most popular trends in the ongoing SLAM research [23]. Aligning with this pursuit, we believe the concept of localizability could be introduced to more aspects of SLAM.



(a) Degeneration in $x-y$ plane. (b) Degeneration in z. (c) Degeneration in roll.

Figure 8.2: Examples of motion degeneration caused by special visual patterns.

One interesting question is how to generalize the localizability to cameras, which, compared to LiDAR, is more widely accessible. We believe the idea of measuring the constraint strength is fundamental enough to be applied for visual data. However, since cameras have different sensing model that captures textural data instead of structural data, the analysis should then be based on texture information. In Fig. 8.2, we show three extreme examples of visual patterns that could cause zero optical flow in some directions, which indicates motion degeneration. We recognize that the examples shown here are almost impossible to happen in the real world, but these basic cases can offer intuitive insights to understand the degeneration of visual data. In practice, it might make more sense to study how the distribution of features affects the motion estimation problem, since many SLAM systems rely on sparse visual features.

Another promising line of research is to use localizability to coordinate estimations based on different sensors. For instance, works in [138] and [159] switch between the LiDAR and cameras based on the confidence of estimations. In this case, localizability could be used as a criterion to determine which source of information to trust.

Primitive-based SLAM and mapping

In Chapter 7, we showed that the concept of unified representation using decomposed quadrics has the benefits of concise and geometrically meaningful formulations. However, in order to use geometric primitives for SLAM, one has to consider the front-end problem where quadrics are first extracted from raw sensing data. Region-growing based [82][146] or RANSAC-based [125][17] methods could be optimized for online usages. Alternatively, Deep Neural Networks (DNN) have shown remarkable performance in detection tasks and therefore could be another direction to explore for online primitive extraction [162] [58].

Another interesting direction of research is to utilize the relative geometric relationships between quadrics. For example, there are attempts to enforce perpendicular relationships between detected planes as in [69] or particular spatial arrangements between primitives as in [87]. It is possible to define similar constraints between decomposed quadrics to improve the map quality.

8.4 Final Remarks

The development of this thesis is motivated by autonomous bridge and tunnel inspection. Our goal is to free human inspectors from labor-intensive tasks using autonomous robots. We believe the technologies developed in this thesis can improve structure monitoring efficiency by facilitating the data collection process and automating the 3D modeling procedures. Additionally, although this work is motivated by inspection, we believe the obtained insights are generalizable to more robotics applications. Finally, even though the technology of robot state estimation and mapping has advanced remarkably in the past decades, robustness is still a major challenge in today's SLAM research. New theories, frameworks, and tools have to be developed to ensure reliable, safe performance in the complex physical world.

Appendix A

Derivation of Localizability

A.1 LiDAR Localizability

Without losing generality, we could always define the map frame to align with the robot body frame. In this way, $(\mathbf{x}, \mathbf{R})$ are small and can be treated as perturbations. Therefore the problem is reduced to evaluate how sensitive is ρ_i w.r.t. the perturbations $(\mathbf{x}, \mathbf{R})$. This assumption allows using the small angle approximation $\mathbf{R} \approx \mathbf{I} + [\boldsymbol{\theta}]_\times$ to find the linearized constraint:

$$\begin{aligned}\bar{\mathcal{C}}(\mathbf{x}, \boldsymbol{\theta}, \rho_i) =& \mathbf{n}_i^T(\mathbf{x} + (\mathbf{I} + [\boldsymbol{\theta}]_\times)\mathbf{r}_i\rho_i) + d_i \\ =& \mathbf{n}_i^T\mathbf{x} + \mathbf{n}_i^T\mathbf{r}_i\rho_i + \mathbf{n}_i^T[\boldsymbol{\theta}]_\times\mathbf{r}_i\rho_i + d_i \\ =& \mathbf{n}_i^T\mathbf{x} + \mathbf{n}_i^T\mathbf{r}_i\rho_i - \mathbf{n}_i^T[\mathbf{r}_i]_\times\boldsymbol{\theta}\rho_i + d_i\end{aligned} \tag{A.1}$$

Then based on the Implicit Function Theorem (IFT), we have

$$\frac{\partial\bar{\mathcal{C}}}{\partial\mathbf{x}}\mathrm{d}\mathbf{x} + \frac{\partial\bar{\mathcal{C}}}{\partial\rho_i}\mathrm{d}\rho_i = 0, \quad \frac{\partial\bar{\mathcal{C}}}{\partial\boldsymbol{\theta}}\mathrm{d}\boldsymbol{\theta} + \frac{\partial\bar{\mathcal{C}}}{\partial\rho_i}\mathrm{d}\rho_i = 0 \tag{A.2}$$

which implies

$$\begin{aligned}\frac{\mathrm{d}\rho_i}{\mathrm{d}\mathbf{x}} &= -\left(\frac{\partial\bar{\mathcal{C}}}{\partial\mathbf{x}}\right)\left(\frac{\partial\bar{\mathcal{C}}}{\partial\rho_i}\right)^{-1} = -\frac{\mathbf{n}_i^T}{\mathbf{n}_i^T\mathbf{r}_i} \\ \frac{\mathrm{d}\rho_i}{\mathrm{d}\boldsymbol{\theta}} &= -\left(\frac{\partial\bar{\mathcal{C}}}{\partial\boldsymbol{\theta}}\right)\left(\frac{\partial\bar{\mathcal{C}}}{\partial\rho_i}\right)^{-1} = -\frac{(\rho_i\mathbf{r}_i \times \mathbf{n}_i)^T}{\mathbf{n}_i^T\mathbf{r}_i}\end{aligned} \tag{A.3}$$

The derivatives are then stacked into matrix $\mathbf{F}$ and $\mathbf{T}$.

