Sliding-Window-Based Tightly-Coupled LIDAR-Inertial-Visual Odometry and Mapping

by

Kyungmin Jung

Department of Mechanical Engineering
McGill University, Montreal
August 2021

A thesis submitted to McGill University
in partial fulfillment of the requirements of the degree of
Master of Science

Supervisor:

Professor James Richard Forbes

Kyungmin Jung
kyungmin.jung@mail.mcgill.ca

Abstract

This thesis investigates the design of an autonomous navigation algorithm in GPS-denied environments. The performance of the current navigation solutions is dictated by the change in the operating environment. As a result, in the past two decades, various Simultaneous Localization and Mapping (SLAM) formulations have been proposed to adapt various operating conditions and improve robustness. Multi-sensor fusion has been one of the solutions. Every sensor has its own strengths and weaknesses to a given environment. By using multiple sensors, the weaknesses of one sensor can be counterbalanced by strengths of others. To this end, the proposed approach presents a novel tightly-coupled LIDAR-Inertial-Visual Odometry in a sliding window framework. Inertial measurement unit (IMU) provides high frequency inertial measurements, LIDAR provides an accurate map of surrounding, and visual odometry provides detailed information about the scene. Current state-of-the-art multi-sensor fusion works around coupling various odometry solutions in a pose-graph formulation. Such method is referred as loosely-coupled system where each sensor provides the robot pose estimate with uncertainty. However, a tightly-coupled system where all the sensor measurements are optimized in a factor-graph has been shown to achieve better accuracy and robustness compared to a loosely-coupled framework. In particular, the vision community has seen major improvements since adopting a tightly-coupled sensor fusion framework. This thesis investigates the performance of a tightly-coupled multi-sensor fusion in an optimization level. Further, inspired by the recently developed invariant extended Kalman filter (IEKF), the proposed odometry solution provides an accurate uncertainty estimate using matrix Lie group theory. The pipeline is validated, tested, and compared to the state-of-the-art SLAM algorithms.

Résumé

Cette thèse étudie la conception d'un algorithme de navigation autonome dans des environnements sans GPS. Les performances des solutions de navigation actuelles sont dictées par l'évolution de l'environnement d'exploitation. En conséquence, au cours des deux dernières décennies, diverses formulations de localisation et de cartographie simultanées (SLAM) ont été proposées pour s'adapter à diverses conditions d'exploitation et améliorer la robustesse. La fusion multicapteur est l'une de ces formulations. Chaque capteur a ses propres forces et faiblesses dans un environnement donné. En utilisant plusieurs capteurs, les faiblesses d'un capteur peuvent être contrebalancées par les forces des autres. À cette fin, l'approche proposée présente une nouvelle odométrie LIDAR-inertielle-visuelle étroitement couplée dans un cadre de fenêtre coulissante. L'unité de mesure inertielle (IMU) fournit des mesures inertielles à haute fréquence, un LIDAR fournit une carte précise de l'environnement et l'odométrie visuelle fournit des informations détaillées sur la scène. La fusion multicapteur à la pointe de la technologie fonctionne en couplant diverses solutions d'odométrie dans une formulation de graphe de pose. Ce type de système est dit lâchement couplé. Chaque capteur fournit une estimation de la pose du robot avec une incertitude. Cependant, il a été démontré qu'un système étroitement couplé où toutes les mesures des capteurs sont optimisées dans un graphe factoriel permet d'obtenir une meilleure précision et robustesse par rapport à un cadre lâchement couplé. En particulier, depuis l'adoption d'un cadre de fusion de capteurs étroitement couplé, les algorithmes de vision ont bénéficié d'améliorations majeures. Cette thèse étudie la performance d'une fusion multicapteur étroitement couplée à un niveau d'optimisation. De plus, inspirée par le filtre de Kalman étendu invariant récemment développé (IEKF), la solution d'odométrie proposée fournit une estimation précise de l'incertitude en utilisant la théorie des groupes de Lie matricielle. Le tout est validé, testé et comparé aux algorithmes SLAM de pointe.

Acknowledgements

First, I would like to thank Professor James Richard Forbes for his invaluable advice and support throughout my Masters program. His passion and extensive knowledge in robotics make him an exemplary supervisor, challenging me every step towards the excellence. His enlightenment guided me through all the difficulties and hardships I had in research. I was extremely fortunate to work with him. I extend these thanks to all of the DECAR systems group members, current and former. To Mitchell Cohen, I am grateful for your development of Visual-Inertial Odometry front-end allowing me to focus on other priorities. Thanks to Jonathan Arsenault for proof reading this thesis document.

This research was supported by Mitacs through the Mitacs Accelerate program. Thanks to ARA Robotique, I had access to many resources and opportunities to learn hardware. Special thanks to Guillaume Charland-Arcand, Pascal Chiva-Bernard and the rest of the colleagues at ARA Robotique for their resources and opportunity. In particular, I would like to thank Duowen Qian for enduring all of my questions on implementation and coding. I would have not been able to achieve such an accomplishment without all their support.

Last but not least, I would like to thank my friends and family for their support in every way.

Preface

The contributions of this thesis that are original to the author's knowledge are as follows.

- Chapter 4
  - Tightly-coupling the LIDAR, IMU and monocular camera sensors on an optimization level in a sliding window framework.

All text, plots, figures and results in this thesis are produced by Kyungmin Jung. The monocular VIO formulation is based on the [1] where the depth features are optimized, and the LIO derivation is based on [2] where the in-between LIDAR point correspondences are optimized. Kyungmin Jung tightly-coupled two odometry solutions in the optimization level leading to a novel tightly-coupled LIDAR-Inertial-Visual Odometry.

Table of Contents

List of Figures

Figure

List of Tables

List of Abbreviations

AUV Autonomous Underwater Vehicle

UAV Unmanned Aerial Vehicle

GNC Guidance Navigation & Control

LIDAR Light Detection and Ranging

IMU Inertial Measurement Unit

SLAM Simultaneous Localization and Mapping

VIO Visual-Inertial Odometry

LIO LIDAR-Inertial Odometry

LIVO LIDAR-Inertial-Visual Odometry and Mapping

KF Kalman Filter

EKF Extended Kalman Filter

IEKF Invariant Extended Kalman Filter

SWF Sliding Window Filter

MAP Maximum A Posteriori

BCH Baker-Campbell-Hausdorff

PSD Power Spectral Density

RMI Relative Motion Increment

ICP Iterative Closest Point

KLT Kanada-Lucas-Tomasi

RANSAC Random Sample Consensus

GN Gauss-Newton

LM Levenberg-Marquardt

List of Symbols

$\lVert\cdot\rVert$	the Euclidian norm of a physical vector
$\mathbb{R}^n$	the vector space of real n-dimensional vectors
$\mathbb{R}^{m\times n}$	the space of real $m \times n$-dimensional matrices
$(\cdot)^{\mathsf{T}}$	transpose
$(\cdot)^{\times}$	cross operator for $\mathfrak{so}(3)$
$(\cdot)^{\wedge}$	operator mapping an element of $\mathbb{R}^d$ to $\mathfrak{g}$
$(\cdot)^{\vee}$	operator mapping an element of $\mathfrak{g}$ to $\mathbb{R}^d$
$\mathbf{0}$	zero matrix
$\mathbf{1}$	identity matrix
$\underrightarrow{\mathcal{F}}_i$	reference frame
$\underrightarrow{r}$	a physical vector
$\underrightarrow{r}^{zw}$	the position of point z relative to point w
$\underrightarrow{r}^{\bullet a}$	the time derivative of $\underrightarrow{r}$ with respect to $\underrightarrow{\mathcal{F}}_a$
$\underrightarrow{r}^{zw^{\bullet a}} = \underrightarrow{v}^{zw/a}$	velocity of point z relative to point w with respect to $\underrightarrow{\mathcal{F}}_a$
$\mathbf{r}_a$	the physical vector $\underrightarrow{r}$ resolved in $\underrightarrow{\mathcal{F}}_a$
$\mathbf{C}_{ab}$	a DCM parameterizing the attitude of $\underrightarrow{\mathcal{F}}_a$ relative to $\underrightarrow{\mathcal{F}}_b$
$\underrightarrow{\omega}^{ba}$	angular velocity of $\underrightarrow{\mathcal{F}}_b$ relative to $\underrightarrow{\mathcal{F}}_a$

Chapter 1

Introduction

Research on autonomous mobile robots such as unmanned aerial vehicles (UAV) and autonomous underwater vehicles (AUV) has been on the rise due to their ability to perform various tasks. One such task is the mapping of mining and construction sites with increased safety and speed without explicit human control. The robot's autonomy is achieved using a guidance, navigation and control (GNC) system. The guidance system determines the desired path to take from the robot's current location. The navigation system solves for the current state of the robot such as its location, velocity, heading, etc. Lastly, the control system determines the required input to achieve the desired path. Each system is connected in a closed-loop to achieve full autonomy of a robot; the output of one system affects the others. Thus, these three systems are of equal importance and the quality of the solution to each system affects the overall performance of a robot.

This thesis focuses on solving the navigation problem, namely the state estimation problem. State estimation refers to the process of estimating a robot's state given noisy and biased sensor measurements. Estimating the position and attitude of a mobile robot and mapping the surrounding environment accurately are crucial to increase the robustness of the robot's autonomous ability. Doing so is referred to as simultaneous localization and mapping (SLAM).

Over the past two decades, numerous solutions to the SLAM problem have been presented using different sensor suites for different purposes. Every sensor has its own strengths and weaknesses. For example, a camera is suitable for its cost effectiveness, ubiquity, and its ability to operate in GPS denied environments. However, the major disadvantage of vision-based SLAM is tracking failure due to low illumination, texture-less areas, insufficient scene overlap between frames or motion blur [1]. Light detection and ranging (LIDAR) is insensitive to illumination change and optical texture in the scene, and is well known for its robustness.

However, LIDAR-based solutions fail in environments with repeating structure such as a long tunnel, hallway or large open field. Fusing multiple sensors has been proposed as a method if mitigating these deficiencies, leading to an improved navigation solution. Camera and IMU, sonar and vision, camera and LIDAR, and LIDAR and IMU set-ups have been considered. First attempts at fusing sensor was done in a loosely-coupled manner. In a loosely coupled system, each sensor is used to obtain a probabilistic state estimate which is combined during an optimization step to yield the final state estimate. More recently, fusing sensor data in a tightly-coupled way was shown to achieve better accuracy and robustness when compared to the loosely-coupled framework. A tightly-coupled system is one in which the output of each sensor is fused together during the optimization step to yield a single state estimate. In particular, the vision community has seen major performance improvements since adopting a tightly-coupled sensor fusion framework. In this thesis, tightly-coupled multi-sensor fusion is considered for UAVs operating in GPS-denied environments. This is done to leverage the advantages of all three sensors, namely, the accuracy of LIDAR, high frequency of IMU, and visual information using a monocular camera.

1.1 Thesis Objective

The primary objective of this thesis is to develop a SLAM solution to be used in unknown, GPS denied, enviroment. Tightly-coupled LIDAR-Inertial-Visual Odometry and Mapping (LIVOM) in a sliding window framework is proposed to be incorporated in the SLAM pipeline. The combination of LIDAR, camera and IMU can leverage their strengths in their degenerate cases. Based on these objectives, the SLAM problem is split into two goals. Front-end data association consists of building constraints between robot states using the sensor measurements. Back-end optimization is the factor-graph optimized using linear solvers.

Another contribution of this thesis is the derivation of the factors using matrix Lie group theory. The robot states tend to have the matrix Lie group nature that upon linearization, Jacobian matrices may depend less, or not at all, on the state estimate which leads to improved convergence and consistency properties. Lastly, this thesis provides a novel method of tightly coupling the aforementioned sensors at an optimization level using a sliding window filter (SWF). In order to verify and demonstrate the capabilities of the proposed algorithms, the solution is tested in experimental settings.

1.2 Thesis Overview

This thesis is structured as follows. Chapter 2 covers the summary of mathematical concepts and notations used throughout this thesis. In Chapter 3, data preprocessing techniques and constructing the constraints in the factor graph are discussed. In Chapter 4, the novel LIVOM scheme is derived including the odometry system initialization, tightly-coupled optimization and factor marginalization. Results comparing the proposed odometry solution and state-of-the-art solutions such as Lidar-Inertial Odometry (LIO) and Visual-Inertial Odometry (VIO) are provided. Tests are conducted on real data. Lastly, Chapter 5 concludes the thesis along with recommendations for future work.

Chapter 2

Preliminaries

2.1 State Estimation

State estimation refers to the process of estimating a robot's state from noisy sensor measurements. The estimated states are commonly used in the guidance and control systems to compute an efficient path planning and the necessary control inputs to complete its task. Therefore, state estimation is a critical foundation for a closed-loop system of a robot as the accuracy of the estimate dictates the performance of a robot's ability in accomplishing tasks. In a typical localization problem, information about a robot's surroundings is given to a robot, and the only states to be estimated are the robot's navigation states, namely, position, attitude and velocity. However, in a more realistic scenario, the surrounding is perceived by noisy onboard sensors such as a camera, RADAR, LIDAR, etc. The perceived surrounding is often referred to as a map which ranges from sparse landmarks to dense point clouds. Since the map is perceived relative to the robot's state, and the robot's navigation states are estimated using the perceived surrounding, the navigation community has been interested in solving the simultaneous localization and mapping (SLAM) problem. The SLAM problem consists of estimating the robot navigation states and the map in a single optimization problem. In this section, based on [3], two broad categories of estimation, filtering and smoothing, are discussed.

2.1.1 Extended Kalman Filter

The Kalman filter (KF) is one of the most widely used state estimation algorithms for its simplicity and robustness. The filter is based on a Markov process as shown in Figure 2.1 which states that the state to be estimated only depends on the previous state, and not any other states in the past. Though the Kalman filter is an optimal solution only for a linear system with Gaussian noise assumption, most mobile robots have nonlinear system

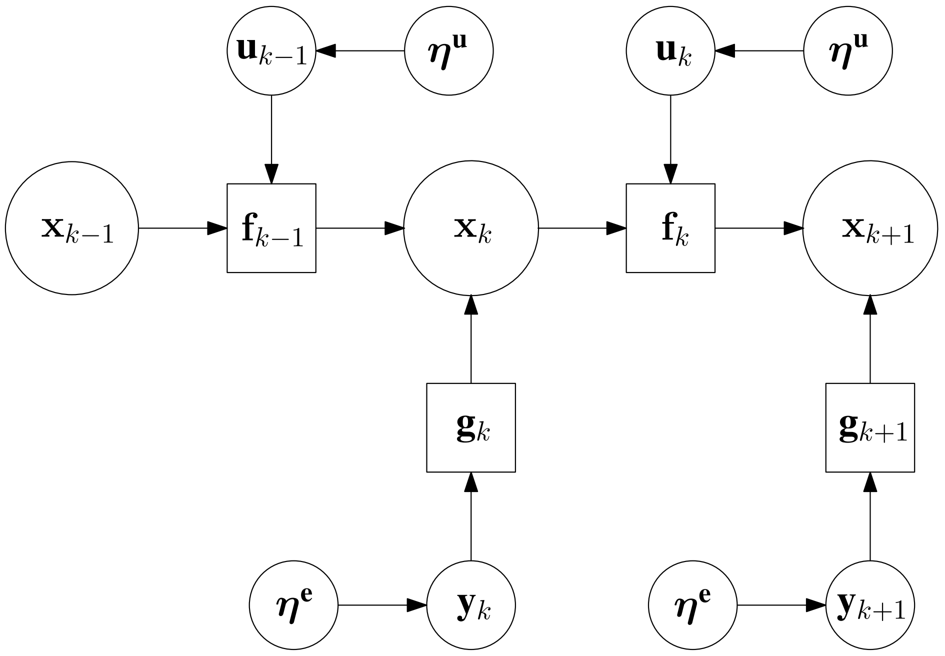


Figure 2.1: Graphical representation of the Markov process [3].

dynamics. The extended Kalman filter (EKF) is the nonlinear variant suitable for more practical purposes [3, Ch. 4.2]. Filtering consists of two steps: prediction and correction. During the prediction step, the robot state and its covariance are propagated forward in time using the process model. Subsequently, the predicted state and its covariance are corrected using an exteroceptive measurement model. Let $\mathbf{x} \in \mathbb{R}^n$ be the state, $\mathbf{u} \in \mathbb{R}^m$ be the interoceptive measurement or the system input, and $\boldsymbol{\eta}^{\boldsymbol{u}} \in \mathbb{R}^m$ be the noise associated with $\mathbf{u}$. Then, the continuous-time nonlinear process model is

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\eta}^u(t)\right). \tag{2.1}$$

The input noise is assumed to be additive white Gaussian noise, $\boldsymbol{\eta}^u \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}\delta(t-\tau)\right)$, where $\mathbf{Q}$ is the noise power spectral density (PSD). And the discrete-time exteroceptive measurement model at time t_k is given by

$$\mathbf{y}_k = \mathbf{g}_k\left(\mathbf{x}_k, \boldsymbol{\eta}^e_k\right), \tag{2.2}$$

where $\boldsymbol{\eta}^e_k \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_k\right)$. For brevity, (t) will be omitted unless required for clarity throughout this thesis.

The equations (2.1) and (2.2) are linearized about an operating point using a first-order

Taylor-series expansion

$$\delta\dot{\mathbf{x}} = \mathbf{F}\delta\mathbf{x} + \mathbf{L}\delta\boldsymbol{\eta}^u, \tag{2.3}$$

$$\delta\mathbf{y}_k = \mathbf{G}_k\delta\mathbf{x}_k + \mathbf{M}_k\delta\boldsymbol{\eta}_k^e, \tag{2.4}$$

where

$$\mathbf{F} = \left.\frac{\partial\mathbf{f}\left(\mathbf{x},\mathbf{u},\boldsymbol{\eta}^u\right)}{\partial\mathbf{x}}\right|_{\bar{\mathbf{x}},\mathbf{u},\mathbf{0}}, \tag{2.5}$$

$$\mathbf{L} = \left.\frac{\partial\mathbf{f}\left(\mathbf{x},\mathbf{u},\boldsymbol{\eta}^u\right)}{\partial\boldsymbol{\eta}^{\boldsymbol{u}}}\right|_{\bar{\mathbf{x}},\mathbf{u},\mathbf{0}}, \tag{2.6}$$

$$\mathbf{G}_k = \left.\frac{\partial\mathbf{y}_k\left(\mathbf{x}_k,\boldsymbol{\eta}_k^e\right)}{\partial\mathbf{x}}\right|_{\check{\mathbf{x}}_k,\mathbf{0}}, \tag{2.7}$$

$$\mathbf{M}_k = \left.\frac{\partial\mathbf{y}_k\left(\mathbf{x}_k,\boldsymbol{\eta}_k^e\right)}{\partial\boldsymbol{\eta}_{\boldsymbol{k}}^{\boldsymbol{e}}}\right|_{\check{\mathbf{x}}_k,\mathbf{0}}, \tag{2.8}$$

are the process and measurement-model Jacobian matrices around the operating point.