A.2 UWB Localizability

The derivation for the UWB is more straightforward. Based on the IFT, we have

$$\frac{\partial\mathcal{C}}{\partial\mathbf{x}}\mathrm{d}\mathbf{x} + \frac{\partial\mathcal{C}}{\partial\gamma}\mathrm{d}\gamma = 0 \tag{A.4}$$

which implies

$$\frac{\mathrm{d}\gamma}{\mathrm{d}\mathbf{x}} = -\left(\frac{\partial \mathcal{C}}{\partial \mathbf{x}}\right)\left(\frac{\partial \mathcal{C}}{\partial \gamma}\right)^{-1} = \frac{\mathbf{x} - \mathbf{x}_a}{\gamma} \tag{A.5}$$

Appendix B

Derivation of Quadrics Jacobians

B.1 Notations

Following the notation convention of the paper, the quadric states in the world frame is represented by $(\mathbf{R}_q, \mathbf{t}_q, \mathbf{s}_q)$, and the robot pose is $(\mathbf{R}_r, \mathbf{t}_r)$. Therefore, the state vector involved in a single observation is $\mathbf{x} = [\mathbf{R}_r, \mathbf{t}_r, \mathbf{R}_q, \mathbf{t}_q, \mathbf{s}_q]$.

To simplify the presentation, we derive the Jacobian matrix based on a single observation, while the complete Jacobian can be constructed by filling in per observation Jacobians.

B.2 Error Function

As presented in the paper, the observation error is given by

$$\begin{aligned} \mathbf{e} &= \begin{pmatrix} \mathbf{e}_\mathbf{R} \\ \mathbf{e}_\mathbf{t} \\ \mathbf{e}_\mathbf{s} \end{pmatrix} \\ &= \begin{pmatrix} \text{diag}(\mathbf{I}^\mathbf{R})\,(\mathbf{V} \otimes \mathbf{\Delta}_\mathbf{R})^T \\ \text{diag}(\mathbf{I}^\mathbf{t})(\mathbf{D}\mathbf{V}^T\mathbf{\Delta}_\mathbf{t} + \mathbf{V}^T\mathbf{l}) \\ \text{diag}(\mathbf{I}^\mathbf{s})(\mathbf{s}^2 - \mathbf{\Lambda}) \end{pmatrix} \in \mathbb{R}^{15} \end{aligned} \tag{B.1}$$

where $\otimes$ means column-wise cross product. $\mathbf{\Delta}_\mathbf{R} = \mathbf{R}_r^T\mathbf{R}_q$ and $\mathbf{\Delta}_\mathbf{t} = \mathbf{R}_r^T(\mathbf{t}_q - \mathbf{t}_r)$ are the rotation and translation of quadrics pose transformed into the robot frame. $\mathbf{\Lambda} = [\lambda_1, \lambda_2, \lambda_3]$ is the vector of eigenvalues which are stored as the diagonal elements of $\mathbf{D}$. Here, $\mathbf{e}_\mathbf{R}$ is a 3×3 matrix and will be vectorized and then stacked into the error vector.

Then we have the derivative $\frac{\partial \mathbf{e}}{\partial \mathbf{x}} \in \mathbb{R}^{15\times 15}$ as

$$\frac{\partial \mathbf{e}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial \mathbf{e}_\mathbf{R}}{\partial \mathbf{R}_r} & \frac{\partial \mathbf{e}_\mathbf{R}}{\partial \mathbf{t}} & \frac{\partial \mathbf{e}_\mathbf{R}}{\partial \mathbf{R}_q} & \frac{\partial \mathbf{e}_\mathbf{R}}{\partial \mathbf{t}_q} & \frac{\partial \mathbf{e}_\mathbf{R}}{\partial \mathbf{s}_q} \\ \frac{\partial \mathbf{e}_\mathbf{t}}{\partial \mathbf{R}_r} & \frac{\partial \mathbf{e}_\mathbf{t}}{\partial \mathbf{t}} & \frac{\partial \mathbf{e}_\mathbf{t}}{\partial \mathbf{R}_q} & \frac{\partial \mathbf{e}_\mathbf{t}}{\partial \mathbf{t}_q} & \frac{\partial \mathbf{e}_\mathbf{t}}{\partial \mathbf{s}_q} \\ \frac{\partial \mathbf{e}_\mathbf{s}}{\partial \mathbf{R}_r} & \frac{\partial \mathbf{e}_\mathbf{s}}{\partial \mathbf{t}} & \frac{\partial \mathbf{e}_\mathbf{t}}{\partial \mathbf{R}_q} & \frac{\partial \mathbf{e}_\mathbf{s}}{\partial \mathbf{t}_q} & \frac{\partial \mathbf{e}_\mathbf{s}}{\partial \mathbf{s}_q} \end{pmatrix} \tag{B.2}$$

Note that the first dimension size 15 is the number of constraints or the error terms. The above Jacobian can be simplified by identifying zero blocks:

$$\frac{\partial \mathbf{e}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial \mathbf{e}_\mathbf{R}}{\partial \mathbf{R}_r} & \mathbf{0} & \frac{\partial \mathbf{e}_\mathbf{R}}{\partial \mathbf{R}_q} & \mathbf{0} & \mathbf{0} \\ \frac{\partial \mathbf{e}_\mathbf{t}}{\partial \mathbf{R}_r} & \frac{\partial \mathbf{e}_\mathbf{t}}{\partial \mathbf{t}} & \frac{\partial \mathbf{e}_\mathbf{t}}{\partial \mathbf{R}_q} & \frac{\partial \mathbf{e}_\mathbf{t}}{\partial \mathbf{t}_q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{e}_\mathbf{s}}{\partial \mathbf{s}_q} \end{pmatrix} \tag{B.3}$$