The robot state is predicted by integrating the process model

$$\check{\mathbf{x}}_k = \bar{\mathbf{x}}_{k-1} + \int_{k-1}^{k} \mathbf{f}\left(\bar{\mathbf{x}},\mathbf{u},\mathbf{0}\right)\mathrm{d}t, \tag{2.9}$$

where $\check{(\cdot)}$ represents the predicted state. The state covariance matrix $\mathbf{P}_k$ is propagated forward in time using the Riccati equation

$$\dot{\bar{\mathbf{P}}} = \mathbf{F}\bar{\mathbf{P}} + \bar{\mathbf{P}}\mathbf{F}^{\mathsf{T}} + \mathbf{L}\mathbf{Q}\mathbf{L}^{\mathsf{T}}. \tag{2.10}$$

The discrete-time Riccati equation is written as

$$\check{\mathbf{P}}_k = \mathbf{F}_{k-1}\bar{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^{\mathsf{T}} + \mathbf{L}_{k-1}\mathbf{Q}_{k-1}\mathbf{L}_{k-1}^{\mathsf{T}}. \tag{2.11}$$

When an exteroceptive measurement is received, the predicted state is corrected using

$$\bar{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k\left(\mathbf{y}_k - \mathbf{g}\left(\check{\mathbf{x}}_k,\mathbf{0}\right)\right), \tag{2.12}$$

where $\mathbf{y}_k - \mathbf{g}\left(\check{\mathbf{x}}_k,\mathbf{0}\right)$ is the innovation term and the Kalman gain $\mathbf{K}_k$ is given by

$$\mathbf{S}_k = \mathbf{G}_k\check{\mathbf{P}}_k\mathbf{G}_k^{\mathsf{T}} + \mathbf{M}_k\mathbf{R}_k\mathbf{M}_k^{\mathsf{T}}, \tag{2.13}$$

$$\mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{G}_k^{\mathsf{T}}\mathbf{S}_k^{-1}. \tag{2.14}$$

The state covariance matrix is corrected via

$$\bar{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{G}_k)\,\check{\mathbf{P}}_k\,(\mathbf{1} - \mathbf{K}_k\mathbf{G}_k)^{\mathsf{T}} + \mathbf{K}_k\mathbf{M}_k\mathbf{R}_k\mathbf{M}_k^{\mathsf{T}}\mathbf{K}_k^{\mathsf{T}}. \tag{2.15}$$

2.1.2 Factor-Graph-based Smoothing

Unlike solving for a single state at a time like in the filtering method, the smoothing method estimates a set of states in a single optimization problem given past measurements [3, Ch. 4.3]. A factor graph is a bipartite graph that represents the factorization of a function of several variables. Usually, the functions used in factorization are the functions of robot states. A graph-based method generally has nodes that represent robot states and the map, and the measurements that connect between the nodes as constraints. The goal of graph-based methods is to jointly optimize the poses of the nodes so as to minimize the error introduced by these constraints [5]. Let $\mathbf{x}$ be a set of robot states

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k+1} \\ \vdots \\ \mathbf{x}_{k+n} \end{bmatrix}, \tag{2.16}$$

where k is an arbitrary point in time and n is the size of a window to be optimized. The smoothing method uses a maximum a posteriori (MAP) apporach, which maximizes the posterior probability of the states given a set of measurements. The MAP approach solves the optimization problem

$$\mathbf{x} = \arg\min_{\mathbf{x}}\; J(\mathbf{x}), \tag{2.17}$$

where $J(\mathbf{x})$ is the objective function. The objective function comprises of the residuals as a function of the robot state in the factor graph. In this section, the factor graph is presented with the two functions, the process model (2.1) and the exteroceptive measurement model (2.2). The residuals associated with them are

$$\mathbf{e}_p(\mathbf{x}) = \mathbf{x}_k - \bar{\mathbf{x}}_k, \tag{2.18}$$

$$\mathbf{e}_{u,t}(\mathbf{x}) = \mathbf{x}_t - \mathbf{f}_{t-1}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}, \mathbf{0}), \tag{2.19}$$

$$\mathbf{e}_{y,t}(\mathbf{x}) = \mathbf{y}_t - \mathbf{g}_t(\mathbf{x}_t, \mathbf{0}), \tag{2.20}$$

where $t = k+1, \dots k+n$. $\mathbf{e}_p(\mathbf{x})$ is the residual associated with the prior factor. A prior factor constrains the current state estimate with the prior. Notice how the process model in the residual is in discrete time. This is due to the nature of the problem where a few instances of a trajectory are being optimized. After linearization about an operating point, $\bar{\mathbf{x}}$, using a first order Taylor-series expansion, the residuals have the form

$$\mathbf{e}_{u,t}(\mathbf{x}) = \mathbf{e}_{u,t}(\bar{\mathbf{x}}) + \left.\frac{\partial \mathbf{e}_{u,t}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}_t} \delta\mathbf{x} \tag{2.21}$$

$$= \underbrace{\bar{\mathbf{x}}_t - \mathbf{f}_{t-1}(\bar{\mathbf{x}}_{t-1}, \mathbf{u}_{t-1}, \mathbf{0})}_{\mathbf{e}_{u,t}(\bar{\mathbf{x}})} + \delta\mathbf{x}_t - \mathbf{F}_{t-1}\delta\mathbf{x}_{t-1}, \tag{2.22}$$

$$\mathbf{e}_{y,t}(\mathbf{x}) = \mathbf{e}_{y,t}(\bar{\mathbf{x}}) + \left.\frac{\partial \mathbf{e}_{y,t}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}_t} \delta\mathbf{x} \tag{2.23}$$

$$= \underbrace{\bar{\mathbf{y}}_t - \mathbf{g}_t(\bar{\mathbf{x}}_t, \mathbf{0})}_{\mathbf{e}_{y,t}(\bar{\mathbf{x}})} + \mathbf{G}_t\delta\mathbf{x}_t. \tag{2.24}$$

The objective function is rewritten in a quadratic form as

$$J(\mathbf{x}) = \tfrac{1}{2}\|\mathbf{e}_p(\mathbf{x})\|^2_{\mathbf{P}_0} + \tfrac{1}{2}\sum_{t\in[k+1,n]} \|\mathbf{e}_{u,t}(\mathbf{x})\|^2_{\mathbf{Q}} + \tfrac{1}{2}\sum_{t\in[k+1,n]} \|\mathbf{e}_{y,t}(\mathbf{x})\|^2_{\mathbf{R}}, \tag{2.25}$$

where $\mathbf{P}_0$ is the prior covariance associated with the initial state at time $t = k$, and $\|\mathbf{e}\|^2_{\mathbf{\Sigma}} \triangleq \mathbf{e}^{\mathsf{T}}\mathbf{\Sigma}^{-1}\mathbf{e}$ is defined as the Mahalanobis distance squared. The nonlinear weighted least-squares optimization problem can be solved using various methods such as gradient descent, Gauss-Newton, Levenberg-Marquardt, etc.

2.2 Matrix Lie Groups

In mathematics, Lie groups are smooth differentiable manifolds. A manifold is a space that locally resembles Euclidean space. Matrix Lie groups are Lie groups composed of $n \times n$ matrices that are closed under matrix multiplication. In this section, matrix Lie groups theory is discussed based on [3, 6, 7].

2.2.1 Lie Algebra

Let G be a matrix Lie group composed of invertible $n \times n$ matrices that is closed under matrix multiplication. The associated Lie algebra, denoted as $\mathfrak{g}$, is defined as the tangent space at the identity element of the group, $\mathfrak{g} \triangleq T_{\mathbf{1}}G$. The tangent space of G at any $\mathbf{X} \in G$ is denoted as $T_{\mathbf{X}}G$. The matrix Lie algebra is a vector space closed under the operation of

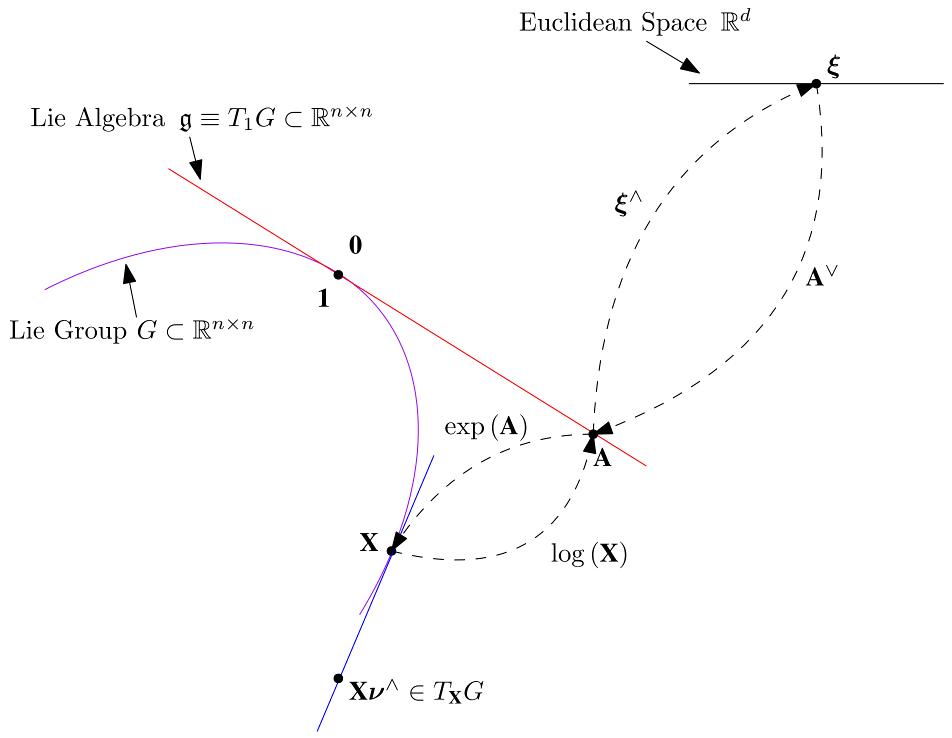

Euclidean Space $\mathbb{R}^d$
$\boldsymbol{\xi}$
Lie Algebra $\mathfrak{g} \equiv T_1 G \subset \mathbb{R}^{n\times n}$
$\boldsymbol{\xi}^\wedge$
$\mathbf{0}$
$\mathbf{1}$
$\mathbf{A}^\vee$
Lie Group $G \subset \mathbb{R}^{n\times n}$
$\exp(\mathbf{A})$
$\mathbf{A}$
$\mathbf{X}$
$\log(\mathbf{X})$
$\mathbf{X}\boldsymbol{\nu}^\wedge \in T_{\mathbf{X}} G$

Figure 2.2: Relation between matrix Lie group, matrix Lie algebra, and $\mathbb{R}^d$ [4].

the matrix Lie bracket defined as $[\mathbf{A}, \mathbf{B}] = \mathbf{AB} - \mathbf{BA}, \forall \mathbf{A}, \mathbf{B} \in \mathfrak{g}$. The elements of the Lie algebra can be mapped back and forth between $\mathfrak{g}$ and $\mathbb{R}^n$ using the operators

$$(\cdot)^\wedge : \mathbb{R}^n \to \mathfrak{g}, \tag{2.26}$$

$$(\cdot)^\vee : \mathfrak{g} \to \mathbb{R}^n. \tag{2.27}$$

2.2.2 Exponential and Logarithm

The exponential map of a Lie group maps an element of a Lie algebra to a corresponding point in the Lie group. For matrix Lie groups, the exponential map is simply the matrix exponential,

$$\mathbf{X} = \exp\left(\boldsymbol{\xi}^\wedge\right). \tag{2.28}$$

One useful property of the exponential map is

$$\mathbf{X}\exp\left(\boldsymbol{\xi}^\wedge\right)\mathbf{X}^\mathsf{T} = \exp\left(\mathbf{X}\boldsymbol{\xi}^\wedge\mathbf{X}^\mathsf{T}\right) = \exp\left(\mathbf{X}\boldsymbol{\xi}^\wedge\right), \tag{2.29}$$

$$\exp\left(\boldsymbol{\xi}^\wedge\right)\mathbf{X} = \mathbf{X}\exp\left(\mathbf{X}^\mathsf{T}\boldsymbol{\xi}^\wedge\right). \tag{2.30}$$

The inverse of the exponential map is the logarithmic map of the Lie group. It takes an element of a Lie group and maps it to a corresponding point in the Lie algebra

$$\boldsymbol{\xi}^{\wedge} = \log\left(\mathbf{X}\right). \tag{2.31}$$

For brevity of notation, the exponential map is denoted as $\exp\left(\boldsymbol{\xi}^{\wedge}\right) = \mathrm{Exp}\left(\boldsymbol{\xi}\right)$, and the logarithmic map is denoted as $\log\left(\boldsymbol{\xi}\right)^{\vee} = \mathrm{Log}\left(\boldsymbol{\xi}\right)$.

2.2.3 Adjoint Representation

The adjoint representation of a Lie group is a way of representing linear transformations of the group's Lie algebra. For any $\mathbf{X} \in G$, the adjoint representation is denoted as $\mathrm{Ad}_{\mathbf{X}} : \mathfrak{g} \to \mathfrak{g}$. The representation is not unique as it depends on the parametrization. However, a linear map is defined as $\mathrm{Ad}_{\mathbf{X}}\left(\boldsymbol{\xi}^{\wedge}\right) = \mathbf{X}\boldsymbol{\xi}^{\wedge}\mathbf{X}^{-1}$. The adjoint map transforms vectors from the tangent space about one matrix Lie group element to a different tangent space about another matrix Lie group. Shifting an element of the group across the exponential map is done by using

$$\mathbf{X}\mathrm{Exp}\left(\boldsymbol{\xi}\right) = \mathrm{Exp}\left(\mathrm{Ad}_{\mathbf{X}}\boldsymbol{\xi}\right)\mathbf{X}. \tag{2.32}$$

Given $\boldsymbol{\xi}^{\wedge}, \boldsymbol{\zeta}^{\wedge} \in \mathfrak{g}$, the adjoint representation of an element of the matrix Lie algebra, denoted as $\mathrm{ad}_{\boldsymbol{\xi}}$, is defined as

$$\boldsymbol{\xi}^{\wedge}\boldsymbol{\zeta}^{\wedge} - \boldsymbol{\zeta}^{\wedge}\boldsymbol{\xi}^{\wedge} = \left(-\mathrm{ad}_{\boldsymbol{\zeta}}\boldsymbol{\xi}\right)^{\wedge}. \tag{2.33}$$

2.2.4 Uncertainty Representation

Uncertainty representation in the Euclidean vector space is additive

$$\mathbf{x} = \bar{\mathbf{x}} + \delta\mathbf{x}, \tag{2.34}$$

where $\mathbf{x}, \bar{\mathbf{x}} \in \mathbb{R}^n$, and $\delta\mathbf{x} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Sigma}\right)$. Matrix Lie groups resemble Euclidean space, and yet the addition and subtraction in Euclidean space are described by matrix multiplication [8]. Multiplicative uncertainties are

$$\mathbf{X} = \bar{\mathbf{X}}\mathrm{Exp}\left(\delta\boldsymbol{\xi}\right), \qquad \mathbf{X} = \mathrm{Exp}\left(\delta\boldsymbol{\xi}\right)\bar{\mathbf{X}}, \tag{2.35}$$

where they are referred to as the left-invariant error and the right-invariant error respectively. When using the left-invariant error definition, the right Jacobian relates additive perturbation in the tangent space with multiplication

$$\operatorname{Exp}(\boldsymbol{\phi} + \delta\boldsymbol{\phi}) \approx \operatorname{Exp}(\boldsymbol{\phi}) \operatorname{Exp}(\mathbf{J}_r(\boldsymbol{\phi})\,\delta\boldsymbol{\phi}). \tag{2.36}$$

Further, the inverse right Jacobian is used in the first-order approximation of the logarithmic map

$$\operatorname{Log}(\operatorname{Exp}(\boldsymbol{\phi}) \operatorname{Exp}(\delta\boldsymbol{\phi})) \approx \boldsymbol{\phi} + \mathbf{J}_r^{-1}(\boldsymbol{\phi})\,\delta\boldsymbol{\phi}. \tag{2.37}$$

On the other hand, when using the right-invariant error definition, the right Jacobian in (2.36) and (2.37) is replaced by the left Jacobian

$$\operatorname{Exp}(\boldsymbol{\phi} + \delta\boldsymbol{\phi}) \approx \operatorname{Exp}(\mathbf{J}_l(\boldsymbol{\phi})\,\delta\boldsymbol{\phi}) \operatorname{Exp}(\boldsymbol{\phi}), \tag{2.38}$$

$$\operatorname{Log}(\operatorname{Exp}(\boldsymbol{\phi}) \operatorname{Exp}(\delta\boldsymbol{\phi})) \approx \boldsymbol{\phi} + \mathbf{J}_l^{-1}(\boldsymbol{\phi})\,\delta\boldsymbol{\phi}. \tag{2.39}$$

Definition 2.2.1 (Baker-Campbell-Hausdorff (BCH) formula)**.** Let $\mathbf{x}$ and $\mathbf{y}$ be two vectors in $\mathbb{R}^n$. Multiplication of the two corresponding matrix Lie group elements can be computed as an infinite sum of the corresponding elements of the Lie algebra,

$$\mathbf{z} = \operatorname{Log}(\operatorname{Exp}(\mathbf{x}) \operatorname{Exp}(\mathbf{y})) \tag{2.40}$$

$$= \left(\mathbf{x}^\wedge + \mathbf{y}^\wedge + \tfrac{1}{2}[\mathbf{x}^\wedge, \mathbf{y}^\wedge] + \frac{1}{12}\left([\mathbf{x}^\wedge, [\mathbf{x}^\wedge, \mathbf{y}^\wedge]] + [\mathbf{y}^\wedge, [\mathbf{y}^\wedge, \mathbf{x}^\wedge]]\right) - \frac{1}{24}[\mathbf{y}^\wedge, [\mathbf{x}^\wedge, [\mathbf{x}^\wedge, \mathbf{y}^\wedge]]] + \dots\right)^\vee \tag{2.41}$$

where the first-order approximation of the BCH formula is

$$\operatorname{Log}(\operatorname{Exp}(\mathbf{x}) \operatorname{Exp}(\mathbf{y})) \approx \mathbf{x} + \mathbf{y}. \tag{2.42}$$

The detailed derivation is shown in [3, Ch. 7.1.5].

2.2.5 The Special Orthogonal Group $SO(3)$

The special orthogonal group represents the three dimensional rotations and it is defined as

$$SO(3) = \left\{\mathbf{C} \in \mathbb{R}^{3\times 3} \mid \mathbf{C}^\mathsf{T}\mathbf{C} = \mathbf{1}, \det \mathbf{C} = 1\right\}. \tag{2.43}$$

The associated Lie algebra is defined as

$$\mathfrak{so}(3) = \left\{ \boldsymbol{\phi}^\times \in \mathbb{R}^{3\times 3} \mid \boldsymbol{\phi} \in \mathbb{R}^3 \right\}, \tag{2.44}$$

where $\boldsymbol{\phi}^\times$ is the skew-symmetric matrix representation of $\boldsymbol{\phi}$,

$$\boldsymbol{\phi}^\times = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\times = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}. \tag{2.45}$$

The adjoint representations of $SO(3)$ and $\mathfrak{so}(3)$ are simply the elements themselves

$$\mathrm{Ad}_{\mathbf{C}} = \mathbf{C}, \qquad \mathrm{ad}_{\boldsymbol{\phi}} = \boldsymbol{\phi}^\times. \tag{2.46}$$

The closed-form expression for the exponential map of $SO(3)$ is given by Rodrigue's formula,

$$\mathrm{Exp}(\boldsymbol{\phi}) = \mathbf{1} + \left(\frac{\sin\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|} \boldsymbol{\phi}^\times \right) + \left(\frac{1-\cos\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^2} \boldsymbol{\phi}^{\times 2} \right). \tag{2.47}$$

The closed-form expression of the logarithmic map is given as

$$\mathrm{Log}(\mathbf{C}) = \frac{\theta}{2\sin\theta} \left(\mathbf{C} - \mathbf{C}^\mathsf{T} \right)^\vee, \tag{2.48}$$

where

$$\theta = \cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{C}) - 1}{2} \right). \tag{2.49}$$

The closed-form expression of the left and right Jacobians of $SO(3)$ are given as

$$\mathbf{J}_r(\boldsymbol{\phi}) = \mathbf{1} - \left(\frac{1-\cos\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^2} \right) \boldsymbol{\phi}^\times + \left(\frac{\|\boldsymbol{\phi}\| - \sin\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^3} \right) \boldsymbol{\phi}^{\times 3}, \tag{2.50}$$

$$\mathbf{J}_l(\boldsymbol{\phi}) = \mathbf{1} + \left(\frac{1-\cos\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^2} \right) \boldsymbol{\phi}^\times + \left(\frac{\|\boldsymbol{\phi}\| - \sin\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^3} \right) \boldsymbol{\phi}^{\times 3}, \tag{2.51}$$

and their inverses as

$$\mathbf{J}_r^{-1}(\boldsymbol{\phi}) = \mathbf{1} + \tfrac{1}{2}\boldsymbol{\phi}^\times + \left(\frac{1}{\|\boldsymbol{\phi}\|^2} - \frac{1+\cos\|\boldsymbol{\phi}\|}{2\|\boldsymbol{\phi}\|\sin\|\boldsymbol{\phi}\|} \right) \boldsymbol{\phi}^{\times 2}, \tag{2.52}$$

$$\mathbf{J}_l^{-1}(\boldsymbol{\phi}) = \mathbf{1} - \tfrac{1}{2}\boldsymbol{\phi}^\times + \left(\frac{1}{\|\boldsymbol{\phi}\|^2} - \frac{1+\cos\|\boldsymbol{\phi}\|}{2\|\boldsymbol{\phi}\|\sin\|\boldsymbol{\phi}\|} \right) \boldsymbol{\phi}^{\times 2}. \tag{2.53}$$

2.2.6 The Special Euclidean Group $SE(3)$

The special Euclidean group represents the three dimensional rigid body transformation also known as a pose [3, Ch. 7], and it is defined as

$$SE\left(3\right) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} | \mathbf{C} \in SO\left(3\right), \mathbf{r} \in \mathbb{R}^3 \right\}. \tag{2.54}$$

The associated Lie algebra is defined as

$$\mathfrak{se}\left(3\right) = \left\{ \mathbf{\Xi} = \boldsymbol{\xi}^\wedge \in \mathbb{R}^{4\times 4} | \boldsymbol{\xi} \in \mathbb{R}^6 \right\}, \tag{2.55}$$

where

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\xi}^\phi \\ \boldsymbol{\xi}^r \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\xi}^{\phi\times} & \boldsymbol{\xi}^r \\ \mathbf{0} & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \boldsymbol{\xi}^\phi, \boldsymbol{\xi}^r \in \mathbb{R}^3. \tag{2.56}$$

The adjoint representation of elements of $SE\left(3\right)$ and $\mathfrak{se}\left(3\right)$ are

$$\mathrm{Ad}_{\mathbf{T}} = \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{r}^\times \mathbf{C} & \mathbf{C} \end{bmatrix} \in \mathbb{R}^{6\times 6}, \qquad \mathrm{ad}_{\boldsymbol{\xi}} = \begin{bmatrix} \boldsymbol{\xi}^{\phi\times} & \mathbf{0} \\ \boldsymbol{\xi}^{r\times} & \boldsymbol{\xi}^{\phi\times} \end{bmatrix}, \tag{2.57}$$

repsectively. The exponential map of $SE\left(3\right)$ is

$$\mathrm{Exp}\left(\boldsymbol{\xi}\right) = \begin{bmatrix} \mathrm{Exp}\left(\boldsymbol{\xi}^\phi\right) & \mathbf{J}_l\left(\boldsymbol{\xi}^\phi\right)\boldsymbol{\xi}^r \\ \mathbf{0} & 1 \end{bmatrix}, \tag{2.58}$$

where $\mathrm{Exp}\left(\boldsymbol{\xi}^\phi\right)$ and $\mathbf{J}_l\left(\boldsymbol{\xi}^\phi\right)$ are the exponential map and the left Jacobian of $SO\left(3\right)$. The expression of the logarithmic map is given as

$$\mathrm{Log}\left(\mathbf{T}\right) = \begin{bmatrix} \mathrm{Log}\left(\mathbf{C}\right) \\ \mathbf{J}_l^{-1}\mathrm{Log}\left(\mathbf{C}\right)\mathbf{r} \end{bmatrix}, \tag{2.59}$$

where $\mathrm{Log}\left(\mathbf{C}\right)$ is the logarithmic map of $SO\left(3\right)$.