Now we rewrite error terms explicitly to prepare for the derivation of $\frac{\partial \mathbf{e}}{\partial \mathbf{x}}$:

$$\begin{aligned} \mathbf{e}_\mathbf{R} &= \begin{pmatrix} \cdots \\ [\mathbf{v}_i]_\times \mathbf{R}_r^T \mathbf{R}_q \mathbf{u}_i \\ \cdots \end{pmatrix} \in \mathbb{R}^{9\times 1} \\ \mathbf{e}_\mathbf{t} &= \begin{pmatrix} \cdots \\ \lambda_i \mathbf{u}_i^T \mathbf{R}_q^T (\mathbf{t}_q - \mathbf{t}) + \mathbf{u}_i^T \mathbf{R}_q^T \mathbf{R}_r \mathbf{l} \\ \cdots \end{pmatrix} \in \mathbb{R}^{3\times 1} \\ \mathbf{e}_\mathbf{s} &= \begin{pmatrix} \cdots \\ \mathbf{s}_i^2 - \lambda_i \\ \cdots \end{pmatrix} \in \mathbb{R}^{3\times 1} \end{aligned} \tag{B.4}$$

where $\mathbf{u}_i$ are unit vectors:

$$\mathbf{u}_1 = (1,0,0)^T,\ \mathbf{u}_2 = (0,1,0)^T,\ \mathbf{u}_3 = (0,0,1)^T \tag{B.5}$$

B.3 Linearization

Computing Jacobian involving $\mathbf{R}_r$ and $\mathbf{R}_q$ requires linearization which can be achieved by applying the small angle approximation:

$$\mathbf{R}_r = \mathbf{R}_r \delta\mathbf{R}, \quad \delta\mathbf{R}_r \approx \mathbf{I} + [\mathbf{w}_r]_\times \tag{B.6}$$

and

$$\mathbf{R}_q = \mathbf{R}_q \delta\mathbf{R}_q, \quad \delta\mathbf{R}_q \approx \mathbf{I} + [\mathbf{w}_q]_\times \tag{B.7}$$

where $[\cdot]_\times$ is the skew-symmetric operator:

$$[\mathbf{w}]_\times = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w1 \\ -w_2 & w_1 & 0 \end{bmatrix} \tag{B.8}$$

and $\mathbf{I}$ is the identity matrix. Now we apply the anti-commutative rule of cross product

$$\begin{aligned} \mathbf{a} \times \mathbf{b} &= [\mathbf{a}]_\times \mathbf{b} \\ &= -\mathbf{b} \times \mathbf{a} = -[\mathbf{b}]_\times \mathbf{a} \end{aligned} \quad (\mathbf{a}, \mathbf{b} \in \mathbb{R}^3) \tag{B.9}$$

to linearize the error terms w.r.t. rotation $\mathbf{R}_r$ and $\mathbf{R}_q$:

$$\begin{aligned} \bar{\mathbf{e}}_\mathbf{R}|_{\mathbf{R}_r} &= \begin{pmatrix} \cdots \\ [\mathbf{v}_i]_\times (\mathbf{I} + [\mathbf{w}_r]_\times)^T \mathbf{R}_r^T \mathbf{R}_q \mathbf{u}_i \\ \cdots \end{pmatrix} \\ &= \begin{pmatrix} \cdots \\ [\mathbf{v}_i]_\times (\mathbf{I} - [\mathbf{w}_r]_\times) \mathbf{R}_r^T \mathbf{R}_q \mathbf{u}_i \\ \cdots \end{pmatrix} \\ &= \begin{pmatrix} \cdots \\ [\mathbf{v}_i]_\times \mathbf{R}_r^T \mathbf{R}_q \mathbf{u}_i - [\mathbf{v}_i]_\times [\mathbf{w}_r]_\times \mathbf{R}_r^T \mathbf{R}_q \mathbf{u}_i \\ \cdots \end{pmatrix} \\ &= \begin{pmatrix} \cdots \\ [\mathbf{v}_i]_\times \mathbf{R}_r^T \mathbf{R}_q \mathbf{u}_i + [\mathbf{v}_i]_\times [\mathbf{R}_r^T \mathbf{R}_q \mathbf{u}_i]_\times \mathbf{w}_r \\ \cdots \end{pmatrix} \end{aligned} \tag{B.10}$$

$$\begin{aligned} \bar{\mathbf{e}}_\mathbf{R}|_{\mathbf{R}_q} &= \begin{pmatrix} \cdots \\ [\mathbf{v}_i]_\times \mathbf{R}_r^T \mathbf{R}_q (\mathbf{I} + [\mathbf{w}_q]_\times) \mathbf{u}_i \\ \cdots \end{pmatrix} \\ &= \begin{pmatrix} \cdots \\ [\mathbf{v}_i]_\times \mathbf{R}_r^T \mathbf{R}_q \mathbf{u}_i + [\mathbf{v}_i]_\times \mathbf{R}_r^T \mathbf{R}_q [\mathbf{w}_q]_\times \mathbf{u}_i \\ \cdots \end{pmatrix} \\ &= \begin{pmatrix} \cdots \\ [\mathbf{v}_i]_\times \mathbf{R}_r^T \mathbf{R}_q \mathbf{u}_i - [\mathbf{v}_i]_\times \mathbf{R}_r^T \mathbf{R}_q [\mathbf{u}_i]_\times \mathbf{w}_q \\ \cdots \end{pmatrix} \end{aligned} \tag{B.11}$$

$$\begin{aligned} \bar{\mathbf{e}}_\mathbf{t}|_{\mathbf{R}_r} &= \begin{pmatrix} \cdots \\ \lambda_i \mathbf{u}_i^T \mathbf{R}_q^T (\mathbf{t}_q - \mathbf{t}) + \mathbf{u}_i^T \mathbf{R}_q^T \mathbf{R}_r (\mathbf{I} + [\mathbf{w}_r]_\times) \mathbf{l} \\ \cdots \end{pmatrix} \\ &= \begin{pmatrix} \cdots \\ \lambda_i \mathbf{u}_i^T \mathbf{R}_q^T (\mathbf{t}_q - \mathbf{t}) + \mathbf{u}_i^T \mathbf{R}_q^T \mathbf{R}_r \mathbf{l} + \mathbf{u}_i^T \mathbf{R}_q^T \mathbf{R}_r [\mathbf{w}_r]_\times \mathbf{l} \\ \cdots \end{pmatrix} \\ &= \begin{pmatrix} \cdots \\ \lambda_i \mathbf{u}_i^T \mathbf{R}_q^T (\mathbf{t}_q - \mathbf{t}) + \mathbf{u}_i^T \mathbf{R}_q^T \mathbf{R}_r \mathbf{l} - \mathbf{u}_i^T \mathbf{R}_q^T \mathbf{R}_r [\mathbf{l}]_\times \mathbf{w}_r \\ \cdots \end{pmatrix} \end{aligned} \tag{B.12}$$