2.2.7 Group of K Direct Isometries $SE_K(3)$

The properties of the group of K direct isometries are from [9, Ch. 3.4]. Various robot states including the 3D rotation can be represented using the special orthogonal group of K

direct isometries, and it is defined as

$$SE_k\left(3\right) = \left\{ \mathbf{X} = \begin{bmatrix} \mathbf{C} & \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_k \\ \mathbf{0} & 1 & 0 & \dots & 0 \\ \mathbf{0} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & 0 & 0 & \dots & 1 \end{bmatrix} \in \mathbb{R}^{(k+3)\times(k+3)} | \mathbf{C} \in SO\left(3\right), \mathbf{p}_i \in \mathbb{R}^3 \right\}. \quad (2.60)$$

The associated Lie algebra is defined as

$$\mathfrak{se}_k\left(3\right) = \left\{ \boldsymbol{\Xi} = \boldsymbol{\xi}^\wedge \in \mathbb{R}^{(k+3)\times(k+3)} | \boldsymbol{\xi} \in \mathbb{R}^{3\times(k+1)} \right\}, \quad (2.61)$$

where

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\xi}^{\phi\times} & \boldsymbol{\xi}^{\mathbf{p}_1} & \boldsymbol{\xi}^{\mathbf{p}_2} & \dots & \boldsymbol{\xi}^{\mathbf{p}_K} \\ \mathbf{0} & 0 & 0 & \dots & 0 \\ \mathbf{0} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \mathbf{0} & 0 & 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}^{(k+3)\times(k+3)}. \quad (2.62)$$

The adjoint representation of elements of $SE_k\left(3\right)$, and $\mathfrak{se}_k\left(3\right)$ are

$$\mathrm{Ad}_{\mathbf{X}} = \begin{bmatrix} \mathbf{C} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{p}_1{}^\times\mathbf{C} & \mathbf{C} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{p}_2{}^\times\mathbf{C} & \mathbf{0} & \mathbf{C} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{p}_k{}^\times\mathbf{C} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{C} \end{bmatrix}, \qquad \mathrm{ad}_{\boldsymbol{\xi}} = \begin{bmatrix} \boldsymbol{\xi}^{\phi\times} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \boldsymbol{\xi}^{\mathbf{p}_1\times} & \boldsymbol{\xi}^{\phi\times} & \mathbf{0} & \dots & \mathbf{0} \\ \boldsymbol{\xi}^{\mathbf{p}_2\times} & \mathbf{0} & \boldsymbol{\xi}^{\phi\times} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\xi}^{\mathbf{p}_k\times} & \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{\xi}^{\phi\times} \end{bmatrix}, \quad (2.63)$$

respectively. The exponential map of $SE_k\left(3\right)$ is

$$\mathrm{Exp}\left(\boldsymbol{\xi}\right) = \begin{bmatrix} \mathrm{Exp}\left(\boldsymbol{\xi}^{\phi}\right) & \mathbf{J}_l\left(\boldsymbol{\phi}\right)\boldsymbol{\xi}^{\mathbf{p}_1} & \mathbf{J}_l\left(\boldsymbol{\phi}\right)\boldsymbol{\xi}^{\mathbf{p}_2} & \dots \mathbf{J}_l\left(\boldsymbol{\phi}\right)\boldsymbol{\xi}^{\mathbf{p}_k} & \\ \mathbf{0} & 1 & 0 & \dots & 0 \\ \mathbf{0} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & 0 & 0 & \dots & 1 \end{bmatrix}, \quad (2.64)$$

where $\mathrm{Exp}\left(\boldsymbol{\xi}^{\phi}\right)$ and $\mathbf{J}_l\left(\boldsymbol{\xi}^{\phi}\right)$ are the exponential map and the left Jacobian of $SO(3)$. The expression of the logarithmic map is given as

$$\mathrm{Log}\left(\mathbf{X}\right) = \begin{bmatrix} \mathrm{Log}\left(\mathbf{C}\right) \\ \mathbf{J}_l^{-1}\mathrm{Log}\left(\mathbf{C}\right)\mathbf{p}_1 \\ \mathbf{J}_l^{-1}\mathrm{Log}\left(\mathbf{C}\right)\mathbf{p}_2 \\ \vdots \\ \mathbf{J}_l^{-1}\mathrm{Log}\left(\mathbf{C}\right)\mathbf{p}_k \end{bmatrix}, \tag{2.65}$$

where $\mathrm{Log}\left(\mathbf{C}\right)$ is the logarithmic map of $SO(3)$.

2.3 State Estimation on Matrix Lie Groups

The robot navigation state, which was initially expressed in the linear vector space $\mathbf{x} \in \mathbb{R}^n$, is now expressed as an element of a matrix Lie group $\mathbf{X} \in SE_k(3)$. The continuous-time nonlinear process model from (2.1) can be rewritten as

$$\dot{\mathbf{X}} = \mathbf{F}\left(\mathbf{X}, \Xi, \boldsymbol{\eta}^u\right). \tag{2.66}$$

The matrix Lie group states must be updated using a matrix multiplication instead of addition or subtraction. The input noise is assumed to be a zero-mean Gaussian with a covariance of $\mathbf{Q}^u$, $\boldsymbol{\eta}^u \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^u\right)$. Equation (2.66) represents the most general case as the input noise is embedded in the nonlinear process model. However, this general case complicates derivations. Thus, a more simplified uncertainty representation is used throughout

$$\dot{\mathbf{X}} = \mathbf{F}\left(\mathbf{X}, \Xi\right)\mathrm{Exp}\left(\boldsymbol{\eta}^u\right). \tag{2.67}$$

Furthermore, a measurement model is rewritten using matrix Lie groups

$$\mathbf{y}_k = \mathbf{g}_k\left(\mathbf{X}_k, \boldsymbol{\eta}_k^e\right). \tag{2.68}$$

The innovation term is explicitly defined with respect to the error definition used. In this section, only the left-invariant error definition is considered. Thus, the innovation term using the left-invariant error definition is

$$\mathbf{z}_k = \check{\mathbf{X}}_k^{-1}\left(\mathbf{y}_k - \mathbf{g}_k\left(\check{\mathbf{X}}_k, \mathbf{0}\right)\right). \tag{2.69}$$

In this section, the invariant extended Kalman filter (IEKF), a matrix Lie group variant of the extended Kalman filter (EKF), and smoothing using matrix Lie groups are discussed.

2.3.1 Invariant Extended Kalman Filter

The invariant extended Kalman Filter (IEKF) is a matrix Lie group variant of EKF. [10] states that for a specific class of systems, known as group-affine systems, with an invariant error definition, the error dynamics become state independent. Let $\mathbf{X} \in G$ be the state with its associated Lie algebra denoted as $\boldsymbol{\xi}^{|wedge} \in \mathfrak{g}$. Given the interoceptive measurement in a matrix Lie group $\boldsymbol{\Xi} \in G$ and the associated Gaussian noise $\boldsymbol{\eta}^u \in \mathbb{R}^n$, the robot process model (2.1) can be rewritten in a matrix form as

$$\dot{\mathbf{X}} = \mathbf{F}\left(\bar{\mathbf{X}}, \boldsymbol{\Xi}\right) \operatorname{Exp}\left(\boldsymbol{\eta}^u\right). \tag{2.70}$$

Using the left-invariant error definition, the process model is linearized about an operating point [11, Ch. 3.2] as

$$\delta\dot{\boldsymbol{\xi}} = \mathbf{A}\delta\boldsymbol{\xi} + \mathbf{L}\delta\boldsymbol{\eta}^u, \tag{2.71}$$

where

$$\mathbf{A} = \left.\frac{\partial \mathbf{F}\left(\mathbf{X}, \boldsymbol{\Xi}\right)}{\partial \mathbf{X}}\right|_{\bar{\mathbf{X}}, \boldsymbol{\Xi}}, \tag{2.72}$$

$$\mathbf{L} = \left.\frac{\partial \mathbf{F}\left(\mathbf{X}, \boldsymbol{\Xi}\right)}{\partial \boldsymbol{\eta}}\right|_{\bar{\mathbf{X}}, \boldsymbol{\Xi}}. \tag{2.73}$$

The state covariance matrix $\mathbf{P}$ is propagated forward in time using the continuous-time Riccati equation

$$\dot{\check{\mathbf{P}}} = \mathbf{A}\bar{\mathbf{P}} + \bar{\mathbf{P}}\mathbf{A}^{\mathsf{T}} + \mathbf{L}\mathbf{Q}\mathbf{L}^{\mathsf{T}}. \tag{2.74}$$

The correction step is performed when an exteroceptive measurement is available. Unlike the usual EKF, the linearization is performed on the innovation instead of the measurement model. The state update is done using

$$\bar{\mathbf{X}}_k = \check{\mathbf{X}}_k \operatorname{Exp}\left(-\mathbf{K}_k \mathbf{z}_k\right). \tag{2.75}$$

The Kalman gain at time t_k, $\mathbf{K}_k$, is computed using

$$\mathbf{S}_k = \mathbf{H}_k \check{\mathbf{P}}_k \mathbf{H}_k^{\mathsf{T}} + \mathbf{M}_k \mathbf{R}_k \mathbf{M}_k^{\mathsf{T}}, \tag{2.76}$$

$$\mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{H}_k^{\mathsf{T}} \mathbf{S}_k^{-1}, \tag{2.77}$$

where

$$\mathbf{H}_k = \left. \frac{\partial \mathbf{z}_k \left(\mathbf{X}_k, \boldsymbol{\eta}_k^e \right)}{\partial \mathbf{X}} \right|_{\bar{\mathbf{X}}_k, \mathbf{0}}, \tag{2.78}$$

$$\mathbf{M}_k = \left. \frac{\partial \mathbf{z}_k \left(\mathbf{X}_k, \boldsymbol{\eta}_k^e \right)}{\partial \boldsymbol{\eta}_{\boldsymbol{k}}^{e}} \right|_{\bar{\mathbf{X}}_k, \mathbf{0}}. \tag{2.79}$$

The covariance update is

$$\bar{\mathbf{P}}_k = \left(\mathbf{1} - \mathbf{K}_k \mathbf{H}_k \right) \check{\mathbf{P}}_k \left(\mathbf{1} - \mathbf{K}_k \mathbf{H}_k \right)^{\mathsf{T}} + \mathbf{K}_k \mathbf{M}_k \mathbf{R}_k \mathbf{M}_k^{\mathsf{T}} \mathbf{K}_k^{\mathsf{T}}. \tag{2.80}$$

Defining the residual definition in a matrix Lie group results in a more accurate covariance estimate, and results in a state independent input Jacobian, $\mathbf{A}$ when the process model is group affine. The detailed derivation of the error dynamics using both left and right-invariant Jacobians are in [11, Ch. 3].

2.3.2 Smoothing in a Matrix Lie Group Framework

Matrix Lie group theory can be incorporated to smoothing algorithm [12]. Let the state $\boldsymbol{\xi}$ be a set of robot states

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\xi}_k \\ \boldsymbol{\xi}_{k+1} \\ \vdots \\ \boldsymbol{\xi}_{k+n} \end{bmatrix}, \tag{2.81}$$

where k is an arbitrary point in time and n is the size of a window to be optimized. Each state $\boldsymbol{\xi}_t \in \mathbb{R}^n$ can be mapped onto the matrix Lie algebra

$$\boldsymbol{\xi}_t^{\wedge} \in \mathfrak{g}. \tag{2.82}$$

Now, the MAP approach solves the following optimization problem

$$\boldsymbol{\xi} = \arg\min_{\boldsymbol{\xi}} J(\boldsymbol{\xi}). \tag{2.83}$$

The residuals used in the cost function are

$$\mathbf{e}_p\left(\boldsymbol{\xi}\right) = \operatorname{Log}\left(\bar{\mathbf{X}}_k^{-1}\mathbf{X}\right), \tag{2.84}$$

$$\mathbf{e}_{u,t}\left(\boldsymbol{\xi}\right) = \operatorname{Log}\left(\mathbf{F}_{t-1}\left(\mathbf{X}_{t-1}, \boldsymbol{\Xi}_{t-1}, \mathbf{0}\right)^{-1}\mathbf{X}_t\right), \tag{2.85}$$

$$\mathbf{e}_{y,t}\left(\boldsymbol{\xi}\right) = \mathbf{X}_t^{-1}\left(\mathbf{y}_t - \mathbf{g}_t\left(\mathbf{X}_t, \mathbf{0}\right)\right), \qquad t = k+1, \ldots, k+n. \tag{2.86}$$

After linearization about an operating point, $\bar{\boldsymbol{\xi}}$, using a Taylor-series expansion, the residuals have the form

$$\mathbf{e}_{u,t}\left(\boldsymbol{\xi}\right) = \mathbf{e}_{u,t}\left(\bar{\boldsymbol{\xi}}\right) + \left.\frac{\partial \mathbf{e}_{u,t}\left(\boldsymbol{\xi}\right)}{\partial \boldsymbol{\xi}}\right|_{\bar{\boldsymbol{\xi}}_t}\delta\boldsymbol{\xi} \tag{2.87}$$

$$= \underbrace{\mathbf{F}_{t-1}\left(\bar{\mathbf{X}}_{t-1}, \boldsymbol{\Xi}_{t-1}, \mathbf{0}\right)^{-1}\bar{\mathbf{X}}_t}_{\mathbf{e}_{u,t}\left(\bar{\boldsymbol{\xi}}\right)} + \mathbf{A}_t\delta\boldsymbol{\xi}_t + \mathbf{A}_{t-1}\delta\boldsymbol{\xi}_{t-1}, \tag{2.88}$$

$$\mathbf{e}_{y,t}\left(\boldsymbol{\xi}\right) = \mathbf{e}_{y,t}\left(\bar{\boldsymbol{\xi}}\right) + \left.\frac{\partial \mathbf{e}_{y,t}\left(\boldsymbol{\xi}\right)}{\partial \boldsymbol{\xi}}\right|_{\bar{\boldsymbol{\xi}}_t}\delta\boldsymbol{\xi} \tag{2.89}$$

$$= \underbrace{\mathbf{X}_t^{-1}\left(\bar{\mathbf{y}}_t - \mathbf{g}_t\left(\bar{\mathbf{X}}_t, \mathbf{0}\right)\right)}_{\mathbf{e}_{y,t}\left(\bar{\boldsymbol{\xi}}\right)} + \mathbf{H}_t\delta\boldsymbol{\xi}_t. \tag{2.90}$$

The objective function J is rewritten in a quadratic form as

$$J\left(\boldsymbol{\xi}\right) = \tfrac{1}{2}\left\|\mathbf{e}_p\left(\boldsymbol{\xi}\right)\right\|_{\mathbf{P}_0}^2 + \tfrac{1}{2}\sum_{t\in[k+1,n]}\left\|\mathbf{e}_{u,t}\left(\boldsymbol{\xi}\right)\right\|_{\mathbf{Q}}^2 + \tfrac{1}{2}\sum_{t\in[k+1,n]}\left\|\mathbf{e}_{y,t}\left(\boldsymbol{\xi}\right)\right\|_{\mathbf{R}}^2. \tag{2.91}$$

Chapter 3

LIVOM - Data Processing

The front-end of local SLAM is mainly responsible for acquiring data from sensors and constructing a graph of measurement constraints. For example, stereo monocular, or RGB-D cameras can be used to extract features such as SIFT, SURF, or ORB from image data and track them in the subsequent frames. Matching features can then be used to compute pose-to-pose constraints. LIDAR-based systems can also be used to form these constraints by performing point cloud alignment and scan matching between successive scans. In this chapter, the process of data association is discussed for a monocular camera, LIDAR, and inertial measurement unit. Each sensor measurement is processed to form a constraint used in a factor graph which will be further investigated in Chapter 4.

3.1 Preintegrated IMU on Manifold

An inertial measurement unit (IMU) sensor usually includes a three-axis accelerometer and a three-axis rate gyroscope, and measures the acceleration and the angular velocity of the sensor with respect to the sensor frame. IMU measurements are acquired at a high frequency compared to other exteroceptive sensors such as a camera and a LIDAR which leads to fast growth of the number of variables used in the estimation. As a result, the use of preintegrated IMU measurements was first introduced by [13]. It consists of combining many inertial measurements between two subsequent exteroceptive measurements into a single relative motion constraint. The preintegration theory on-manifold, proposed by [14], reduces the computation power required significantly. Hence, it enables the application of incremental-smoothing in real-time SLAM to achieve higher accuracy and lower drift in longer trajectories. This section shows the derivation of the in-between preintegrated IMU factor.

3.1.1 Notation

The robot body's reference point is denoted as a point b. This point can be chosen arbitrarily, but it is usually chosen by the location of an IMU. The position of the robot's body relative to an arbitrary point in space, w, is described by the physical vector $\underrightarrow{r}^{bw}$. The rate of change of the position, namely the velocity, of a robot b relative to w, with respect to some inertial frame $\mathcal{F}_W$ is denoted as $\underrightarrow{r}^{bw^{\cdot W}} = \underrightarrow{v}^{bw/W}$. A physical vector $\underrightarrow{r}^{bw}$ resolved in $\mathcal{F}_W$ is denoted as $\mathbf{r}_W^{bw}$. Poisson's equation is

$$\dot{\mathbf{C}}_{WB} = \mathbf{C}_{WB} \boldsymbol{\omega}_B^{BW^\times}, \tag{3.1}$$

where $\boldsymbol{\omega}_B^{BW}$ is the angular velocity of $\mathcal{F}_B$ relative to $\mathcal{F}_W$ resolved in $\mathcal{F}_B$.

3.1.2 IMU Model and Robot Kinematics

Consider a noisy and biased rate-gyro measurement model of the form

$$\tilde{\boldsymbol{\omega}}_B^{BW}(t) = \boldsymbol{\omega}_B^{BW}(t) + \boldsymbol{\beta}_B^g(t) + \boldsymbol{\eta}_B^g(t), \tag{3.2}$$

where $\boldsymbol{\eta}_B^g(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^g \delta(t-\tau))$. The noise power spectral density (PSD) is denoted $\mathbf{Q}^g$ and $\boldsymbol{\beta}_B^g(t)$ is the rate gyro bias modelled as a random walk process,

$$\dot{\boldsymbol{\beta}}_B^g(t) = \boldsymbol{\eta}_B^{bg}(t), \tag{3.3}$$

where $\boldsymbol{\eta}_B^{bg}(t) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{bg} \delta(t-\tau)\right)$. Additionally, consider a noisy and biased accelerometer measurement model of the form

$$\tilde{\mathbf{a}}_B(t) = \mathbf{C}_{WB}^{\mathsf{T}}(t)\left(\mathbf{a}_W^{wb/W/W}(t) - \mathbf{g}_W\right) + \boldsymbol{\beta}_B^a(t) + \boldsymbol{\eta}_B^a(t), \tag{3.4}$$

where $\boldsymbol{\eta}_B^a(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^a \delta(t-\tau))$. The accelerometer bias, $\boldsymbol{\beta}_B^a(t)$, is modelled as a random walk process,

$$\dot{\boldsymbol{\beta}}_B^a(t) = \boldsymbol{\eta}_B^{ba}(t), \tag{3.5}$$

where $\boldsymbol{\eta}_B^{ba}(t) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}^{ba} \delta(t-\tau)\right)$.

The rigid-body kinematics equations in continuous time are

$$\dot{\mathbf{C}}_{WB}(t) = \mathbf{C}_{WB}(t)\,\mathrm{Exp}\left(\boldsymbol{\omega}_B^{BW}(t)\right), \tag{3.6}$$

$$\dot{\mathbf{v}}_W^{bw/W}(t) = \mathbf{a}_W^{bw/W/W}(t), \tag{3.7}$$

$$\dot{\mathbf{r}}_W^{bw}(t) = \mathbf{v}_W^{bw/W}(t). \tag{3.8}$$

Here, for the purpose of readability, (t) is omitted unless for clarity is required. Subscripts to the point b or frame B specify a state at time $t = t_i$ (i.e., $\mathbf{C}_{WB_i} = \mathbf{C}_{WB}(t_i)$ and $\mathbf{r}_W^{b_i w} = \mathbf{r}_W^{bw}(t_i)$). Using (3.2) and (3.4), $\mathbf{a}_W^{bw/W/W}$ and $\boldsymbol{\omega}_B^{BW}$ can be written as a function of the IMU measurement model. Thus (3.6), (3.7), and (3.8) become

$$\dot{\mathbf{C}}_{WB} = \mathbf{C}_{WB}\mathrm{Exp}\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \boldsymbol{\beta}_B^g - \boldsymbol{\eta}_B^g\right), \tag{3.9}$$

$$\dot{\mathbf{v}}_W^{bw/W} = \mathbf{C}_{WB}\left(\tilde{\mathbf{a}}_B - \boldsymbol{\beta}_B^a - \boldsymbol{\eta}_B^a\right) + \mathbf{g}_W, \tag{3.10}$$

$$\dot{\mathbf{r}}_W^{bw} = \mathbf{v}_W^{bw/W}. \tag{3.11}$$

The state at time $t = t_j$ is obtained by integrating (3.9), (3.10), and (3.11)

$$\mathbf{C}_{WB_j} = \mathbf{C}_{WB_i}\mathrm{Exp}\left(\int_{\tau\in[t_i,t_j]} \tilde{\boldsymbol{\omega}}_B^{BW} - \boldsymbol{\beta}_B^g - \boldsymbol{\eta}_B^g \mathrm{d}\tau\right), \tag{3.12}$$

$$\mathbf{v}_W^{b_j w/W} = \mathbf{v}_W^{b_i w/W} + \int_{\tau\in[t_i,t_j]} \left(\mathbf{C}_{WB}\left(\tilde{\boldsymbol{a}}_B - \boldsymbol{\beta}_B^a - \boldsymbol{\eta}_B^a\right) + \mathbf{g}_W\right)\mathrm{d}\tau, \tag{3.13}$$

$$\mathbf{r}_W^{b_j w} = \mathbf{r}_W^{b_i w} + \iint_{\tau\in[t_i,t_j]} \left(\mathbf{C}_{WB}\left(\tilde{\boldsymbol{a}}_B - \boldsymbol{\beta}_B^a - \boldsymbol{\eta}_B^a\right) + \mathbf{g}_W\right)\mathrm{d}\tau^2. \tag{3.14}$$

Note that any integration method can be used. For example, assuming the zero-order hold assumption on the measurements and applying Euler integration yields

$$\mathbf{C}_{WB_j} = \mathbf{C}_{WB_i}\mathrm{Exp}\left(\sum_{k=i}^{j-1} \tilde{\boldsymbol{\omega}}_{B_k}^{BW} - \boldsymbol{\beta}_{B_k}^g - \boldsymbol{\eta}_{B_k}^g \Delta t\right), \tag{3.15}$$

$$\mathbf{v}_W^{b_j w/W} = \mathbf{v}_W^{b_i w/W} + \mathbf{g}\Delta t_{ij} + \sum_{k=i}^{j-1} \mathbf{C}_{WB_k}\left(\tilde{\boldsymbol{a}}_{B_k} - \boldsymbol{\beta}_{B_k}^a - \boldsymbol{\eta}_{B_k}^a\right)\Delta t, \tag{3.16}$$

$$\mathbf{r}_W^{b_j w} = \mathbf{r}_W^{b_i w} + \sum_{k=i}^{j-1}\left(\mathbf{v}_W^{b_k w/W}\Delta t - \tfrac{1}{2}\mathbf{g}\Delta t^2 + \tfrac{1}{2}\mathbf{C}_{WB_k}\left(\tilde{\boldsymbol{a}}_{B_k} - \boldsymbol{\beta}_{B_k}^a - \boldsymbol{\eta}_{B_k}^a\right)\Delta t^2\right), \tag{3.17}$$

where $\Delta t_{ij} = \sum_{k=i}^{j-1}\Delta t$. To prevent any loss of generality, (3.9), (3.10), and (3.11) will be used in the following sections.

3.1.3 Relative Motion Increments

Although (3.9), (3.10), and (3.11) provide an estimate of the relative motion between two instances, the integration has to be repeated whenever there is a change in the rotation $\mathbf{C}_{WB}$. This makes it necessary to re-evaluate the integration at every incoming IMU measurement. To avoid this procedure, a relative motion increment (RMI) that is independent of the robot state is introduced

$$\Delta\mathbf{C}_{ij} \triangleq \mathbf{C}_{WB_i}^{\mathsf{T}}\mathbf{C}_{WB_j} \tag{3.18}$$

$$= \operatorname{Exp}\left(\int_{\tau\in[t_i,t_j]} \tilde{\boldsymbol{\omega}}_B^{BW} - \boldsymbol{\beta}_{B_i}^g - \boldsymbol{\eta}_B^g \mathrm{d}\tau\right), \tag{3.19}$$

$$\Delta\mathbf{v}_{ij} \triangleq \mathbf{C}_{WB_i}^{\mathsf{T}}\left(\mathbf{v}_W^{b_jw/W} - \mathbf{v}_W^{b_iw/W} - \mathbf{g}_W\Delta t_{ij}\right) \tag{3.20}$$

$$= \int_{\tau\in[t_i,t_j]} \Delta\mathbf{C}_{i\tau}\left(\tilde{\mathbf{a}}_B - \boldsymbol{\beta}_{B_i}^a - \boldsymbol{\eta}_B^a\right)\mathrm{d}\tau, \tag{3.21}$$

$$\Delta\mathbf{r}_{ij} \triangleq \mathbf{C}_{WB_i}^{\mathsf{T}}\left(\mathbf{r}_W^{b_jw} - \mathbf{r}_W^{b_iw} - \mathbf{v}_W^{b_iw/W}\Delta t_{ij} - \tfrac{1}{2}\mathbf{g}_W\Delta t_{ij}^2\right) \tag{3.22}$$

$$= \iint_{\tau\in[t_i,t_j]} \Delta\mathbf{C}_{i\tau}\left(\tilde{\mathbf{a}}_B - \boldsymbol{\beta}_{B_i}^a - \boldsymbol{\eta}_B^a\right)\mathrm{d}\tau^2. \tag{3.23}$$

Biases are assumed constant during the integration. Thus, $(\boldsymbol{\beta})_i$ represents the constant bias at $t = t_i$. Substituting the RMI back into (3.9), (3.10), and (3.11) results in

$$\mathbf{C}_{WB_j} = \mathbf{C}_{WB_i}\Delta\mathbf{C}_{ij}, \tag{3.24}$$

$$\mathbf{v}_W^{b_jw/W} = \mathbf{v}_W^{b_iw/W} + \Delta t_{ij}\mathbf{g}_W + \mathbf{C}_{WB_i}\Delta\mathbf{v}_{ij}, \tag{3.25}$$

$$\mathbf{r}_W^{b_jw} = \mathbf{r}_W^{b_iw} + \Delta t_{ij}\mathbf{v}_W^{b_iw/W} + \tfrac{1}{2}\Delta t_{ij}^2\mathbf{g}_W + \mathbf{C}_{WB_i}\Delta\mathbf{r}_{ij}. \tag{3.26}$$

Notice how the RMI is independent from the robot's state at time t_i, as well as from gravity, and it can be obtained solely by integrating the measurements.

The preintegration theory focuses on the analysis of the RMI. The equation (3.19), (3.21), and (3.23) enable integration of the measurements without prior state information, and (3.24), (3.25), and (3.26) relate the integrated measurements to the states using the robot's kinematics model. Thus, the RMI can be considered as a new input to the process model. Recall that an IMU measurement is considered as a random variable with a nominal measurement and some noise associated with it. Integrating the random variable requires integrating the additive Gaussian noise. This can be done by propagating the covariance associated with the RMI forward in time. The error propagation of the RMI is studied in the following section with two different error definitions *MEKF*-based, and *IEKF*-based. Both methods of derivation rely on matrix Lie group theory, where the former uses $SO(3)$ to describe the rotation, and the latter uses $SE_2(3)$ to describe the pose and the velocity of the robot. The latter results less measurement-dependent Jacobian than the former.

The RMI equations (3.19), (3.21), and (3.23) can be rewritten in a differential form as

$$\dot{\Delta\mathbf{C}}_{ik} = \Delta\mathbf{C}_{ik}\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \boldsymbol{\beta}_{B_i}^g - \boldsymbol{\eta}_B^g\right)^\times, \tag{3.27}$$

$$\dot{\Delta\mathbf{v}}_{ik} = \Delta\mathbf{C}_{ik}\left(\tilde{\mathbf{a}}_B - \boldsymbol{\beta}_{B_i}^a - \boldsymbol{\eta}_B^a\right), \tag{3.28}$$

$$\dot{\Delta\mathbf{r}}_{ik} = \Delta\mathbf{v}_{ik}, \tag{3.29}$$

where k can be any time instance. This differential form is useful for deriving error propagation.

3.1.3.1 Linearization - MEKF-based Approach

Multiplicative-EKF (MEKF) is one of many EKF variants that uses traditional way of defining the state errors [15, Ch. 11.5]. Consider the nominal RMI $\bar{\Delta\mathbf{C}}_{ik}$, $\bar{\Delta\mathbf{v}}_{ik}$, and $\bar{\Delta\mathbf{r}}_{ik}$, then the *MEKF*-based errors are defined as

$$\delta\Delta\mathbf{C}_{ik} = \bar{\Delta\mathbf{C}}_{ik}^{\mathsf{T}}\Delta\mathbf{C}_{ik}, \tag{3.30}$$

$$\delta\Delta\mathbf{v}_{ik} = \Delta\mathbf{v}_{ik} - \bar{\Delta\mathbf{v}}_{ik}, \tag{3.31}$$

$$\delta\Delta\mathbf{r}_{ik} = \Delta\mathbf{r}_{ik} - \bar{\Delta\mathbf{r}}_{ik}. \tag{3.32}$$

The error propagation is computed by the time-derivative of the error definitions that leads to equations of the form

$$\begin{aligned}
\delta\dot{\Delta\mathbf{C}}_{ik} &= \dot{\bar{\Delta\mathbf{C}}}_{ik}^{\mathsf{T}}\Delta\mathbf{C}_{ik} + \bar{\Delta\mathbf{C}}_{ik}^{\mathsf{T}}\dot{\Delta\mathbf{C}}_{ik} && (3.33)\\
&= -\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^g\right)^\times \bar{\Delta\mathbf{C}}_{ik}^{\mathsf{T}}\Delta\mathbf{C}_{ik} + \bar{\Delta\mathbf{C}}_{ik}^{\mathsf{T}}\Delta\mathbf{C}_{ik}\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \boldsymbol{\beta}_{B_i}^g - \boldsymbol{\eta}_B^g\right)^\times && (3.34)\\
&= -\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^g\right)^\times \delta\Delta\mathbf{C}_{ik} + \delta\Delta\mathbf{C}_{ik}\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \boldsymbol{\beta}_{B_i}^g - \boldsymbol{\eta}_B^g\right)^\times, && (3.35)
\end{aligned}$$

$$\delta \dot{\Delta \mathbf{v}}_{ik} = \dot{\Delta \mathbf{v}}_{ik} - \dot{\bar{\Delta \mathbf{v}}}_{ik} \tag{3.36}$$
$$= \Delta \mathbf{C}_{ik} \left(\tilde{\mathbf{a}}_B - \boldsymbol{\beta}^a_{B_i} - \boldsymbol{\eta}^a_B \right) - \bar{\Delta \mathbf{C}}_{ik} \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}^a_{B_i} \right) \tag{3.37}$$
$$= \bar{\Delta \mathbf{C}}_{ik} \delta \Delta \mathbf{C}_{ik} \left(\tilde{\mathbf{a}}_B - \boldsymbol{\beta}^a_{B_i} - \boldsymbol{\eta}^a_B \right) - \bar{\Delta \mathbf{C}}_{ik} \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}^a_{B_i} \right), \tag{3.38}$$

$$\delta \dot{\Delta \mathbf{r}}_{ik} = \dot{\Delta \mathbf{r}}_{ik} - \dot{\bar{\Delta \mathbf{r}}}_{ik} \tag{3.39}$$
$$= \Delta \mathbf{v}_{ik} - \bar{\Delta \mathbf{v}}_{ik} \tag{3.40}$$
$$= \delta \Delta \mathbf{v}_{ik}. \tag{3.41}$$

To linearize (3.35) and (3.38), let $\delta \Delta \mathbf{C}_{ik} \approx \mathbf{1} + \delta \Delta \boldsymbol{\phi}^\times_{ik}$, $\boldsymbol{\eta}^g_B \approx \delta \boldsymbol{\eta}^g_B$, and $\boldsymbol{\eta}^a_B \approx \delta \boldsymbol{\eta}^a_B$. Neglecting the second order terms, (3.35) and (3.38) are then approximated as

$$\delta \dot{\Delta \boldsymbol{\phi}}^\times_{ik} = - \left(\tilde{\boldsymbol{\omega}}^{BW}_B - \bar{\boldsymbol{\beta}}^g_{B_i} \right)^\times \left(\mathbf{1} + \delta \Delta \boldsymbol{\phi}^\times_{ik} \right) + \left(\mathbf{1} + \delta \Delta \boldsymbol{\phi}^\times_{ik} \right) \left(\tilde{\boldsymbol{\omega}}^{BW}_B - \bar{\boldsymbol{\beta}}^g_{B_i} - \delta \boldsymbol{\beta}^g_{B_i} - \delta \boldsymbol{\eta}^g_B \right)^\times \tag{3.42}$$
$$= - \left(\tilde{\boldsymbol{\omega}}^{BW}_B - \bar{\boldsymbol{\beta}}^g_{B_i} \right)^\times \delta \Delta \boldsymbol{\phi}^\times_{ik} + \delta \Delta \boldsymbol{\phi}^\times_{ik} \left(\tilde{\boldsymbol{\omega}}^{BW}_B - \bar{\boldsymbol{\beta}}^g_{B_i} \right)^\times - \delta \boldsymbol{\beta}^{g\ \times}_{B_i} - \delta \boldsymbol{\eta}^{g\ \times}_B \tag{3.43}$$
$$= \left(-\mathrm{ad}_{\left(\tilde{\boldsymbol{\omega}}^{BW}_B - \bar{\boldsymbol{\beta}}^g_{B_i} \right)} \delta \Delta \boldsymbol{\phi}_{ik} \right)^\times - \delta \boldsymbol{\beta}^{g\ \times}_{B_i} - \delta \boldsymbol{\eta}^{g\ \times}_B, \tag{3.44}$$
$$\delta \dot{\Delta \boldsymbol{\phi}}_{ik} = -\mathrm{ad}_{\left(\tilde{\boldsymbol{\omega}}^{BW}_B - \bar{\boldsymbol{\beta}}^g_{B_i} \right)} \delta \Delta \boldsymbol{\phi}_{ik} - \delta \boldsymbol{\beta}^g_{B_i} - \delta \boldsymbol{\eta}^g_B, \tag{3.45}$$

$$\delta \dot{\Delta \mathbf{v}}_{ik} = \bar{\Delta \mathbf{C}}_{ik} \left(\mathbf{1} + \delta \Delta \boldsymbol{\phi}^\times_{ik} \right) \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}^a_{B_i} - \delta \boldsymbol{\beta}^a_{B_i} - \delta \boldsymbol{\eta}^a_B \right) - \bar{\Delta \mathbf{C}}_{ik} \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}^a_{B_i} \right) \tag{3.46}$$
$$= \bar{\Delta \mathbf{C}}_{ik} \delta \Delta \boldsymbol{\phi}^\times_{ik} \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}^a_{B_i} \right) - \bar{\Delta \mathbf{C}}_{ik} \delta \boldsymbol{\beta}^a_{B_i} - \bar{\Delta \mathbf{C}}_{ik} \delta \boldsymbol{\eta}^a_B \tag{3.47}$$
$$= -\bar{\Delta \mathbf{C}}_{ik} \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}^a_{B_i} \right)^\times \delta \Delta \boldsymbol{\phi}_{ik} - \bar{\Delta \mathbf{C}}_{ik} \delta \boldsymbol{\beta}^a_{B_i} - \bar{\Delta \mathbf{C}}_{ik} \delta \boldsymbol{\eta}^a_B. \tag{3.48}$$

Similarly, the bias error propagation is linearized by computing the time-derivative of the bias error

$$\delta \boldsymbol{\beta}^u_B = \boldsymbol{\beta}^u_B - \bar{\boldsymbol{\beta}}^u_B, \tag{3.49}$$
$$\delta \dot{\boldsymbol{\beta}}^u_B = \dot{\boldsymbol{\beta}}^u_B - \dot{\bar{\boldsymbol{\beta}}}^u_B = \boldsymbol{\eta}^{bu}_B, \tag{3.50}$$

where u can be either g for a rate gyroscope, and a for an accelerometer. Combining the linearized error dynamics (3.35), (3.38), (3.41), and (3.50) yields

$$\dot{\Delta \boldsymbol{\xi}}_{ik} = \mathbf{F}_{ik} \Delta \boldsymbol{\xi}_{ik} + \mathbf{L}_{ik} \boldsymbol{\eta}, \tag{3.51}$$

where

$$\Delta\boldsymbol{\xi}_{ik} = \begin{bmatrix} \delta\Delta\boldsymbol{\phi}_{ik}^{\mathsf{T}} & \delta\Delta\mathbf{v}_{ik}^{\mathsf{T}} & \delta\Delta\mathbf{r}_{ik}^{\mathsf{T}} & \delta\boldsymbol{\beta}_{ik}^{a\,\mathsf{T}} & \delta\boldsymbol{\beta}_{ik}^{g\,\mathsf{T}} \end{bmatrix}^{\mathsf{T}}, \tag{3.52}$$

$$\boldsymbol{\eta} = \begin{bmatrix} \delta\boldsymbol{\eta}_{B}^{a\,\mathsf{T}} & \delta\boldsymbol{\eta}_{B}^{g\,\mathsf{T}} & \delta\boldsymbol{\eta}_{B}^{ba\,\mathsf{T}} & \delta\boldsymbol{\eta}_{B}^{bg\,\mathsf{T}} \end{bmatrix}^{\mathsf{T}}, \tag{3.53}$$

and

$$\mathbf{F}_{ik} = \begin{bmatrix} -\mathrm{ad}_{\left(\tilde{\boldsymbol{\omega}}_{B}^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^{g}\right)} & & & & -\mathbf{1} \\ -\Delta\bar{\mathbf{C}}_{ik}\left(\tilde{\mathbf{a}}_{B} - \bar{\boldsymbol{\beta}}_{B_i}^{a}\right)^{\times} & & & -\Delta\bar{\mathbf{C}}_{ik} & \\ & \mathbf{1} & & & \\ & & & & \\ & & & & \end{bmatrix}, \qquad \mathbf{L}_{ik} = \begin{bmatrix} & \mathbf{1} & & \\ -\Delta\mathbf{C}_{ik} & & & \\ & & & \\ & & \mathbf{1} & \\ & & & \mathbf{1} \end{bmatrix}. \tag{3.54}$$

Notice $\mathbf{F}_{ik}$ and $\mathbf{L}_{ik}$ depend on preintegrated measurements, $\Delta\bar{\mathbf{C}}_{ik}$. Integrated measurements accumulate the noise associated with each measurement which may result in inaccurate Jacobian matrices.