$$
\begin{aligned}
\bar{\mathbf{e}}_{\mathbf{t}}|_{\mathbf{R}_q} &= \begin{pmatrix} \cdots \\ \lambda_i \mathbf{u}_i^T (\mathbf{I} + [\mathbf{w}_r]_\times)^T \mathbf{R}_q^T (\mathbf{t}_q - \mathbf{t}) + \mathbf{u}_i^T (\mathbf{I} + [\mathbf{w}_r]_\times)^T \mathbf{R}_q^T \mathbf{R}_r \mathbf{l} \\ \cdots \end{pmatrix} \\
&= \begin{pmatrix} \cdots \\ \lambda_i \mathbf{u}_i^T (\mathbf{I} - [\mathbf{w}_r]_\times) \mathbf{R}_q^T (\mathbf{t}_q - \mathbf{t}) + \mathbf{u}_i^T (\mathbf{I} - [\mathbf{w}_r]_\times) \mathbf{R}_q^T \mathbf{R}_r \mathbf{l} \\ \cdots \end{pmatrix} \\
&= \begin{pmatrix} \cdots \\ -\lambda_i \mathbf{u}_i^T [\mathbf{w}_r]_\times \mathbf{R}_q^T (\mathbf{t}_q - \mathbf{t}) - \mathbf{u}_i^T [\mathbf{w}_r]_\times \mathbf{R}_q^T \mathbf{R}_r \mathbf{l} + \cdots \\ \cdots \end{pmatrix} \\
&= \begin{pmatrix} \cdots \\ \lambda_i \mathbf{u}_i^T [\mathbf{R}_q^T (\mathbf{t}_q - \mathbf{t})]_\times \mathbf{w}_r + \mathbf{u}_i^T [\mathbf{R}_q^T \mathbf{R}_r \mathbf{l}]_\times \mathbf{w}_r + \cdots \\ \cdots \end{pmatrix}
\end{aligned}
\tag{B.13}
$$

In the linearized cost function $\mathbf{e}_{\mathbf{t}}|_{\mathbf{R}_q}$, constant terms not related to $\mathbf{x}$ are omitted. The error terms are linearized in that now they are linear w.r.t. $\mathbf{w}_r$ and $\mathbf{w}_q$.

B.4 Jacobians

Finally, from the above linearized equations, we can have the Jacobian blocks:

$$
\begin{aligned}
\frac{\partial \mathbf{e}_{\mathbf{R}}}{\partial \mathbf{R}_r} &= \begin{pmatrix} \cdots \\ [\mathbf{v}_i]_\times [\mathbf{R}_r^T \mathbf{R}_q \mathbf{u}_i]_\times \\ \cdots \end{pmatrix} \\
\frac{\partial \mathbf{e}_{\mathbf{t}}}{\partial \mathbf{R}_r} &= \begin{pmatrix} \cdots \\ -\mathbf{u}_i^T \mathbf{R}_q^T \mathbf{R}_r [\mathbf{l}]_\times \\ \cdots \end{pmatrix} \\
\frac{\partial \mathbf{e}_{\mathbf{R}_r}}{\partial \mathbf{R}_q} &= \begin{pmatrix} \cdots \\ -[\mathbf{v}_i]_\times \mathbf{R}_r^T \mathbf{R}_q [\mathbf{u}_i]_\times \\ \cdots \end{pmatrix} \\
\frac{\partial \mathbf{e}_{\mathbf{t}}}{\partial \mathbf{R}_q} &= \begin{pmatrix} \cdots \\ \lambda_i \mathbf{u}_i^T [\mathbf{R}_q^T (\mathbf{t}_q - \mathbf{t})]_\times + \mathbf{u}_i^T [\mathbf{R}_q^T \mathbf{R}_r \mathbf{l}]_\times \\ \cdots \end{pmatrix}
\end{aligned}
\tag{B.14}
$$

As to Jacobian w.r.t. translation and scale, it is straight forward:

$$\begin{aligned}
\frac{\partial \mathbf{e_t}}{\partial \mathbf{t}_r} &= \begin{pmatrix} \cdots \\ -\lambda_i \mathbf{u}_i^T \mathbf{R}_q^T \\ \cdots \end{pmatrix} \\
\frac{\partial \mathbf{e_t}}{\partial \mathbf{t}_q} &= \begin{pmatrix} \cdots \\ \lambda_i \mathbf{u}_i^T \mathbf{R}_q^T \\ \cdots \end{pmatrix} \\
\frac{\partial \mathbf{e_s}}{\partial \mathbf{s}_q} &= \begin{pmatrix} \cdots \\ 2\mathbf{s}_i \\ \cdots \end{pmatrix}
\end{aligned} \tag{B.15}$$

The computed Jacobian blocks can then be filled into (B.3) and finally used to construct the complete Jacobian matrix used in the LM method (see Algorithm 2).

Bibliography

[1] Ahmed Abdelhafiz, Björn Riedel, and Wolfgang Niemeier. Towards a 3d true colored space by the fusion of laser scanner point cloud and digital photos. In *Proceedings of the ISPRS Working Group V/4 Workshop (3D-ARCH*. Citeseer, 2005. 5.2

[2] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`. 2.3.1, 6.3.6

[3] Ali Agha, Kyohei Otsu, Benjamin Morrell, David D Fan, Rohan Thakker, Angel Santamaria-Navarro, Sung-Kyun Kim, Amanda Bouman, Xianmei Lei, Jeffrey Edlund, et al. Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge. *arXiv preprint arXiv:2103.11470*, 2021. 4.2

[4] Stéphane Allaire, Valérie Burdin, J-J Jacq, Gregory Moineau, Eric Stindel, and Ch Roux. Robust quadric fitting and mensuration comparison in a mapping space applied to 3d morphological characterization of articular surfaces. In *2007 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 972–975, 2007. 7.3.4

[5] Stéphane Allaire, Jean-José Jacq, Valérie Burdin, Christian Roux, and Christine Couture. Type-constrained robust fitting of quadrics with application to the 3d morphological characterization of saddle-shaped articular surfaces. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007. 7.4.5

[6] Irvin Aloise, Bartolomeo Della Corte, Federico Nardi, and Giorgio Grisetti. Systematic handling of heterogeneous geometric primitives in graph-slam optimization. *IEEE Robotics and Automation Letters*, 4(3):2738–2745, 2019. 7.2

[7] Howard Anton and Chris Rorres. *Elementary linear algebra: applications version*. John Wiley & Sons, 2013. 7.1

[8] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987. 2.2.5

[9] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, 2006. 7.3.2

[10] Yashar Balazadegan Sarvrood, Siavash Hosseinyalamdary, and Yang Gao. Visual-lidar odometry aided by reduced imu. *ISPRS International Journal of Geo-Information*, 5(1): 3, 2016. 6.2

[11] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A

randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 2.3.2

[12] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006. 2.3.3, 5.3.2

[13] Paul Beardsley, Phil Torr, and Andrew Zisserman. 3d model acquisition from extended image sequences. In *European conference on computer vision*, pages 683–695. Springer, 1996. 6.2

[14] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992. 2.2.5, 5.3.3

[15] Simone Bianco, Gianluigi Ciocca, and Davide Marelli. Evaluating the performance of structure from motion pipelines. *Journal of Imaging*, 4(8):98, 2018. 6.2

[16] Tolga Birdal, Benjamin Busam, Nassir Navab, Slobodan Ilic, and Peter Sturm. A minimalist approach to type-agnostic detection of quadrics in point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3530–3540, 2018. 7.3.2

[17] Tolga Birdal, Benjamin Busam, Nassir Navab, Slobodan Ilic, and Peter Sturm. Generic primitive detection in point clouds using novel minimal quadric fits. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1333–1347, 2019. 8.3

[18] Yunsu Bok, Dong-Geol Choi, and In So Kweon. Sensor fusion of cameras and a laser for city-scale 3d reconstruction. *Sensors*, 14(11):20882–20909, 2014. 5.2

[19] Silvere Bonnabel, Martin Barczyk, and François Goulette. On the covariance of icp-based scan-matching techniques. In *2016 American Control Conference (ACC)*, pages 5498–5503. IEEE, 2016. 4.2

[20] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 5.4.1

[21] Jonathan Brookshire and Seth Teller. Extrinsic calibration from per-sensor egomotion. *Robotics: Science and Systems VIII*, pages 504–512, 2013. 5.2

[22] Adam Bry, Abraham Bachrach, and Nicholas Roy. State estimation for aggressive flight in gps-denied environments using onboard sensing. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1–8. IEEE, 2012. 2.3.1, 2.3.3, 3.2, 4.4

[23] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016. 8.3

[24] Jose A Castellanos, JMM Montiel, José Neira, and Juan D Tardós. The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–952, 1999. 7.2

[25] Robert Oliver Castle, Darren J Gawley, Georg Klein, and David W Murray. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages

4102–4107. IEEE, 2007. 2.3.3

[26] Andrea Censi. An accurate closed-form estimate of ICP's covariance. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3167–3172. IEEE, 2007. 4.2

[27] Andrea Censi. On achievable accuracy for range-finder localization. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4170–4175. IEEE, 2007. 4.2

[28] Andrea Censi. On achievable accuracy for pose tracking. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1–7. IEEE, 2009. 4.2

[29] Dan Cernea. OpenMVS: Multi-view stereo reconstruction library. 2020. URL `https://cdcseacave.github.io/openMVS`. 6.4.5

[30] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguere, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE, 2019. 7.1

[31] Jian Cheng, Cong Leng, Jiaxiang Wu, Hainan Cui, and Hanqing Lu. Fast and accurate image matching with cascade hashing for 3d reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2014. 6.3.1

[32] Hugo Courtois and Nabil Aouf. Fusion of stereo and lidar data for dense depth map computation. In *Research, Education and Development of Unmanned Aerial Systems (RED-UAS), 2017 Workshop on*, pages 186–191. IEEE, 2017. 5.2

[33] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996. 2.3.3

[34] César Debeunne and Damien Vivet. A review of visual-lidar fusion based simultaneous localization and mapping. *Sensors*, 20(7):2068, 2020. 2.3.2

[35] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012. 2.2.3, 2.3.1, 7.2, 7.4.4

[36] Frank Dellaert, Steven M Seitz, Charles E Thorpe, and Sebastian Thrun. Structure from motion without correspondence. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 2, pages 557–564. IEEE, 2000. 6.2

[37] Frank Dellaert, Michael Kaess, et al. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017. 2.2.3, 2

[38] Mahmut Demir and Kikuo Fujimura. Robust localization with low-mounted multiple lidars in urban environments. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3288–3293. IEEE, 2019. 3.2

[39] Albert Diosi and Lindsay Kleeman. Uncertainty of line segments extracted from static SICK PLS laser scans. In *SICK PLS laser. In Australiasian Conference on Robotics and Automation*, 2003. 4.2

[40] Jennifer Dolson, Jongmin Baek, Christian Plagemann, and Sebastian Thrun. Upsampling range data in dynamic environments. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1141–1148. IEEE, 2010. 5.2

[41] Zhen Dong, Bisheng Yang, Yuan Liu, Fuxun Liang, Bijun Li, and Yufu Zang. A novel binary shape context for 3d local surface description. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130:431–452, 2017. 5.3.3

[42] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the rgb-d slam system. In *2012 IEEE International Conference on Robotics and Automation*, pages 1691–1696. IEEE, 2012. 2.3.2

[43] Alexandre Eudes and Maxime Lhuillier. Error propagations for local bundle adjustment. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2411–2418. IEEE, 2009. 4.2