3.1.3.2 Linearization - IEKF-based Approach

Linearization on the RMI in a matrix Lie group framework is first proposed by [16]. The RMI are concatenated to form an element of $SE_K(3)$. The robot pose and velocity are cast to form a group

$$\Delta\mathbf{T}_{ik} = \begin{bmatrix} \Delta\mathbf{C}_{ik} & \Delta\mathbf{v}_{ik} & \Delta\mathbf{r}_{ik} \\ & 1 & \\ & & 1 \end{bmatrix}, \tag{3.55}$$

and the sensor biases are cast to form another group

$$\mathbf{B}_{B} = \begin{bmatrix} \mathbf{1} & \boldsymbol{\beta}_{B_i}^{a} & \boldsymbol{\beta}_{B_i}^{g} \\ & 1 & \\ & & 1 \end{bmatrix}. \tag{3.56}$$

According to [17], the two groups can be cast into a single matrix Lie group

$$\Delta\mathbf{X}_{ik} = \begin{bmatrix} \Delta\mathbf{T}_{ik} & \\ & \mathbf{B}_{B} \end{bmatrix}. \tag{3.57}$$

There are two ways of defining errors: by left multiplication $\delta\Delta\mathbf{X}^L_{ik} = \bar{\Delta\mathbf{X}}^{-1}_{ik}\Delta\mathbf{X}$, and by right multiplication $\delta\Delta\mathbf{X}^R_{ik} = \Delta\mathbf{X}_{ik}\bar{\Delta\mathbf{X}}^{-1}_{ik}$, namely *LIEKF* and *RIEKF* respectively. Only the error defined by the left multiplication is analyzed in this section. The *LIEKF*-based errors are

$$\delta\Delta\mathbf{X}^L_{ik} = \bar{\Delta\mathbf{X}}^{-1}_{ik}\Delta\mathbf{X}_{ik} = \begin{bmatrix} \bar{\Delta\mathbf{T}}^{-1}_{ik} & \\ & \bar{\mathbf{B}}^{-1}_{B} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{T}_{ik} & \\ & \mathbf{B}_{B} \end{bmatrix} = \begin{bmatrix} \delta\Delta\mathbf{T}_{ik} & \\ & \delta\mathbf{B}_{B} \end{bmatrix}, \tag{3.58}$$

where

$$\begin{aligned} \delta\Delta\mathbf{T}^L_{ik} &= \bar{\Delta\mathbf{T}}^{-1}_{ik}\Delta\mathbf{T}_{ik} && (3.59)\\ &= \begin{bmatrix} \bar{\Delta\mathbf{C}}^{\mathsf{T}}_{ik} & -\Delta\mathbf{C}^{\mathsf{T}}_{ik}\bar{\Delta\mathbf{v}} & -\Delta\mathbf{C}^{\mathsf{T}}_{ik}\bar{\Delta\mathbf{r}} \\ & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{C}_{ik} & \Delta\mathbf{v} & \Delta\mathbf{r} \\ & 1 & \\ & & 1 \end{bmatrix} && (3.60)\\ &= \begin{bmatrix} \bar{\Delta\mathbf{C}}^{\mathsf{T}}_{ik}\Delta\mathbf{C}_{ik} & \bar{\Delta\mathbf{C}}^{\mathsf{T}}_{ik}\left(\Delta\mathbf{v}_{ik} - \bar{\Delta\mathbf{v}}_{ik}\right) & \bar{\Delta\mathbf{C}}^{\mathsf{T}}_{ik}\left(\Delta\mathbf{r}_{ik} - \bar{\Delta\mathbf{r}}_{ik}\right) \\ & 1 & \\ & & 1 \end{bmatrix}, && (3.61) \end{aligned}$$

and

$$\begin{aligned} \delta\mathbf{B}^L_{B} &= \bar{\mathbf{B}}^{-1}_{B}\mathbf{B}_{B} && (3.62)\\ &= \begin{bmatrix} \mathbf{1} & -\bar{\boldsymbol{\beta}}^a_{B_i} & -\bar{\boldsymbol{\beta}}^g_{B_i} \\ & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} \mathbf{1} & \boldsymbol{\beta}^a_{B_i} & \boldsymbol{\beta}^g_{B_i} \\ & 1 & \\ & & 1 \end{bmatrix} && (3.63)\\ &= \begin{bmatrix} \mathbf{1} & \boldsymbol{\beta}^a_{B_i} - \bar{\boldsymbol{\beta}}^a_{B_i} & \boldsymbol{\beta}^g_{B_i} - \bar{\boldsymbol{\beta}}^g_{B_i} \\ & 1 & \\ & & 1 \end{bmatrix}. && (3.64) \end{aligned}$$

Each element in the matrix Lie group is extracted

$$\begin{aligned} \delta\Delta\mathbf{C}_{ik} &= \bar{\Delta\mathbf{C}}^{\mathsf{T}}_{ik}\Delta\mathbf{C}_{ik}, && (3.65)\\ \delta\Delta\mathbf{v}_{ik} &= \bar{\Delta\mathbf{C}}^{\mathsf{T}}_{ik}\left(\Delta\mathbf{v}_{ik} - \bar{\Delta\mathbf{v}}_{ik}\right), && (3.66)\\ \delta\Delta\mathbf{r}_{ik} &= \bar{\Delta\mathbf{C}}^{\mathsf{T}}_{ik}\left(\Delta\mathbf{r}_{ik} - \bar{\Delta\mathbf{r}}_{ik}\right), && (3.67)\\ \delta\boldsymbol{\beta}^a_{B_i} &= \boldsymbol{\beta}^a_{B_i} - \bar{\boldsymbol{\beta}}^a_{B_i}, && (3.68)\\ \delta\boldsymbol{\beta}^g_{B_i} &= \boldsymbol{\beta}^g_{B_i} - \bar{\boldsymbol{\beta}}^g_{B_i}. && (3.69) \end{aligned}$$

The attitude and bias errors are identical to (3.30) and (3.49). Thus, the linearization on the error dynamics remains the same. On the other hand, the velocity and position errors are propagated as

$$\begin{aligned}
\delta \dot{\Delta \mathbf{v}}_{ik} &= \dot{\bar{\Delta \mathbf{C}}}_{ik}^{\mathsf{T}} \left(\Delta \mathbf{v}_{ik} - \bar{\Delta \mathbf{v}}_{ik}\right) + \bar{\Delta \mathbf{C}}_{ik}^{\mathsf{T}} \left(\dot{\Delta \mathbf{v}}_{ik} - \dot{\bar{\Delta \mathbf{v}}}_{ik}\right) && (3.70)\\
&= \left(\bar{\Delta \mathbf{C}}_{ik} \left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^{g}\right)^{\times}\right)^{\mathsf{T}} \left(\Delta \mathbf{v}_{ik} - \bar{\Delta \mathbf{v}}_{ik}\right) + && (3.71)\\
&\qquad \bar{\Delta \mathbf{C}}_{ik}^{\mathsf{T}} \left(\Delta \mathbf{C}_{ik} \left(\tilde{\mathbf{a}}_B - \boldsymbol{\beta}_{B_i}^{a} - \boldsymbol{\eta}_B^{a}\right) - \bar{\Delta \mathbf{C}}_{ik} \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^{a}\right)\right) && (3.72)\\
&= -\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^{g}\right)^{\times} \bar{\Delta \mathbf{C}}_{ik}^{\mathsf{T}} \left(\Delta \mathbf{v}_{ik} - \bar{\Delta \mathbf{v}}_{ik}\right) + \delta \Delta \mathbf{C}_{ik} \left(\tilde{\mathbf{a}}_B - \boldsymbol{\beta}_{B_i}^{a} - \boldsymbol{\eta}_B^{a}\right) - \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^{a}\right) && (3.73)\\
&= -\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^{g}\right)^{\times} \delta \Delta \mathbf{v}_{ik} + \delta \Delta \mathbf{C}_{ik} \left(\tilde{\mathbf{a}}_B - \boldsymbol{\beta}_{B_i}^{a} - \boldsymbol{\eta}_B^{a}\right) - \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^{a}\right) && (3.74)\\
&= -\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^{g}\right)^{\times} \delta \Delta \mathbf{v}_{ik} + \left(\mathbf{1} + \delta \Delta \boldsymbol{\phi}_{ik}^{\times}\right) \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^{a} - \delta \boldsymbol{\beta}_{B_i}^{a} - \delta \boldsymbol{\eta}_B^{a}\right) - \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^{a}\right) && (3.75)\\
&= -\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^{g}\right)^{\times} \delta \Delta \mathbf{v}_{ik} + \delta \Delta \boldsymbol{\phi}_{ik}^{\times} \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^{a}\right) - \delta \boldsymbol{\beta}_{B_i}^{a} - \delta \boldsymbol{\eta}_B^{a} && (3.76)\\
&= -\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^{g}\right)^{\times} \delta \Delta \mathbf{v}_{ik} - \left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^{a}\right)^{\times} \delta \Delta \boldsymbol{\phi}_{ik} - \delta \boldsymbol{\beta}_{B_i}^{a} - \delta \boldsymbol{\eta}_B^{a}, && (3.77)
\end{aligned}$$

$$\begin{aligned}
\delta \dot{\Delta \mathbf{r}}_{ik} &= \dot{\bar{\Delta \mathbf{C}}}_{ik}^{\mathsf{T}} \left(\Delta \mathbf{r}_{ik} - \bar{\Delta \mathbf{r}}_{ik}\right) + \bar{\Delta \mathbf{C}}_{ik}^{\mathsf{T}} \left(\dot{\Delta \mathbf{r}}_{ik} - \dot{\bar{\Delta \mathbf{r}}}_{ik}\right), && (3.78)\\
&= \left(\Delta \mathbf{C}_{ik} \left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^{g}\right)^{\times}\right)^{\mathsf{T}} \left(\Delta \mathbf{r}_{ik} - \bar{\Delta \mathbf{r}}_{ik}\right) + \bar{\Delta \mathbf{C}}_{ik}^{\mathsf{T}} \left(\Delta \mathbf{v}_{ik} - \bar{\Delta \mathbf{v}}_{ik}\right) && (3.79)\\
&= -\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^{g}\right)^{\times} \bar{\Delta \mathbf{C}}_{ik}^{\mathsf{T}} \left(\Delta \mathbf{r}_{ik} - \bar{\Delta \mathbf{r}}_{ik}\right) + \delta \Delta \mathbf{v}_{ik} && (3.80)\\
&= -\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^{g}\right)^{\times} \delta \Delta \mathbf{r}_{ik} + \delta \Delta \mathbf{v}_{ik}. && (3.81)
\end{aligned}$$

Combining the linearized error dynamics yields

$$\dot{\Delta \boldsymbol{\xi}}_{ik} = \mathbf{F}_{ik} \Delta \boldsymbol{\xi}_{ik} + \mathbf{L}_{ik} \boldsymbol{\eta}, \tag{3.82}$$

where

$$\mathbf{F}_{ik} = \begin{bmatrix} -\mathrm{ad}_{\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^g\right)} & & & & -\mathbf{1} \\ -\left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^a\right)^\times & -\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^g\right)^\times & & -\mathbf{1} & \\ & \mathbf{1} & -\left(\tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^g\right)^\times & & \\ & & & & \\ & & & & \end{bmatrix}, \tag{3.83}$$

$$\mathbf{L}_{ik} = \begin{bmatrix} & -\mathbf{1} & & \\ -\mathbf{1} & & & \\ & & & \\ & & \mathbf{1} & \\ & & & \mathbf{1} \end{bmatrix}. \tag{3.84}$$

Notice $\mathbf{F}_{ik}$ and $\mathbf{L}_{ik}$ no longer depends on $\Delta\mathbf{C}_{ik}$ which may result in more accurate Jacobian matrices.

3.1.4 Preintegrated Measurement Covariance

Under the zero-order hold assumption, the Jacobian matrices $\mathbf{F}_{ik}$ and $\mathbf{L}_{ik}$ are constant over the integration period, such that $\mathbf{F}_{d_{ik}} = \exp\left(\Delta t \mathbf{F}_{ik}\right)$. From the linearized RMI model and given the continuous-time covariance $\mathbf{Q}$ associated with the raw IMU measurement noise, it is possible to compute the preintegrated measurement covariance iteratively using [15, Ch. 4.7],

$$\boldsymbol{\Sigma}_{ik+1} = \mathbf{F}_{d_{ik}} \boldsymbol{\Sigma}_{ik} \mathbf{F}_{d_{ik}} + \mathbf{Q}_d, \tag{3.85}$$

where $\boldsymbol{\Sigma}_{ii} = \mathbf{0}$ and

$$\mathbf{Q}_d = \int_{t_i}^{t_j} \mathbf{F}_{d_{i\tau}} \mathbf{L}_{i\tau}\left(\tau\right) \mathbf{Q} \mathbf{L}_{i\tau}\left(\tau\right)^\mathsf{T} \mathbf{F}_{d_{i\tau}}^\mathsf{T} \mathrm{d}\tau. \tag{3.86}$$

The integral can be solved by matrix exponential using Van Loan's method. It is shown that the exponential of the matrix

$$\boldsymbol{\Xi} = \begin{bmatrix} -\mathbf{F} & \mathbf{L}\mathbf{Q}\mathbf{L}^\mathsf{T} \\ \mathbf{0} & \mathbf{F}^\mathsf{T} \end{bmatrix} \Delta t \tag{3.87}$$

is

$$\boldsymbol{\Psi} = \exp\left(\boldsymbol{\Xi}\right) = \begin{bmatrix} * & \mathbf{F}_{d_{ik}}^{-1}\mathbf{Q}_d \\ \mathbf{0} & \mathbf{F}_{d_{ik}}^{\mathsf{T}} \end{bmatrix}. \tag{3.88}$$

This discretization assumes that $\mathbf{F}$ are $\mathbf{Q}$ are slowly time varying and are constant over the time of integration between two measurements, which is not an exact solution of (3.86). Hence, for IMU measurements at lower frequency, one may consider solving (3.86) using a higher order numerical integration.

3.1.5 Bias Update

During optimization, the states are updated at every iteration. The RMIs are not functions of robot navigation states, however they are still functions of IMU sensor biases. Biases are part of the optimization variables, which means they are likely to vary during the optimization. The bias update at every iteration must be incorporated into the RMI somehow. In this section, the RMI with bias updates are discussed.

Biases are usually very small in its magnitude, and they are updated using the previously defined error definition (3.49)

$$\hat{\boldsymbol{\beta}}^{u}_{B_i} = \bar{\boldsymbol{\beta}}^{u}_{B_i} + \delta\boldsymbol{\beta}^{u}_{B_i}, \tag{3.89}$$

where $\hat{\boldsymbol{\beta}}^{u}_{B_i}$ is the updated bias. Thus, corresponding change must be applied to the RMI as well.

Consider the RMI with nominal bias estimates

$$\hat{\Delta\mathbf{C}}_{ik} = \operatorname{Exp}\left(\int_{\tau\in[i,k]} \tilde{\boldsymbol{\omega}}^{BW}_{B} - \hat{\boldsymbol{\beta}}^{g}_{B_i}\mathrm{d}\tau\right), \tag{3.90}$$

$$\hat{\Delta\mathbf{v}}_{ik} = \int_{\tau\in[i,k]} \hat{\Delta\mathbf{C}}_{i\tau}\left(\tilde{\mathbf{a}}_B - \hat{\boldsymbol{\beta}}^{a}_{B_i}\right)\mathrm{d}\tau, \tag{3.91}$$

$$\hat{\Delta\mathbf{r}}_{ik} = \iint_{\tau\in[i,k]} \hat{\Delta\mathbf{C}}_{i\tau}\left(\tilde{\mathbf{a}}_B - \hat{\boldsymbol{\beta}}^{a}_{B_i}\right)\mathrm{d}\tau^2. \tag{3.92}$$

Applying Taylor-series expansion to the RMI with respect to the biases results in

$$\hat{\Delta \mathbf{C}}_{ik} \approx \operatorname{Exp}\left(\int_{\tau\in[i,k]} \tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^g - \delta\boldsymbol{\beta}_{B_i}^g \mathrm{d}\tau\right) \tag{3.93}$$

$$= \operatorname{Exp}\left(\int_{\tau\in[i,k]} \tilde{\boldsymbol{\omega}}_B^{BW} - \bar{\boldsymbol{\beta}}_{B_i}^g \mathrm{d}\tau\right) \operatorname{Exp}\left(\int_{\tau\in[i,k]} -\mathbf{J}_r\left(\boldsymbol{\tau}\right)\delta\boldsymbol{\beta}_{B_i}^g \mathrm{d}\tau\right) \tag{3.94}$$

$$= \bar{\Delta \mathbf{C}}_{ik} \operatorname{Exp}\left(\left.\frac{\partial \Delta \mathbf{C}_{ik}}{\partial \boldsymbol{\beta}_{B_i}^g}\right|_{\bar{\boldsymbol{\beta}}_{B_i}^g} \delta\boldsymbol{\beta}_{B_i}^g\right), \tag{3.95}$$

$$\hat{\Delta \mathbf{v}}_{ik} \approx \int_{\tau\in[i,k]} \bar{\mathbf{C}}_{i\tau} \operatorname{Exp}\left(\left.\frac{\partial \Delta \mathbf{C}_{i\tau}}{\partial \boldsymbol{\beta}_{B_i}^g}\right|_{\bar{\boldsymbol{\beta}}_{B_i}^g} \delta\boldsymbol{\beta}_{B_i}^g\right)\left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^a - \delta\boldsymbol{\beta}_{B_i}^a\right)\mathrm{d}\tau \tag{3.96}$$

$$= \int_{\tau\in[i,k]} \bar{\Delta \mathbf{C}}_{i\tau}\left(\mathbf{1} + \left(\left.\frac{\partial \Delta \mathbf{C}_{i\tau}}{\partial \boldsymbol{\beta}_{B_i}^g}\right|_{\bar{\boldsymbol{\beta}}_{B_i}^g} \delta\boldsymbol{\beta}_{B_i}^g\right)^{\times}\right)\left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^a - \delta\boldsymbol{\beta}_{B_i}^a\right)\mathrm{d}\tau \tag{3.97}$$

$$= \int_{\tau\in[i,k]} \bar{\Delta \mathbf{C}}_{i\tau}\left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^a\right)\mathrm{d}\tau - \int_{\tau\in[i,k]} \bar{\Delta \mathbf{C}}_{i\tau}\left(\tilde{\mathbf{a}}_B - \bar{\boldsymbol{\beta}}_{B_i}^a\right)^{\times} \left.\frac{\partial \Delta \mathbf{C}_{i\tau}}{\partial \boldsymbol{\beta}_{B_i}^g}\right|_{\bar{\boldsymbol{\beta}}_{B_i}^g} \delta\boldsymbol{\beta}_{B_i}^g \mathrm{d}\tau - \tag{3.98}$$

$$\int_{\tau\in[i,k]} \bar{\Delta \mathbf{C}}_{i\tau}\left(\left.\frac{\partial \Delta \mathbf{C}_{i\tau}}{\partial \boldsymbol{\beta}_{B_i}^g}\right|_{\bar{\boldsymbol{\beta}}_{B_i}^g} \delta\boldsymbol{\beta}_{B_i}^g\right)^{\times} \delta\boldsymbol{\beta}_{B_i}^a \mathrm{d}\tau \tag{3.99}$$

$$= \bar{\Delta \mathbf{v}}_{ik} + \left.\frac{\partial \Delta \mathbf{v}_{ik}}{\partial \boldsymbol{\beta}_{B_i}^a}\right|_{\bar{\boldsymbol{\beta}}_{B_i}^a} \delta\boldsymbol{\beta}_{B_i}^a + \left.\frac{\partial \Delta \mathbf{v}_{ik}}{\partial \boldsymbol{\beta}_{B_i}^g}\right|_{\bar{\boldsymbol{\beta}}_{B_i}^g} \delta\boldsymbol{\beta}_{B_i}^g. \tag{3.100}$$

The bias update for position has a similar form of equation as the velocity

$$\hat{\Delta \mathbf{r}}_{ik} \approx \bar{\Delta \mathbf{r}}_{ik} + \left.\frac{\partial \Delta \mathbf{r}_{ik}}{\partial \boldsymbol{\beta}_{B_i}^a}\right|_{\bar{\boldsymbol{\beta}}_{B_i}^a} \delta\boldsymbol{\beta}_{B_i}^a + \left.\frac{\partial \Delta \mathbf{r}_{ik}}{\partial \boldsymbol{\beta}_{B_i}^g}\right|_{\bar{\boldsymbol{\beta}}_{B_i}^g} \delta\boldsymbol{\beta}_{B_i}^g. \tag{3.101}$$

The bias Jacobian matrices can be computed recursively with the initial Jacobian $\mathbf{J}_i = \mathbf{1}$,

$$\mathbf{J}_{i+1} = \mathbf{F}_{d_{ii+1}} \mathbf{J}_i. \tag{3.102}$$

Notice the bias correction is only a first-order approximation. This is possible because the bias change during the optimization is assumed to be small. On the other hand, the update is not incorporated in the bias Jacobian matrices. They remain constant and are precomputed during the preintegration.

3.1.6 Preintegrated IMU Factor

Recall that the preintegrated measurements are considered as new inputs to the process model. Preintegrated measurements and their noise covariance can be computed without any robot state information using the relative motion increment. Incorporating the bias update to the kinematic equations (3.24), (3.25), and (3.26) yields

$$\begin{aligned}
\mathbf{C}_{WB_j} &= \mathbf{C}_{WB_i}\Delta\hat{\mathbf{C}}_{ij}, && (3.103)\\
\mathbf{v}_W^{b_jw/W} &= \mathbf{v}_W^{b_iw/W} + \Delta t_{ij}\mathbf{g}_W + \mathbf{C}_{WB_i}\Delta\hat{\mathbf{v}}_{ij}, && (3.104)\\
\mathbf{r}_W^{b_jw} &= \mathbf{r}_W^{b_iw} + \Delta t_{ij}\mathbf{v}_W^{b_iw/W} + \tfrac{1}{2}\Delta t_{ij}^2\mathbf{g}_W + \mathbf{C}_{WB_i}\Delta\hat{\mathbf{r}}_{ij}. && (3.105)
\end{aligned}$$

Now, the residual errors associated with the IMU can be constructed as

$$\mathbf{e}_{\mathcal{I},i} = \begin{bmatrix}\mathbf{e}_{\Delta\mathbf{C}_{ij}}^{\mathsf{T}} & \mathbf{e}_{\Delta\mathbf{v}_{ij}}^{\mathsf{T}} & \mathbf{e}_{\Delta\mathbf{r}_{ij}}^{\mathsf{T}} & \mathbf{e}_{\boldsymbol{\beta}_{ij}^a}^{\mathsf{T}} & \mathbf{e}_{\boldsymbol{\beta}_{ij}^g}^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}}, \quad (3.106)$$

where

$$\begin{aligned}
\mathbf{e}_{\Delta\mathbf{C}_{ij}} &\triangleq \operatorname{Log}\left(\Delta\hat{\mathbf{C}}_{ij}^{\mathsf{T}}\mathbf{C}_{WB_i}^{\mathsf{T}}\mathbf{C}_{WB_j}\right), && (3.107)\\
\mathbf{e}_{\Delta\mathbf{v}_{ij}} &\triangleq \mathbf{C}_{WB_i}^{\mathsf{T}}\left(\mathbf{v}_W^{b_jw/W} - \mathbf{v}_W^{b_iw/W} - \Delta t_{ij}\mathbf{g}_W\right) - \Delta\hat{\mathbf{v}}_{ij}, && (3.108)\\
\mathbf{e}_{\Delta\mathbf{r}_{ij}} &\triangleq \mathbf{C}_{WB_i}^{\mathsf{T}}\left(\mathbf{r}_W^{b_jw} - \mathbf{r}_W^{b_iw} - \Delta t_{ij}\mathbf{v}_W^{b_iw/W} - \tfrac{1}{2}\Delta t_{ij}^2\mathbf{g}_W\right) - \Delta\hat{\mathbf{r}}_{ij}, && (3.109)\\
\mathbf{e}_{\boldsymbol{\beta}_{ij}^a} &\triangleq \boldsymbol{\beta}_{B_j}^a - \boldsymbol{\beta}_{B_i}^a, && (3.110)\\
\mathbf{e}_{\boldsymbol{\beta}_{ij}^g} &\triangleq \boldsymbol{\beta}_{B_j}^g - \boldsymbol{\beta}_{B_i}^g. && (3.111)
\end{aligned}$$

The IMU residual is linearized about an operating point $\bar{\mathbf{X}}$ using a first order Taylor-series expansion. The linearization can be done using two different error definitions mentioned in section 3.1.3.1 and in section 3.1.3.2. The preintegrated IMU factors constraint two robot states: the poses, velocities, and IMU sensor biases at $t = t_i$ and $t = t_j$.

3.2 Depth-based Vision Factor

Computer vision research has been around since 1970 to mimic human visual systems. Subsequently, this technology has been widely used in the robot perception and navigation community. Robot perception is one of the most important fields to achieve the complete autonomy of a robot. Sub-domains of computer vision such as scene recognition, object detection, object recognition, etc. are used in robot guidance and navigation systems. Particularly, the 3D pose estimation via feature tracking is performed in visual navigation systems.

In this section, Kanade-Lucas-Tomasi (KLT) sparse optical flow algorithm is presented using Shi-Tomasi corner features in Section 3.2.1 and in Section 3.2.2 [18]. The image processing discussed in this section is based on Section 4 of [1]. Further, a depth-based vision factor is derived using the tracked features in Section 3.2.4.

3.2.1 Feature Extraction

A feature is an area of a digital image that contains meaningful information, such as a corner or edge. For example, corner features are the areas of an image where neighboring pixels have different intensities in all directions. Finding these corners was first done by [19]. Consider an equalized and undistorted grayscale image I with a pixel (x_0, y_0). The difference in intensity for a displacement of $(\delta x, \delta y)$ is expressed as

$$\sum_{\delta x, \delta y \in \mathcal{W}} \left(I\left(x_0, y_0\right) - I\left(x_0 + \delta x, y_0 + \delta y\right)\right)^2, \tag{3.112}$$

where $\mathcal{W}$ is the patch of an image. It is likely a corner if the difference is large. The neighboring pixels can be approximated using a first order Taylor-series expansion

$$I\left(x_0 + \delta x, y_0 + \delta y\right) \approx I\left(x_0, y_0\right) + \left.\frac{\partial I}{\partial x}\right|_{x_0, y_0} \delta x + \left.\frac{\partial I}{\partial y}\right|_{x_0, y_0} \delta y. \tag{3.113}$$

Thus, (3.112) can be rewritten as

$$\begin{aligned}
\sum_{\delta x, \delta y \in \mathcal{W}} \left(I\left(x_0, y_0\right) - I\left(x_0 + \delta x, y_0 + \delta y\right)\right)^2 &\approx \sum \left(\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y\right)^2 && (3.114)\\
&= \sum \left(\begin{bmatrix}\delta x & \delta y\end{bmatrix} \begin{bmatrix}\frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y}\end{bmatrix} \begin{bmatrix}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}\end{bmatrix} \begin{bmatrix}\delta x \\ \delta y\end{bmatrix}\right) && (3.115)\\
&= \begin{bmatrix}\delta x & \delta y\end{bmatrix} \left(\sum \begin{bmatrix}\frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y}\end{bmatrix} \begin{bmatrix}\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}\end{bmatrix}\right) \begin{bmatrix}\delta x \\ \delta y\end{bmatrix} && (3.116)\\
&= \begin{bmatrix}\delta x & \delta y\end{bmatrix} \mathbf{M} \begin{bmatrix}\delta x \\ \delta y\end{bmatrix}, && (3.117)
\end{aligned}$$

where

$$\mathbf{M} = \begin{bmatrix} \sum \left(\frac{\partial I}{\partial x}\right)^2 & \sum \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \sum \left(\frac{\partial I}{\partial y}\right)\left(\frac{\partial I}{\partial x}\right) & \sum \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \tag{3.118}$$

is known as the second moment matrix. A corner is determined by analyzing the eigenvalues of this matrix in the following way:

1. if $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$ then (x_0, y_0) has no features,
2. if $\lambda_1 \approx 0$ and λ_2 has some large positive value, then (x_0, y_0) is an edge,
3. if λ_1 and λ_2 have large positive values, then (x_0, y_0) is a corner.

The Harris corner detector has this criteria where both eigenvalues of $\mathbf{M}$ has to meet a naive criteria, where the eigenvalues satisfy

$$\lambda_1\lambda_2 - k\left(\lambda_1 + \lambda_2\right)^2 > \text{threshold}, \tag{3.119}$$

for small values of k. Shi-Tomasi corner features are based on the Harris corner detector with a slight variation in different response function [20]. Shi-Tomasi detector has the following evaluation criteria,

$$\min\left(\lambda_1, \lambda_2\right) > \text{threshold}. \tag{3.120}$$

The points in image where (3.120) holds as a feature candidate. For implementation, the Shi-Tomasi corner detection algorithm from `openCV` is used [21]. The algorithm finds the N strongest corners in the grayscale image with corresponding quality levels. Then the algorithm finds the strongest corners around the neighborhood and provides the minimum Euclidean distance between corners such that the features are sparse within the image.

3.2.2 Feature Tracking

To track features over a series of images, the extracted features from one image must be matched to the features of subsequent images. The features are matched by comparing the feature descriptors. This means that the intensities of the tracking features must not change between the two frames being compared, and the neighboring pixels, namely the descriptor as a whole, have a similar motion. The KLT feature tracker is used to track extracted Shi-Tomasi corner features.

The KLT method takes a patch of the initial image around the feature point, for example a 3×3 patch resulting in 9 neighboring points, and uses them to find the corresponding feature point on the subsequent image. Consider a 1D problem with a feature point x_0 in an image $J(x)$. A corresponding point $x_0 + h$ in a subsequent image $I(x)$ such that $I(x_0 + h) \approx J(x_0)$

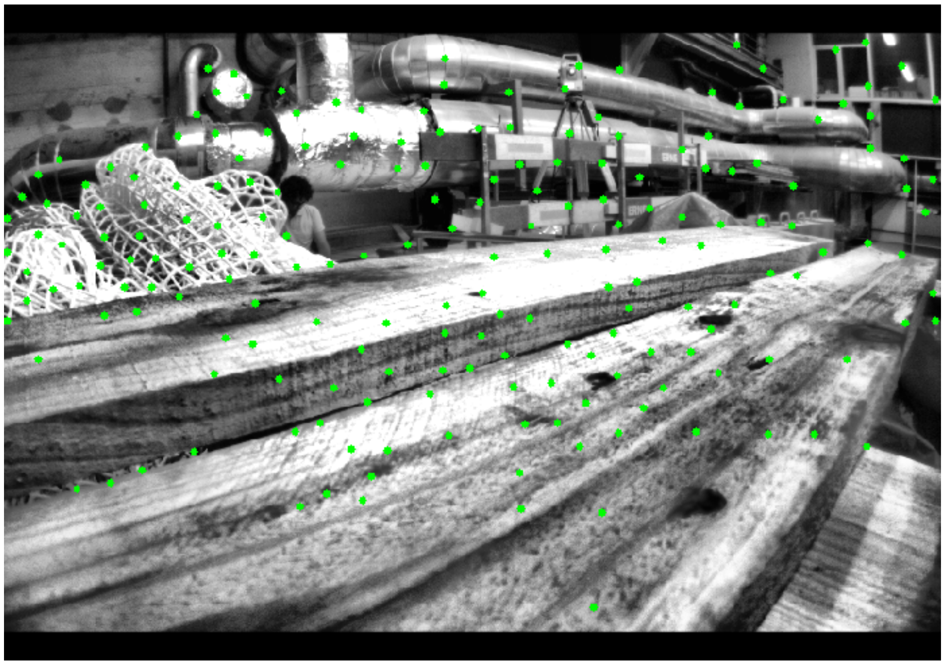


Figure 3.1: Example of Shi-Tomasi corner feature detection shown in green using EuRoC Machine Hall 01 Dataset [22].

can be found by minimizing the following cost function

$$\sum_{x \in N} \left(I\left(x+h\right) - J\left(x\right)\right)^2, \tag{3.121}$$

where N is the number of neighboring pixels around x_0. Based on the assumption that the image is blurred with the Gaussian function and the intensity is smooth, applying a first order Taylor series expansion on (3.121) yields

$$\sum_{x \in N} \left(I\left(x\right) - J\left(x\right) + h\frac{\mathrm{d}I\left(x\right)}{\mathrm{d}x}\right)^2. \tag{3.122}$$

Because the corresponding point in the subsequent image $I\left(x+h\right)$ is relatively close to the initial point in the image $J\left(x\right)$, it is safe to assume that h is also very small. To find the corresponding point, the value of h that minimizes the cost function is obtained by taking the derivative with respect to h and setting it to 0,

$$\sum_{x \in N} \left(J\left(x\right) - I\left(x\right) + h\frac{\mathrm{d}I\left(x\right)}{\mathrm{d}x}\right)\frac{\mathrm{d}I\left(x\right)}{\mathrm{d}x} = 0. \tag{3.123}$$

The intensities of an image are usually non-linear, which means in practice, the feature tracking process is done iteratively.

Now consider a 2D problem with 2D images $I(x, y)$ and $J(x, y)$. The cost function becomes

$$\sum_{(x,y)\in N} \left(I\left(x + h_x, y + h_y\right) - J(x, y)\right)^2. \tag{3.124}$$

By applying a first order Taylor series expansion, the cost function is

$$\sum_{(x,y)\in N} \left(I(x, y) - J(x, y) + \frac{\partial I(x, y)}{\partial x} h_x + \frac{\partial I(x, y)}{\partial y} h_y\right)^2. \tag{3.125}$$

Taking the derivative with respect to h_x and h_y gives

$$\begin{bmatrix} \sum \left(\frac{\partial I}{\partial x}\right)^2 & \sum \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \sum \left(\frac{\partial I}{\partial y}\right)\left(\frac{\partial I}{\partial x}\right) & \sum \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \begin{bmatrix} h_x \\ h_y \end{bmatrix} = - \begin{bmatrix} \sum \left(I(x, y) - J(x, y)\right) \frac{\partial I}{\partial x} \\ \sum \left(I(x, y) - J(x, y)\right) \frac{\partial I}{\partial y} \end{bmatrix}, \tag{3.126}$$

where the matrix on the left-hand side is equivalent to the second moment matrix. Since the eigenvalues of the second moment matrix are non-zero around corners, the image registration can be solved. The KLT tracker outputs points in a target image $I(x, y)$ that correspond to corner features in a reference image $J(x, y)$. These corresponding points can now be utilized to form a constraint between the two frames.

3.2.3 Outlier Rejection

Consider $\mathbf{x}_1$ and $\mathbf{x}_2$ are the corresponding points in the first and the second images respectively. Given an epipolar geometry, the fundamental matrix $\mathbf{F}$ relates corresponding points in two image frames as

$$\mathbf{x}_2^{\mathsf{T}} \mathbf{F} \mathbf{x}_1 = 0, \tag{3.127}$$

where

$$\mathbf{F} \triangleq \mathbf{K}^{-\mathsf{T}} \mathbf{E} \mathbf{K}^{-1}, \tag{3.128}$$

$\mathbf{K}$ is the camera intrinsics, and $\mathbf{E}$ is the essential matrix. The fundamental matrix is estimated using an eight point algorithm [23, Ch. 11]. Random sample consensus (RANSAC) is used to reject outliers using the algorithm 11.4 in [23, Ch. 11.6].

3.2.4 Depth-based Projection Factor

The features extracted and tracked are all resolved in the pixel coordinates in the image plane. To use this information for pose estimation, the relationship between the image plane and the physical world must be known. In the following section, the basic principles of a pinhole camera are discussed. Lastly, the depth-based projection factor formulation is derived.

3.2.4.1 Pinhole Camera

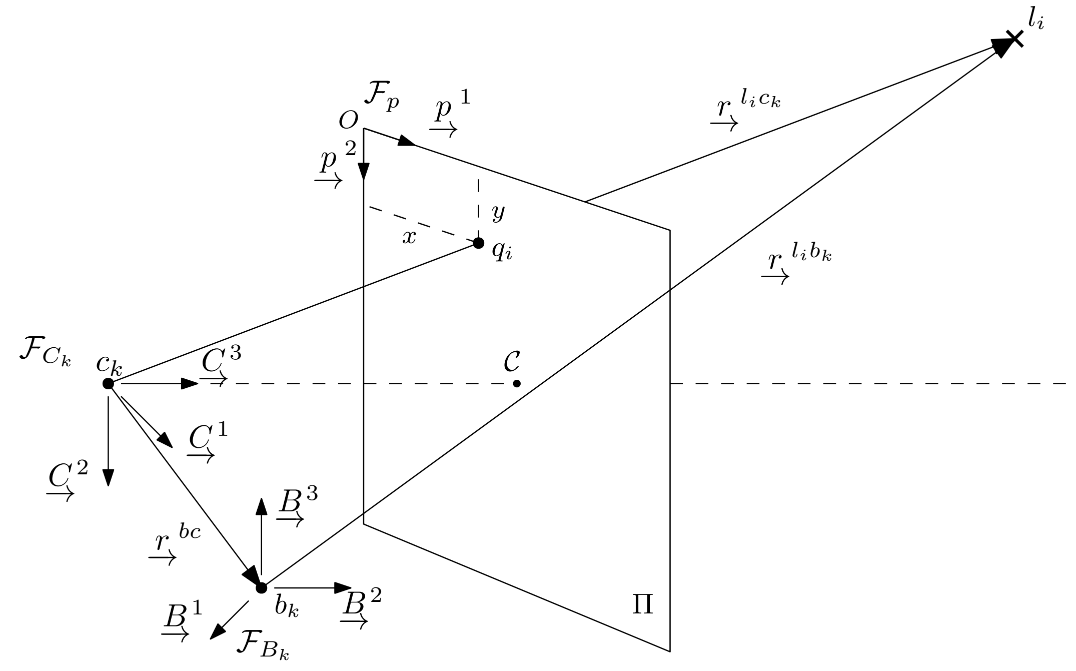
A perspective projection model of a pinhole camera is shown in Figure 3.2. The feature extraction and tracking are performed on the image plane Π in pixel coordinates. The center of a robot's body is denoted as b_k while $\mathcal{F}_{B_k}$ is the frame associated with it. The position of the camera, also known as the optical center, is denoted as c_k with the camera frame $\mathcal{F}_{C_k}$. The image plane has a reference point O with a 2D pixel frame $\mathcal{F}_p$. Consider a feature point l_i in a physical space that is projected onto the image plane as q_i. The image plane is at the camera's focal length f. Thus, the feature in the 3D space is projected onto the plane using

$$\mathbf{r}_P^{q_i\mathcal{C}} = \left(\frac{x}{z}f, \frac{y}{z}f\right), \tag{3.129}$$

where (x, y, z) is the position of the features relative to the optical center resolved in the camera frame, $\mathbf{r}_{C_k}^{l_i c_k}$, with $z > 0$. With the feature point on the image plane written in homogeneous coordinates as $\left(\frac{x}{z}f, \frac{y}{z}f, 1\right)$, the above equation can be rewritten in matrix form as

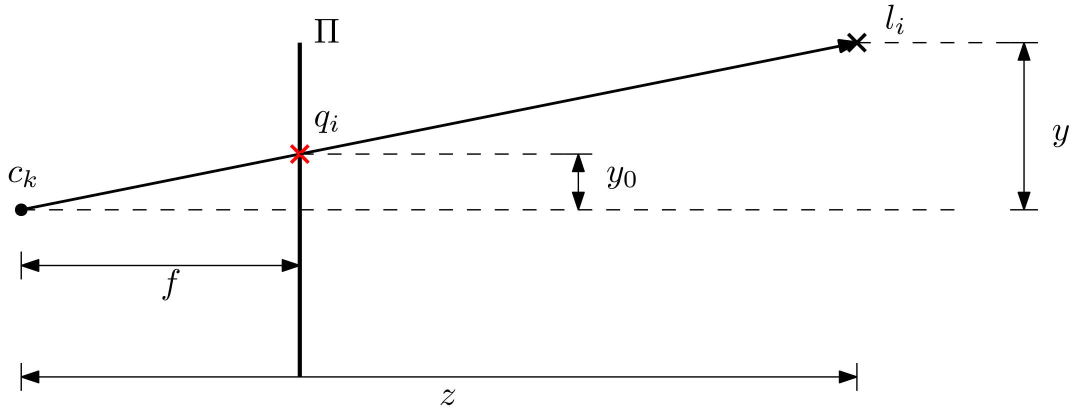
$$\begin{bmatrix} xf \\ xf \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{3.130}$$

However, the projected feature point on an image plane has to be transformed into finite discontinuous pixel coordinates rather than a physical position. Thus, a scaling factor s is involved. Further, the reference point of the image plane is at point o and thus the translation of the reference point to the image center $\mathbf{r}_P^{O\mathcal{C}} = \begin{bmatrix} c_x & c_y \end{bmatrix}^\mathsf{T}$ is added. The camera calibration



l_i
$\mathcal{F}_p$
O
$\underrightarrow{p}^{1}$
$\underrightarrow{p}^{2}$
$\underrightarrow{r}^{l_i c_k}$
y
x
q_i
$\underrightarrow{r}^{l_i b_k}$
$\mathcal{F}_{C_k}$
c_k
$\underrightarrow{C}^{3}$
$\mathcal{C}$
$\underrightarrow{C}^{1}$
$\underrightarrow{C}^{2}$
$\underrightarrow{B}^{3}$
$\underrightarrow{r}^{bc}$
b_k
$\underrightarrow{B}^{2}$
$\underrightarrow{B}^{1}$
$\mathcal{F}_{B_k}$
Π

(a) Geometrical representation of an perspective projection model [3].



(b) Top view of the perspective projection model.