[44] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2.2.5

[45] Andrew W Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In *European conference on computer vision*, pages 311–326. Springer, 1998. 6.2

[46] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Georgia Institute of Technology, 2015. 2.3.1

[47] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999(343-349):2–2, 1999. 3.2

[48] Andrea Garulli, Antonio Giannitrapani, Andrea Rossi, and Antonio Vicino. Mobile robot slam for line-based environment representation. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 2041–2046. IEEE, 2005. 7.1

[49] Abel Gawel, Carlo Del Don, Roland Siegwart, Juan Nieto, and Cesar Cadena. X-view: Graph-based semantic multi-view localization. *IEEE Robotics and Automation Letters*, 3 (3):1687–1694, 2018. 7.1

[50] Andreas Geiger, Frank Moosmann, Ömer Car, and Bernhard Schuster. Automatic camera and range sensor calibration using a single shot. In *2012 IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, pages 3936–3943. IEEE, 2012. 5.2

[51] Natasha Gelfand, Leslie Ikemoto, Szymon Rusinkiewicz, and Marc Levoy. Geometrically stable sampling for the icp algorithm. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 260–267. IEEE, 2003. 4.2, 4.6

[52] Patrick Geneva, Kevin Eckenhoff, Yulin Yang, and Guoquan Huang. Lips: Lidar-inertial 3d plane slam. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 123–130, 2018. 7.2

[53] Ruben Gomez-Ojeda, Francisco-Angel Moreno, David Zuniga-Noël, Davide Scaramuzza, and Javier Gonzalez-Jimenez. Pl-slam: A stereo slam system through the combination of points and line segments. *IEEE Transactions on Robotics*, 35(3):734–746, 2019. 7.1

[54] Xiaojin Gong, Ying Lin, and Jilin Liu. 3d lidar-camera extrinsic calibration using an arbitrary trihedron. *Sensors*, 13(2):1902–1918, 2013. 5.2

[55] Johannes Graeter, Alexander Wilczynski, and Martin Lauer. Limo: Lidar-monocular visual odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7872–7879. IEEE, 2018. 6.2

[56] W Nicholas Greene and Nicholas Roy. Flame: Fast lightweight mesh estimation using variational smoothing on delaunay graphs. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4696–4704. IEEE, 2017. 2.3.3

[57] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010. 7.4.4

[58] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. Pcpnet learning local shape properties from raw point clouds. In *Computer Graphics Forum*, volume 37, pages 75–85. Wiley Online Library, 2018. 8.3

[59] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3d object recognition in cluttered scenes with local surface features: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2270–2287, 2014. 2.2.5

[60] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89, 2016. 5.3.3

[61] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988. 2.3.3

[62] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 6.3.5

[63] Sebastian Hening, Corey A Ippolito, Kalmanje S Krishnakumar, Vahram Stepanyan, and Mircea Teodorescu. 3d lidar slam integration with gps/ins for uavs in urban gps-degraded environments. In *AIAA Information Systems-AIAA Infotech@ Aerospace*, page 0448. 2017. 2.3.1, 3.2

[64] Robert Hermann and Arthur Krener. Nonlinear controllability and observability. *IEEE Transactions on automatic control*, 22(5):728–740, 1977. 4.1

[65] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE, 2016. 2.2.3, 2.3.1, 3.4

[66] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2008. 5.3.2

[67] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Bur-

gard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. doi: 10.1007/s10514-012-9321-0. URL `http://octomap.github.com`. Software available at `http://octomap.github.com`. 3.5.2

[68] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013. 2.3.3, 3.4, 6.3.3

[69] Ming Hsiao, Eric Westman, and Michael Kaess. Dense planar-inertial slam with structural constraints. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6521–6528. IEEE, 2018. 2.3.3, 8.3

[70] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis. Observability-based rules for designing consistent ekf slam estimators. *The International Journal of Robotics Research*, 29(5):502–528, 2010. 3.6

[71] Daniel Huber, Takeo Kanade, et al. Integrating lidar into stereo for fast and improved disparity computation. In *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 405–412. IEEE, 2011. 5.2

[72] Ryoichi Ishikawa, Takeshi Oishi, and Katsushi Ikeuchi. Lidar and camera calibration using motions estimated by sensor fusion odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7342–7349. IEEE, 2018. 5.2

[73] Andrew Johnson, Patrick Leger, Regis Hoffman, Martial Hebert, and James Osborn. 3-d object modeling and recognition for telerobotic manipulation. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 1, pages 103–110. IEEE, 1995. 7.1

[74] Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004. 2.2.2

[75] Michael Kaess. Simultaneous localization and mapping with infinite planes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611, 2015. 2.3.3, 7.1, 7.2

[76] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008. 2.2.3, 2.3.1

[77] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012. 2.2.3, 2.3.1

[78] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960. 2.2.2

[79] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM transactions on graphics (TOG)*, 22(3):954–961, 2003. 7.3.2

[80] Georg Klein and David Murray. Improving the agility of keyframe-based slam. In *European Conference on Computer Vision*, pages 802–815, 2008. 7.2

[81] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *Robotics and Automation*

(ICRA), 2011 IEEE International Conference on, pages 3607–3613. IEEE, 2011. 2.2.3, 7.2

[82] Florent Lafarge and Clément Mallet. Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. *International journal of computer vision*, 99(1):69–85, 2012. 7.3.2, 8.3

[83] Dana Lahat, Tülay Adali, and Christian Jutten. Multimodal data fusion: an overview of methods, challenges, and prospects. *Proceedings of the IEEE*, 103(9):1449–1477, 2015. 2.3.2

[84] Kwang Wee Lee, W Sardha Wijesoma, and Javier Ibanez Guzman. On the observability and observability analysis of slam. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3569–3574. IEEE, 2006. 3.6, 4.1

[85] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015. 6.2

[86] Jesse Levinson and Sebastian Thrun. Automatic online calibration of cameras and lasers. In *Robotics: Science and Systems*, volume 2, 2013. 2.3.2, 5.2, 5.9e, 5.9, 5.10, 5.4.3, 5.4.5