Figure 3.2: Perspective projection model of a pinhole camera.

matrix $\mathbf{K}$ can then be written as

$$\mathbf{K} = \begin{bmatrix} sf & 0 & c_x \\ 0 & sf & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.131}$$

3.2.4.2 Unit Sphere Model

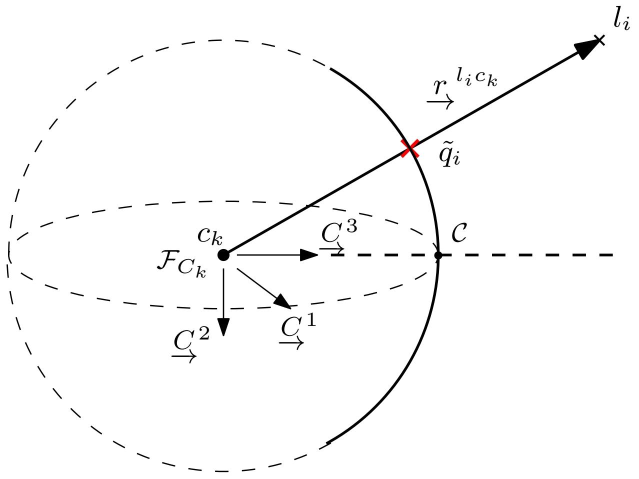


Figure 3.3: Geometrical representation of normalized image coordinates.

Carrying over the concepts of the pinhole camera model, this section defines the camera measurement on a unit sphere such that the measurement model is not constrained to a pinhole camera but can be utilized for any other optical device such as wide-angle, fisheye or omnidirectional cameras. The tracked features are undistorted and back projected onto a unit sphere from an image plane. The normalized features are stored as a unit vector $\tilde{q}_i$ on the sphere with inverse of its depth $\lambda_i = \frac{1}{z}$. The back projection function

$$\pi_c^{-1}\left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}\right) = \text{undistort}\left(\mathbf{K}^{-1}\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}\right) \tag{3.132}$$

can be used to recover the feature location in the physical world from the corresponding feature point in an image plane

$$\mathbf{r}_{C_k}^{l_i c_k} = \frac{1}{\lambda_i}\pi_c^{-1}\left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}\right). \tag{3.133}$$

3.2.4.3 Depth-based Projection Factor

Consider a feature l_k tracked in two consecutive images taken at $t = t_i$ and $t = t_j$. Given a feature at t_i and an initial estimate of a robot's state at both time t_i and t_j, the feature location at t_j can be estimated using

$$
\begin{aligned}
\check{\mathbf{r}}_{C_j}^{l_k c_j} &= \mathbf{C}_{C_i C_j}^{\mathsf{T}} \left(\mathbf{r}_{C_i}^{l_k c_i} - \mathbf{r}_{C_i}^{c_j c_i} \right) && (3.134) \\
&= \mathbf{C}_{C_i C_j}^{\mathsf{T}} \left(\frac{1}{\lambda_k} \pi_c^{-1} \left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}_i \right) - \mathbf{r}_{C_i}^{c_j c_i} \right) && (3.135) \\
&= \mathbf{C}_{BC}^{\mathsf{T}} \left(\mathbf{C}_{WB_j}^{\mathsf{T}} \left(\mathbf{C}_{WB_i} \left(\mathbf{C}_{BC} \frac{1}{\lambda_k} \pi_c^{-1} \left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}_i \right) + \mathbf{r}_B^{bc} \right) + \mathbf{r}_W^{b_i w} - \mathbf{r}_W^{b_j w} \right) - \mathbf{r}_B^{bc} \right), && (3.136)
\end{aligned}
$$

where $\mathbf{C}_{BC}$ and $\mathbf{r}_B^{bc}$ are the camera extrinsic parameters that relate the transformation between the robot's body and the center of the camera. Therefore, the residual of the depth-based projection factor or the vision factor can be obtained by comparing the estimated feature location at t_j with the measured feature as

$$
\mathbf{e}_{\mathcal{C}} \triangleq \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix}^{\mathsf{T}} \left(\frac{\check{\mathbf{r}}_{C_j}^{l_k c_j}}{r_{C_j,3}^{l_k c_j}} - \pi_c^{-1} \left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}_j \right) \right), \tag{3.137}
$$

where $\begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix}$ are two arbitrarily selected orthogonal bases which span the tangent plane of $\pi_c^{-1} \left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}_j \right)$. A camera factor constrains two poses sharing a feature. The equation (3.136) is a function of camera extrinsic parameters and two robot body poses. The camera measurement residual will be used in an optimization problem investigated in Chapter 4.

3.3 LIDAR Point Cloud Registration Factor

A typical Light detection and ranging (LIDAR) sensor emits light in the form of pulsed laser into the surrounding and receives the reflected pulses. The sensor uses the time-of-flight (ToF) to measure the distance between the target object and the center of the sensor using

$$
D = \frac{c t_f}{2}, \tag{3.138}
$$

where D is the distance measured, t_f is the time of flight, and c is the speed of light. There are 3 categories of LIDAR systems: 1D, 2D, or 3D, and all of them share the time-of-flight concept. For a 1D laser scanner, a single laser emitter is used in a static position. A 2D system uses a single emitter in rotation to capture the a planar information of the surrounding. Moreover, a 3D LIDAR sensor functions like their 2D counterparts but multiple emitters are spread out along the vertical axis of the sensor. For example, the Velodyne Puck (VLP-16), a 3D LIDAR, has 16 scan rings, providing 16 layers of planar scans along the vertical axis. When the scan rings complete a full revolution in rotation, a complete 360 degree view of the surrounding, referred to as a *sweep*, is obtained. In this section, the procedures of using a LIDAR in a SLAM are discussed. The procedures are the processing of a 3D LIDAR measurements, feature extraction, and constructing a factor.

3.3.1 LIDAR Data Preprocessing

Unlike camera measurements where a complete image data is given at an instance, raw LIDAR point cloud measures only a subset of a sweep at a time, called a scan *packet*. A scan packet does not provide any meaningful information about the scene. Thus, it cannot be used for navigation. On the other hand, a sweep can be thought of as a 3D image of the surrounding that may contain some salient information about the scene. Acquiring a complete sweep requires preprocessing steps such as sweep accumulation, downsampling, and filtering. Preprocessing is necessary in order to proceed to futher data association like feature extraction and point cloud registration.

3.3.1.1 Sweep Accumulation

LIDAR points are measured by rotating laser beams. When the sensor receives the reflected beams once, a set of points, called a scan *packet*, is measured. As the laser beams rotate one revolution, packets are accumulated and complete a full 360 degree view of the environment. Consider a LIDAR carrying robot is in motion during packet accumulation. Although scan packets are usually measured at a very high frequency such that the transformation between two successive packets is negligible, when multiple packets are accumulated, the transformation between the first packet and the last becomes significantly large that the accumulated scans become distorted or skewed.

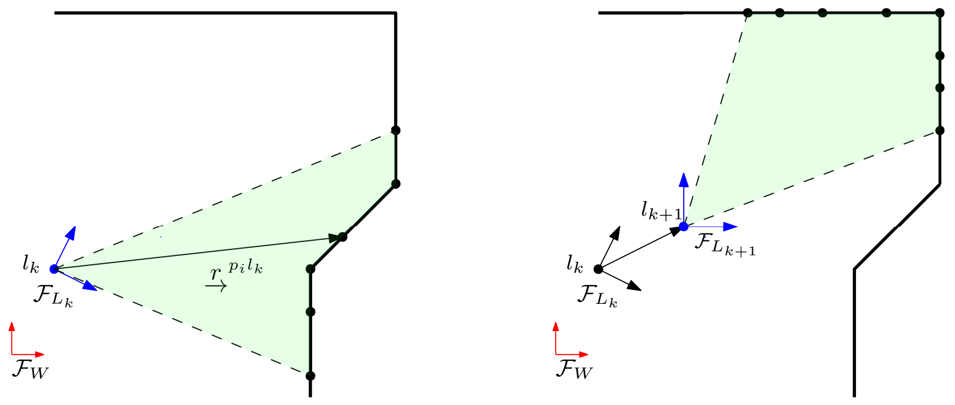


Figure 3.4: A 2D representation of range data accumulation process. Left figure shows scan packet at k. Right figure shows the subsequent scan packet at $k+1$ with the movement of a robot. Each dot represents a single point.

These scan packets are initially resolved in the sensor frame $\mathcal{L}_k$. They are transformed to the robot's body frame using a simple LIDAR extrinsic parameter

$$\mathbf{T}_{BL} = \begin{bmatrix} \mathbf{C}_{BL} & \mathbf{r}_B^{lb} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3.139}$$

to facilitate the further derivations. In practice, this extrinsic parameter is already known based on the 3D model of a robot, and can safely be assumed constant if the robot is considered as a rigid body. For cases where the extrinsic parameter is not given, they are sometimes estimated along with the states in the optimization problem.

The de-skewing process is essentially resolving the scan packets into a common frame to avoid distortion until scanning a complete revolution. This set of de-skewed packets resolved in a common sensor frame is called a *sweep*. A sweep is accumulated using scan packets and IMU sensor measurements. Figure 3.6 shows the sweep accumulation process. When multiple scan packets are measured between two IMU measurements, all the in-between scan packets are assumed to be measured at the same pose although in reality it is not necessarily the case. Every scan packet is assumed to be resolved in the current robot pose. The current robot pose is immediately updated at every IMU measurement using the robot kinematics. For example, a scan packet measured after an IMU measurement is resolved in a different frame than a scan packet measured before. Thus, the scan packets must be transformed to a local frame, usually the frame at which the first scan packet resolved

$$\mathbf{p}_{B_k}^{\mathcal{L}_{k+1} b_k} = \mathbf{T}_{WB_k}^{-1}\mathbf{T}_{WB_{k+1}}\mathbf{r}_{B_{k+1}}^{\mathcal{L}_{k+1} b_{k+1}}, \tag{3.140}$$

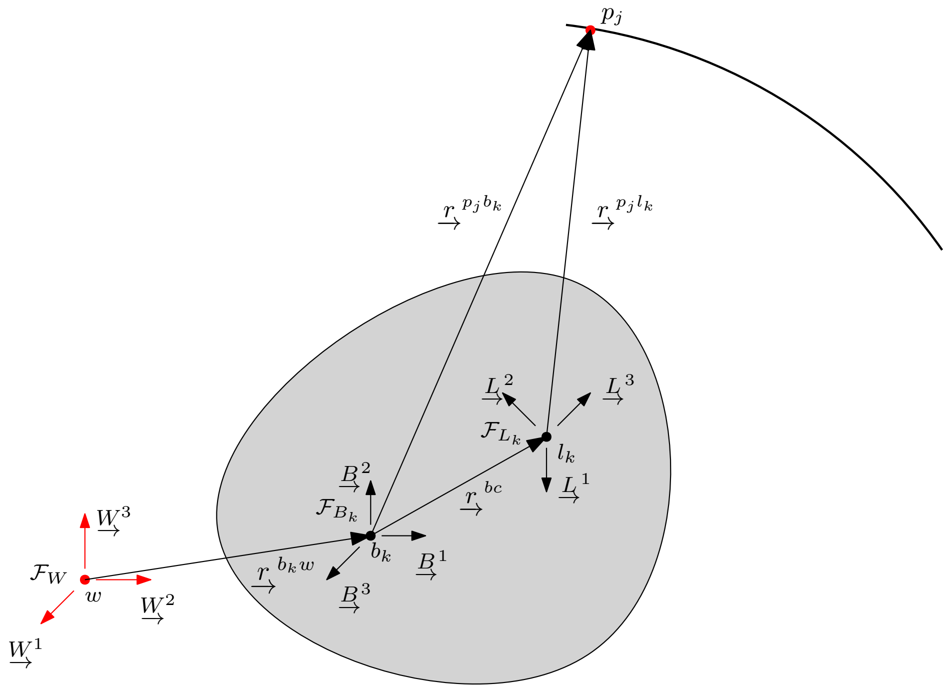


Figure 3.5: Geometric representation of the robot's body and the sensor.

where $\mathbf{T}_{WB_k}$ and $\mathbf{T}_{WB_{k+1}}$ in $SE(3)$ are the poses at $t = t_k$ and $t = t_{k+1}$ respectively. The locations of a set of points $\mathcal{L}_{k+1}$ relative to b_{k+1} and resolved in frame $\mathcal{F}_{B_{k+1}}$ are denoted as $\mathbf{r}_{B_k}^{\mathcal{L}_{k+1} b_k}$. After transformation the set of points are now relative to b_k and resolved in frame $\mathcal{F}_{B_k}$. This process is continued until a sweep is formed. Once the sweep is formed and resolved in a local frame, it is back transformed into the current robot pose.

Figure 3.7 shows a sweep of an office with and without de-skewing process. In Figure 3.7b, the warping is alleviated as the four walls form a rectangular shape unlike the scan in Figure 3.7a.

3.3.1.2 Point Cloud Downsampling

Downsampling a point cloud is essential to reduce computational complexity in point cloud processing. Thus, every incoming sweep is downsampled prior to further processing. A Voxel Filtering method is used in this thesis for its simplicity and effectiveness [24]. A Voxel filter uses a voxel grid to reduce the number of points, where a voxel grid is a set of 3D boxes in space that divides the point cloud into many subspaces. Then, all the points in a voxel are spatially averaged to a single point. The dimension of the voxel is given by a user, and the downsampling rate depends on the voxel size.

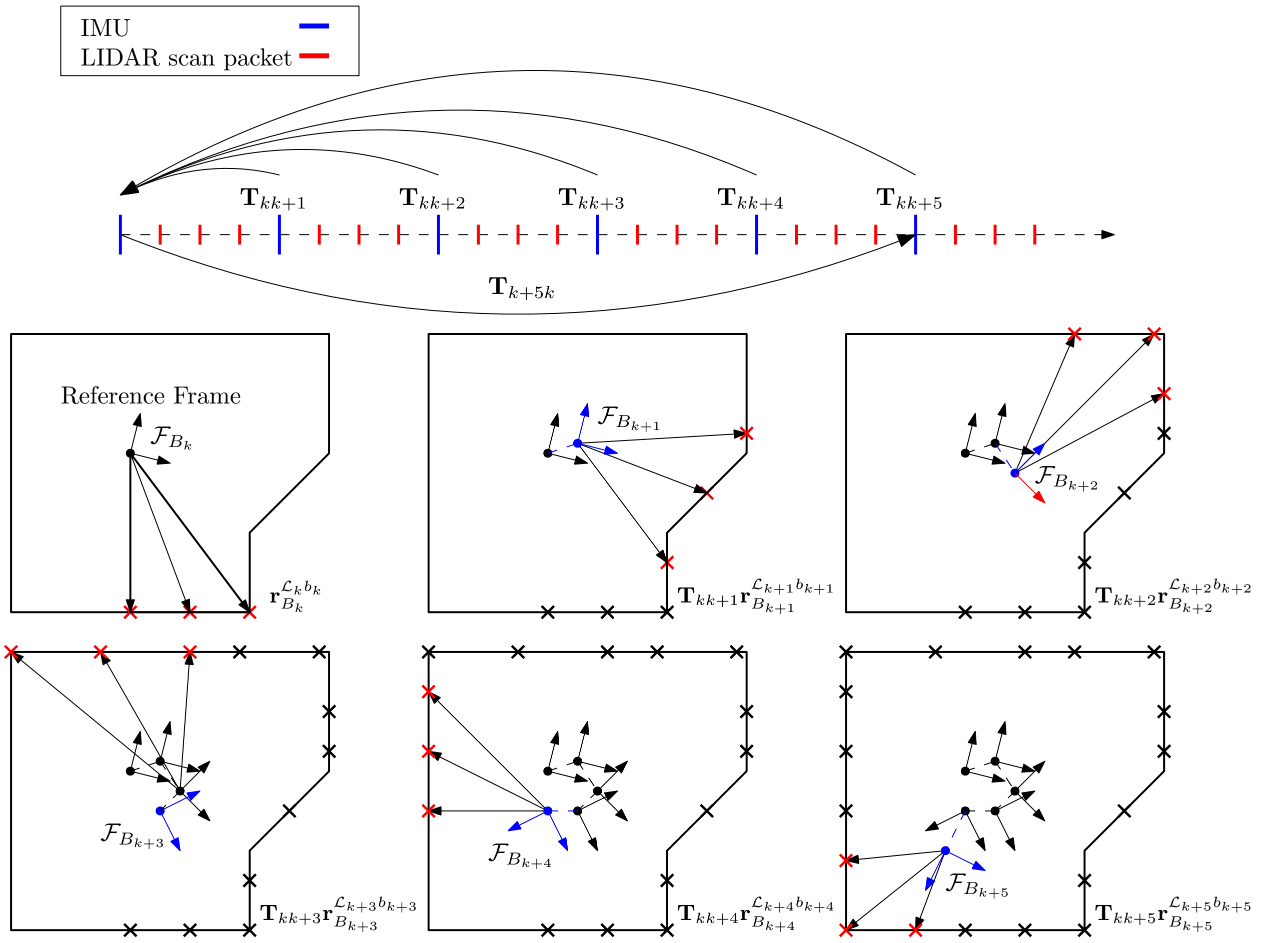
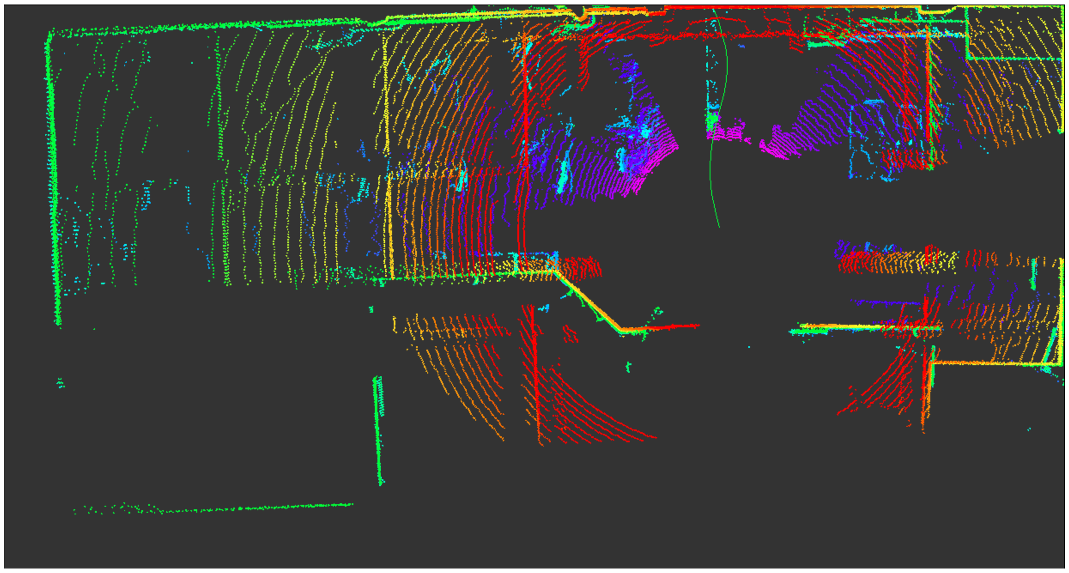
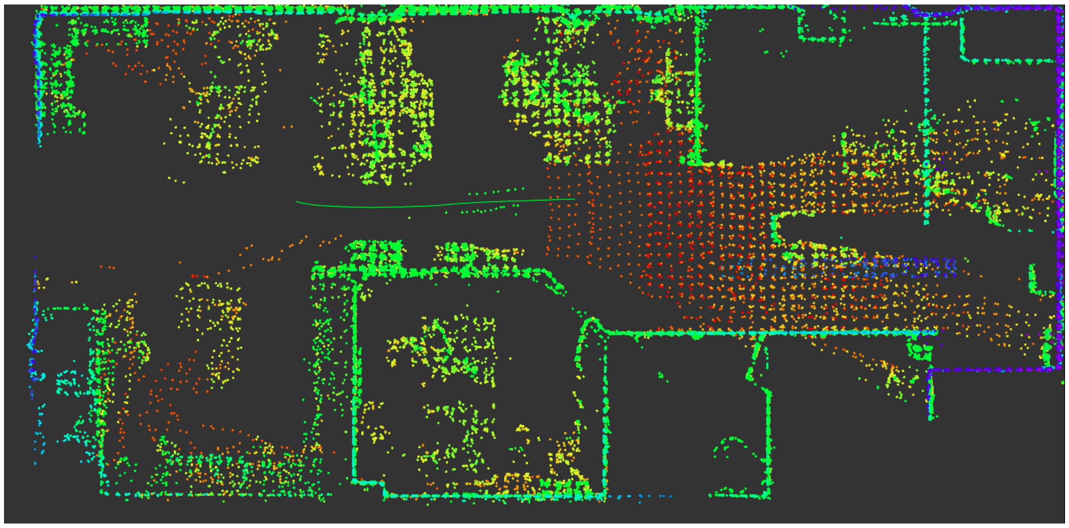

IMU
LIDAR scan packet
$\mathbf{T}_{kk+1}$
$\mathbf{T}_{kk+2}$
$\mathbf{T}_{kk+3}$
$\mathbf{T}_{kk+4}$
$\mathbf{T}_{kk+5}$
$\mathbf{T}_{k+5k}$
Reference Frame
$\mathcal{F}_{B_k}$
$\mathbf{r}_{B_k}^{\mathcal{L}_k b_k}$
$\mathcal{F}_{B_{k+1}}$
$\mathbf{T}_{kk+1}\mathbf{r}_{B_{k+1}}^{\mathcal{L}_{k+1} b_{k+1}}$
$\mathcal{F}_{B_{k+2}}$
$\mathbf{T}_{kk+2}\mathbf{r}_{B_{k+2}}^{\mathcal{L}_{k+2} b_{k+2}}$
$\mathcal{F}_{B_{k+3}}$
$\mathbf{T}_{kk+3}\mathbf{r}_{B_{k+3}}^{\mathcal{L}_{k+3} b_{k+3}}$
$\mathcal{F}_{B_{k+4}}$
$\mathbf{T}_{kk+4}\mathbf{r}_{B_{k+4}}^{\mathcal{L}_{k+4} b_{k+4}}$
$\mathcal{F}_{B_{k+5}}$
$\mathbf{T}_{kk+5}\mathbf{r}_{B_{k+5}}^{\mathcal{L}_{k+5} b_{k+5}}$

Figure 3.6: Graphical representation of de-skewing process is shown. Each cross around the wall represents a packet point cloud taken at different time. The robot's motion is captured only when IMU measurement is obtained.