[87] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J Mitra. Globfit: Consistently fitting primitives by discovering global relations. In *ACM SIGGRAPH 2011 papers*, pages 1–12. 2011. 8.3

[88] Katherine Liu, Kyel Ok, William Vega-Brown, and Nicholas Roy. Deep inference for covariance estimation: Learning gaussian noise models for state estimation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1436–1443. IEEE, 2018. 4.2

[89] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 6.3.1

[90] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28 (1):61–76, 2011. 6.2

[91] Simon Lynen, Markus W Achtelik, Stephan Weiss, Margarita Chli, and Roland Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *2013 IEEE/RSJ international conference on intelligent robots and systems*, pages 3923–3929. IEEE, 2013. 2.3.1, 3.2

[92] Will Maddern and Paul Newman. Real-time probabilistic fusion of sparse 3d lidar and dense stereo. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2181–2188. IEEE, 2016. 2.3.2, 5.2

[93] Venkatesh K Madyastha, Vishal C Ravindra, Srinath Mallikarjunan, and Anup Goyal. Extended kalman filter vs. error state kalman filter for aircraft attitude estimation. In *AIAA GNC*, 2011. 3.3

[94] Robert Maier, Jürgen Sturm, and Daniel Cremers. Submap-based bundle adjustment for 3d reconstruction from rgb-d data. In *German Conference on Pattern Recognition*, pages

54–65. Springer, 2014. 6.5

[95] Ondrej Miksik, Yousef Amar, Vibhav Vineet, Patrick Pérez, and Philip HS Torr. Incremental dense multi-modal 3d scene reconstruction. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 908–915. IEEE, 2015. 5.2

[96] Roger Mohr, Long Quan, and Françoise Veillon. Relative 3d reconstruction using multiple uncalibrated images. *The International Journal of Robotics Research*, 14(6):619–632, 1995. 6.2

[97] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978. 2.2.3, 7.4.4

[98] Pierre Moulon, Pascal Monasse, Renaud Marlet, and Others. Openmvg. `https://github.com/openMVG/openMVG`. 6.3.3, 6.4.3

[99] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Adaptive structure from motion with a contrario model estimation. In *Asian Conference on Computer Vision*, pages 257–270. Springer, 2012. 6.2

[100] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3248–3255, 2013. 6.2

[101] Wassim Moussa, Mohammed Abdel-Wahab, and Dieter Fritsch. Automatic fusion of digital images and laser scanner data for heritage preservation. In *Euro-Mediterranean Conference*, pages 76–85. Springer, 2012. 5.2

[102] Richard M Murray. *A mathematical introduction to robotic manipulation*. CRC press, 2017. 3.3.1, 4.3.1

[103] Federico Nardi, Bartolomeo Della Corte, and Giorgio Grisetti. Unified representation and registration of heterogeneous sets of geometric primitives. *IEEE Robotics and Automation Letters*, 4(2):625–632, 2019. 7.1, 7.2

[104] Wolfang Neubauer, Michael Doneus, Nikolaus Studnicka, and Jeffrey Riegl. Combined high resolution laser scanning and photogrammetrical documentation of the pyramids at giza. In *CIPA XX International Symposium*, pages 470–475. Citeseer, 2005. 5.2

[105] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011. 2.3.2

[106] Kai Ni, Drew Steedly, and Frank Dellaert. Out-of-core bundle adjustment for large-scale 3d reconstruction. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 6.5

[107] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, 4(1):1–8, 2018. 2.3.3, 7.1, 7.2, 7.4.5

[108] Kyel Ok, Katherine Liu, Kris Frey, Jonathan P How, and Nicholas Roy. Robust object-based slam for high-speed autonomous navigation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 669–675. IEEE, 2019. 2.3.3

[109] 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. 5.2

[110] John Papadakis, Andrew Willis, and Jamie Gantert. Rgbd-sphere slam. In *SoutheastCon 2018*, pages 1–5, 2018. 7.2

[111] Jinhyung Park, Yi-Chun Chen, Yu-Jhe Li, and Kris Kitani. Crack detection and refinement via deep reinforcement learning. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 529–533. IEEE, 2021. 6.4.5

[112] Mrinal K Paul, Kejian Wu, Joel A Hesch, Esha D Nerurkar, and Stergios I Roumeliotis. A comparative analysis of tightly-coupled monocular, binocular, and stereo vins. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 165–172. IEEE, 2017. 6.2

[113] Sylvain Petitjean. A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys (CSUR)*, 34(2):211–262, 2002. 7.1

[114] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004. 6.2

[115] Albert Pumarola, Alexander Vakhitov, Antonio Agudo, Alberto Sanfeliu, and Francese Moreno-Noguer. Pl-slam: Real-time monocular visual slam with points and lines. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 4503–4508. IEEE, 2017. 2.3.3, 7.2

[116] Zoltan Pusztai and Levente Hajder. Accurate calibration of lidar-camera systems using ordinary boxes. 2017. 5.2

[117] Chao Qin, Haoyang Ye, Christian E Pranata, Jun Han, Shuyang Zhang, and Ming Liu. Lins: A lidar-inertial state estimator for robust and efficient navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8899–8906. IEEE, 2020. 3.2

[118] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696. IEEE, 2020. 2.3.3

[119] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans. In *Robotics: Science and Systems (RSS)*, 2020. 2.3.3

[120] Stergios I Roumeliotis, Gaurav S Sukhatme, and George A Bekey. Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1656–1663. IEEE, 1999. 3.2

[121] Nicholas Roy and Sebastian Thrun. Coastal navigation with mobile robots. In *Advances in Neural Information Processing Systems*, pages 1043–1049, 2000. 4.2

[122] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons, 2016. 2.2.2

[123] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. 2.2.5

[124] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1352–1359, 2013. 2.3.3, 7.2

[125] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer Graphics Forum*, volume 26, pages 214–226. Wiley Online Library, 2007. 7.3.2, 7.5.3, 8.3

[126] Sebastian Schneider, Thorsten Luettel, and Hans-Joachim Wuensche. Odometry-based online extrinsic sensor calibration. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1287–1292. IEEE, 2013. 5.2

[127] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016. 2.3.2, 6.2, 6.3.5

[128] Johannes L Schönberger, Marc Pollefeys, Andreas Geiger, and Torsten Sattler. Semantic visual localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6896–6906, 2018. 7.1

[129] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018. 2.3.1

[130] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142. IEEE, 2020. 2.2.3, 2.3.1

[131] Weizhao Shao, Srinivasan Vijayarangan, Cong Li, and George Kantor. Stereo visual inertial lidar simultaneous localization and mapping. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2019. 2.3.2, 6.2

[132] Young-Sik Shin, Yeong Sang Park, and Ayoung Kim. Direct visual slam using sparse depth for camera-lidar system. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. 6.2

[133] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846. ACM, 2006. 6.2

[134] Joan Sola. Quaternion kinematics for the error-state kf. *Laboratoire d'Analyse et d'Architecture des Systemes-Centre national de la recherche scientifique (LAAS-CNRS), Toulouse, France, Tech. Rep*, 2012. 4.4

[135] Takayuki Sugiura, Akihiko Torii, and Masatoshi Okutomi. 3d surface reconstruction from point-and-line cloud. In *2015 International Conference on 3D Vision*, pages 264–272. IEEE, 2015. 4.2

[136] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J Taylor, and Vijay Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, 3(2):965–972, 2018. 6.2

[137] Chris Sweeney. Theia multiview geometry library: Tutorial & reference. `http://theia-sfm.org`. 6.2

[138] Andrea Tagliabue, Jesus Tordesillas, Xiaoyi Cai, Angel Santamaria-Navarro, Jonathan P How, Luca Carlone, and Ali-akbar Agha-mohammadi. Lion: Lidar-inertial observability-aware navigator for vision-denied environments. In *International Symposium on Experimental Robotics*, pages 380–390. Springer, 2020. 4.2, 8.3

[139] Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. Point-plane slam for hand-held 3d sensors. In *2013 IEEE International Conference on Robotics and Automation*, pages 5182–5189, 2013. 7.1, 7.2

[140] Gabriel Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(11):1115–1138, 1991. 7.5.3

[141] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005. 2.2.2, 2.2.2, 3.3.2

[142] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999. 2.2.4

[143] Roger Y Tsai and Reimar K Lenz. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. *IEEE Transactions on robotics and automation*, 5(3): 345–358, 1989. 5.2, 5.3.6, 6.4.4

[144] Florian Tschopp, Juan Nieto, Roland Y Siegwart, and Cesar Dario Cadena Lerma. Superquadric object representation for optimization-based semantic slam. 2021. 2.3.3, 7.2

[145] Vladyslav Usenko, Jakob Engel, Jörg Stückler, and Daniel Cremers. Direct visual-inertial odometry with stereo cameras. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1885–1892. IEEE, 2016. 6.2

[146] Narunas Vaskevicius, Kaustubh Pathak, Razvan Pascanu, and Andreas Birk. Extraction of quadrics from noisy point-clouds using a sensor noise model. In *2010 IEEE International Conference on Robotics and Automation*, pages 3466–3471. IEEE, 2010. 8.3

[147] Pavel Vechersky, Mark Cox, Paulo Borges, and Thomas Lowe. Colourising point clouds using independent cameras. *IEEE Robotics and Automation Letters*, 3(4):3575–3582, 2018. 5.2

[148] William Vega-Brown, Abraham Bachrach, Adam Bry, Jonathan Kelly, and Nicholas Roy. Cello: A fast algorithm for covariance estimation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3160–3167. IEEE, 2013. 4.2

[149] Martin Vel'as, Michal Španěl, Zdeněk Materna, and Adam Herout. Calibration of rgb camera with velodyne lidar. 2014. 5.2

[150] Changchang Wu et al. Visualsfm: A visual structure from motion system. 2011. 6.2

[151] Shichao Yang and Sebastian Scherer. Cubeslam: Monocular 3-d object slam. *IEEE Transactions on Robotics*, 35(4):925–938, 2019. 2.3.3, 7.1, 7.2

[152] Yulin Yang, Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang. Visual-inertial odometry with point and line features. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2447–2454. IEEE, 2019. 4.2

[153] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems Conference (RSS)*, pages 109–111, 2014. 2.3.1, 2.3.2, 2.3.3

[154] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181. IEEE, 2015. 5.2, 6.2

[155] Ji Zhang and Sanjiv Singh. Enabling aggressive motion estimation at low-drift and accurate mapping in real-time. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5051–5058. IEEE, 2017. 4.2

[156] Ji Zhang, Michael Kaess, and Sanjiv Singh. Real-time depth enhanced monocular odometry. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4973–4980. IEEE, 2014. 2.3.2

[157] Ji Zhang, Michael Kaess, and Sanjiv Singh. On degeneracy of optimization-based state estimation problems. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 809–816. IEEE, 2016. 4.2

[158] Jiyuan Zhang, Gang Zeng, and Hongbin Zha. Structure-aware slam with planes and lines in man-made environment. *Pattern Recognition Letters*, 127:181–190, 2019. 2.3.3, 7.2

[159] Shibo Zhao, Hengrui Zhang, Peng Wang, Lucas Nogueira, and Sebastian Scherer. Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021. 2.2.3, 2.3.1, 7.5.2, 7.5.3, 8.3

[160] Hang Zhou, Dongxiang Zhou, Keju Peng, Weihong Fan, and Yunhui Liu. Slam-based 3d line reconstruction. In *2018 13th World Congress on Intelligent Control and Automation (WCICA)*, pages 1148–1153. IEEE, 2018. 4.2

[161] Lipu Zhou, Zimo Li, and Michael Kaess. Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5562–5569. IEEE, 2018. 5.2

[162] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 900–909, 2017. 8.3

[163] Xingxing Zuo, Xiaojia Xie, Yong Liu, and Guoquan Huang. Robust visual slam with point and line features. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1775–1782, 2017. 7.2