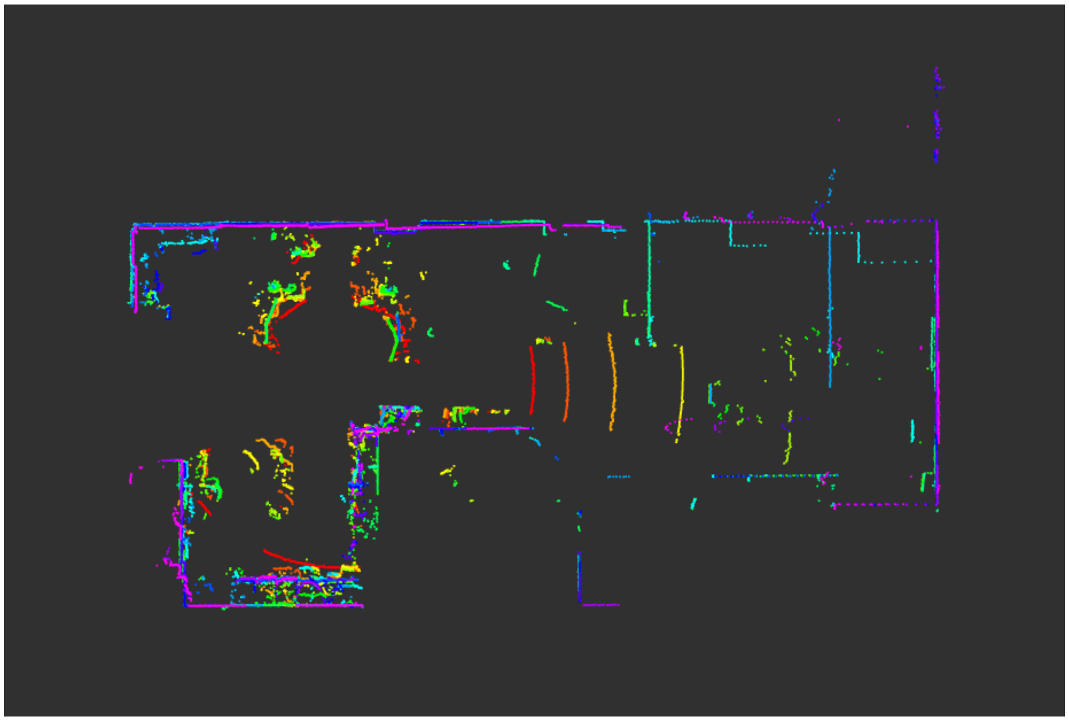


(a) Skewed sweep collected using OS-0 128.

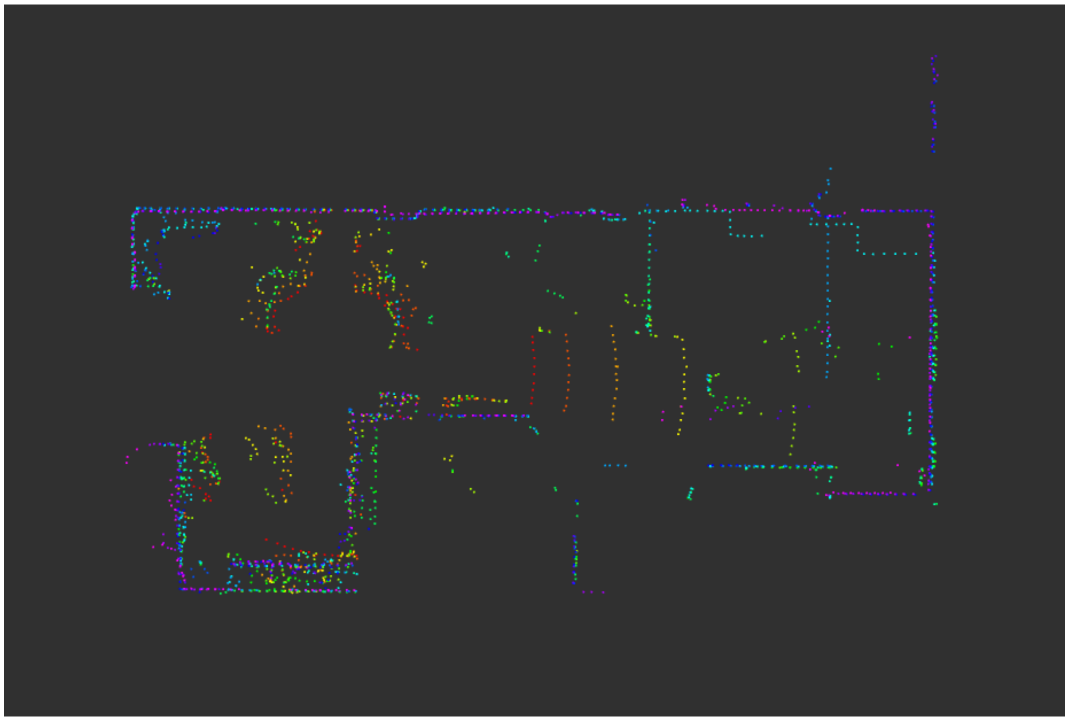


(b) Deskewed sweep collected using Velodyne Puck 16.

Figure 3.7: Scan of an office shown from top down view. Both (a) and (b) are aligned at the top right corner of the image. Top wall looks warped in Figure 3.7a.



(a) LIDAR measurements without filtering.



(b) LIDAR measurements with filtering.

Figure 3.8: LIDAR measurement of an office is filtered using Voxel Filtering method.

3.3.1.3 Point Cloud Filtering

Prior to extracting feature points from the de-skewed sweep, a filtering process is performed. Based on sensor characterization in Chapter 3 of [25], LIDAR points with a small angle of incidence or with a very large distance are filtered out. Additionally, measurements occluded by objects are also omitted because they can appear as corners as shown in Figure 3.9. These points are identified through a sudden change in the vector length between consecutive points.

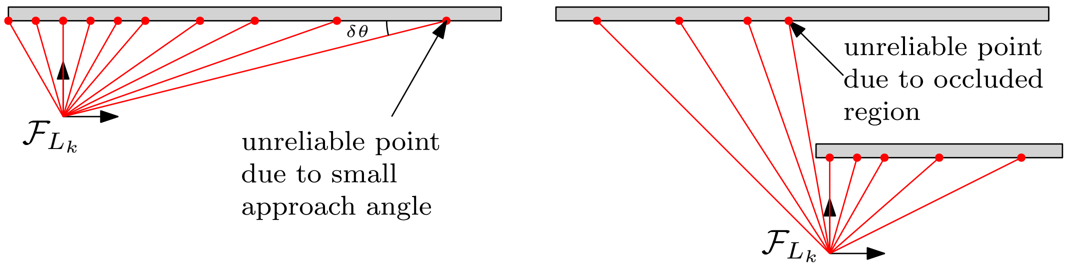

$\delta\theta$
$\mathcal{F}_{L_k}$
unreliable point due to small approach angle
unreliable point due to occluded region
$\mathcal{F}_{L_k}$

Figure 3.9: Unreliable points due to small angle of incidence and due to occlusion [25].

3.3.2 Feature Extraction

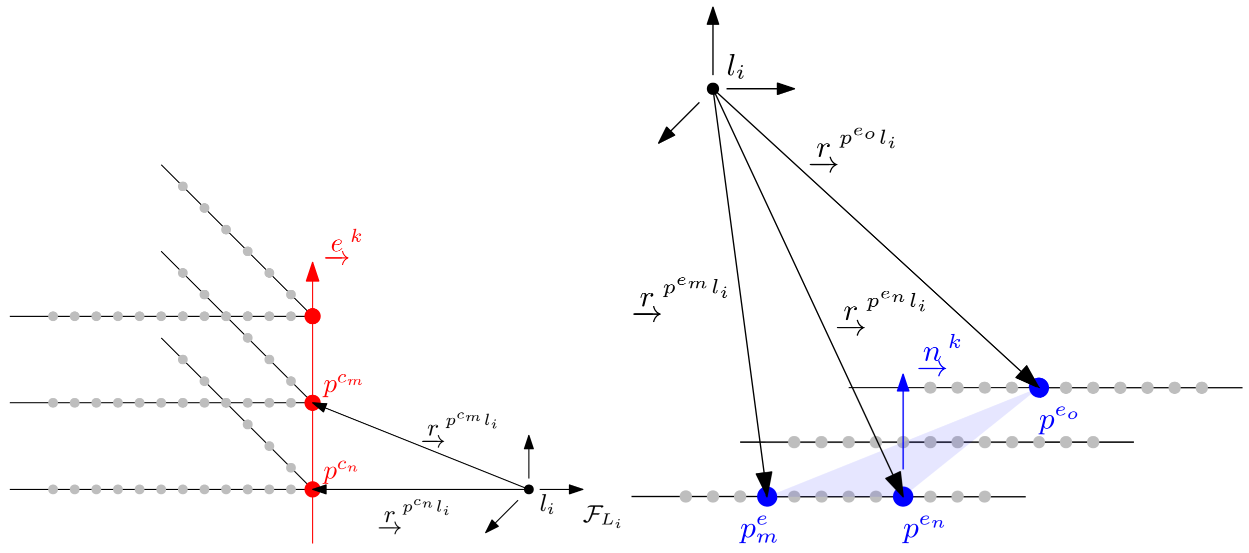
LIDAR sweep provides a complete 360 degree field of view (FoV) of the surrounding. For a 3D LIDAR, multiple laser beams are arranged along the axial direction of the sensor such that each laser beam provides a planar sweep. For example, one of Velodyne LIDARs, VLP-16, has 16 laser beams providing 16 planar scans while one of Ouster LIDAR, OS-0 128, has 128 laser beams providing 128 planar scans. Each planar sweep is called a *ring*, and these rings collectively give a complete 3D view of the surrounding. A sensor with more scan rings can output a denser point cloud.

The point cloud is dense along the scan ring, and is sparse along the axial direction. Therefore, the feature extraction is performed on each scan ring to make a use of the high resolution point cloud. 2D planar features are obtained using the approach described in [26] where the local curvature is computed based on nearby points using

$$s = \frac{1}{N\left\|\mathbf{r}_{B_k}^{p_i b_k}\right\|}\left\|\sum_{j=i-N,j\neq i}^{i+N} \mathbf{r}_{B_k}^{p_i b_k} - \mathbf{r}_{B_k}^{p_j b_k}\right\|, \tag{3.141}$$

where $\|\cdot\|$ is the Euclidean norm of a physical vector, and N is the number of nearby points used to evaluate the local curvature. Note that this is based on the assumption that the neighboring points are spaced relatively equally. In fact, the space between the points on the same ring vary based on the angle of incidence and range.

Once the curvature values are computed for each point, the points with the low curvature value are classified as surface features, and the points with the high curvature value are classified as corner features. When each ring has its own set of surface and corner points, it is then possible to extract planes and lines in 3D space. For example, a group of corner features shown in red in Figure 3.11 can be seen as edges of the room in 3D. Additionally, a cluster of surface features shown in blue can be seen as a plane at the four walls of the room. Suppose there's an corner feature p^{c_k} extracted from a sweep $\mathcal{S}_j$. And p^{c_m} and p^{c_n} are the


$\underset{\rightarrow}{e}^{k}$
p^{c_m}
p^{c_n}
$\underset{\rightarrow}{r}^{p^{c_m} l_i}$
$\underset{\rightarrow}{r}^{p^{c_n} l_i}$
l_i
$\mathcal{F}_{L_i}$
l_i
$\underset{\rightarrow}{r}^{p^{e_o} l_i}$
$\underset{\rightarrow}{r}^{p^{e_m} l_i}$
$\underset{\rightarrow}{r}^{p^{e_n} l_i}$
$\underset{\rightarrow}{n}^{k}$
p^{e_o}
p^{e}_{m}
p^{e_n}

(a) line. (b) plane.

Figure 3.10: Graphical representation of feature extraction.

two nearest corner points from p^{c_k} extracted from a sweep $\mathcal{S}_i$. A unit vector $\underset{\rightarrow}{e}^{k}$ of the edge formed by these two points is

$$\underset{\rightarrow}{e}^{k} = \frac{\underset{\rightarrow}{r}^{p^{c_m} l_i} - \underset{\rightarrow}{r}^{p^{c_n} l_i}}{\left\| \underset{\rightarrow}{r}^{p^{c_m} l_i} - \underset{\rightarrow}{r}^{p^{c_n} l_i} \right\|}. \tag{3.142}$$

Further, consider a surface feature p^{e_k} extracted from a sweep $\mathcal{S}_j$. Let p^{e_m}, p^{e_n}, and p^{e_o} be the surface features near p^{e_k} taken at $\mathcal{S}_i$ near p^{e_k}. Then, a normal vector of a plane $\underset{\rightarrow}{n}^{k}$ near the edge point p^{e_k} is formed by

$$\underset{\rightarrow}{n}^{k} = \frac{\left(\underset{\rightarrow}{r}^{p^{e_m} l_i} - \underset{\rightarrow}{r}^{p^{e_o} l_i}\right) \times \left(\underset{\rightarrow}{r}^{p^{e_n} l_i} - \underset{\rightarrow}{r}^{p^{e_o} l_i}\right)}{\left\| \left(\underset{\rightarrow}{r}^{p^{e_m} l_i} - \underset{\rightarrow}{r}^{p^{e_o} l_i}\right) \times \left(\underset{\rightarrow}{r}^{p^{e_n} l_i} - \underset{\rightarrow}{r}^{p^{e_o} l_i}\right) \right\|}. \tag{3.143}$$

This feature extraction method has been tested on a simulated environment. Figure 3.11 shows the extracted features. Blue points represent surface features, and orange points represent corner features. They collectively form planes and lines in 3D space respectively. Figure 3.12 shows the extracted features of a sweep taken by a Velodyne Puck VLP-16 on a street. White and blue points represent surface features, and red and orange points represent corner features. Once the lines and planes are deduced from the extracted corner and surface features, they can be used in point cloud registration.

3.3.3 Point Cloud Registration

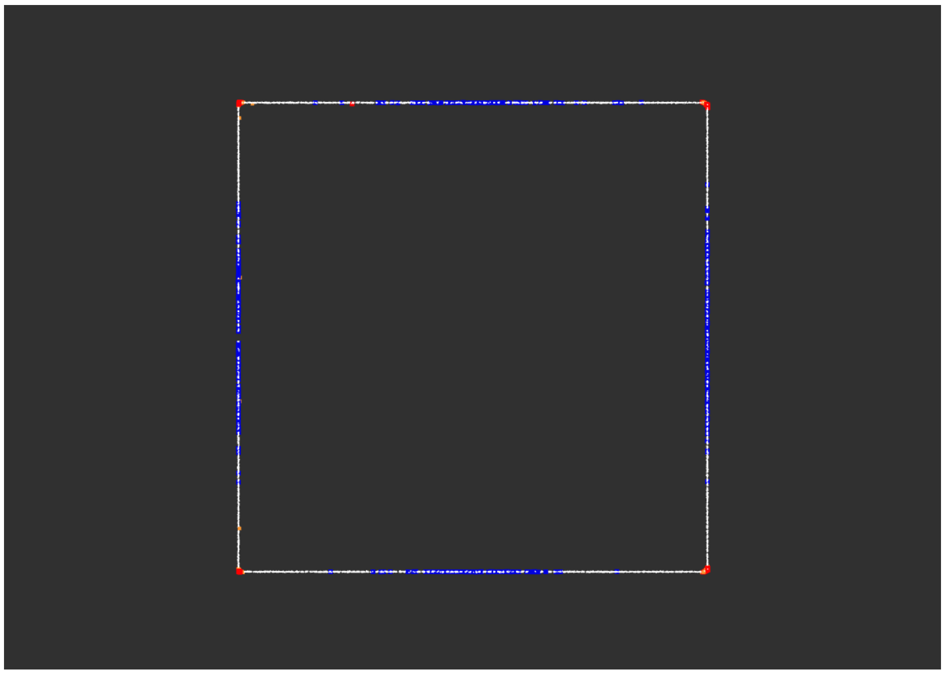
Assume a LIDAR mounted to a mobile robot accumulates a sweep at one instance in time at a particular location, and takes another sweep at another instance in time at a different location. If these two sweep measurements have enough scene overlap, it is possible to align these two point cloud. This process is called *point cloud registration.* Once the two point clouds are aligned, the transformation required to do so is equivalent to the robot's relative motion between the two instances. Point cloud registration can be done using various methods.

Iterative Closest Point (ICP), first introduced by [27], is one of the most popular point cloud registration algorithms. The algorithm steps are as follows: 1) for each point in the source point cloud, find a corresponding point in the reference point cloud, 2) weigh the corresponding pairs, 3) reject outliers, 4) minimize an error via iteration. These steps are discussed in this section.

3.3.3.1 Finding Correspondence

Unlike some camera features where the features are described by descriptors such as ORB, SURF, SIFT, etc., LIDAR features are only defined by their curvature. Further, in most cases, the points in the target sweep do not have an equivalent point in the reference sweep. Despite this, a point correspondence step is performed to find a nearest point-to-point association between two sweeps. To do so, an initial pose estimate is necessary. Thankfully, this initialization is naturally provided in mobile robotics by odometry based on inertial sensors or vision. The target point cloud is transformed using the initial pose, then the transformed target points are matched with the closest points in the reference point cloud based on Euclidean distance. To facilitate the process, k-NN search algorithm [28] is performed using k-d tree space partitioning structure.

According to [29], among the ICP sources of error which include wrong initialization, under-constrained situations, and sensor noise and biases, the initialization is proven to be the dominant error. The wrong initialization makes the algorithm converge to a local


(a) Top-down view.

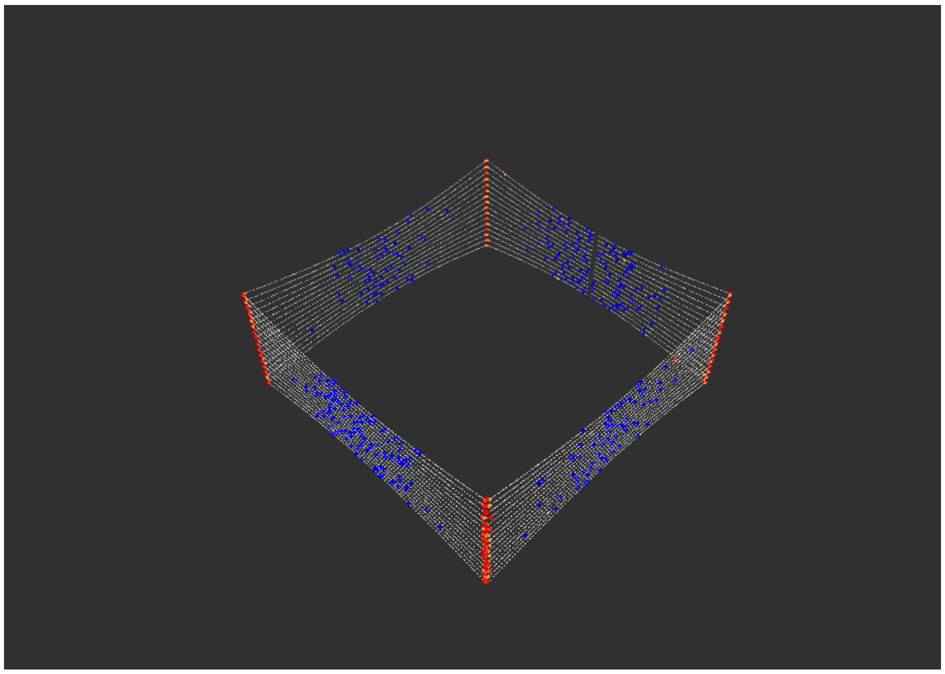
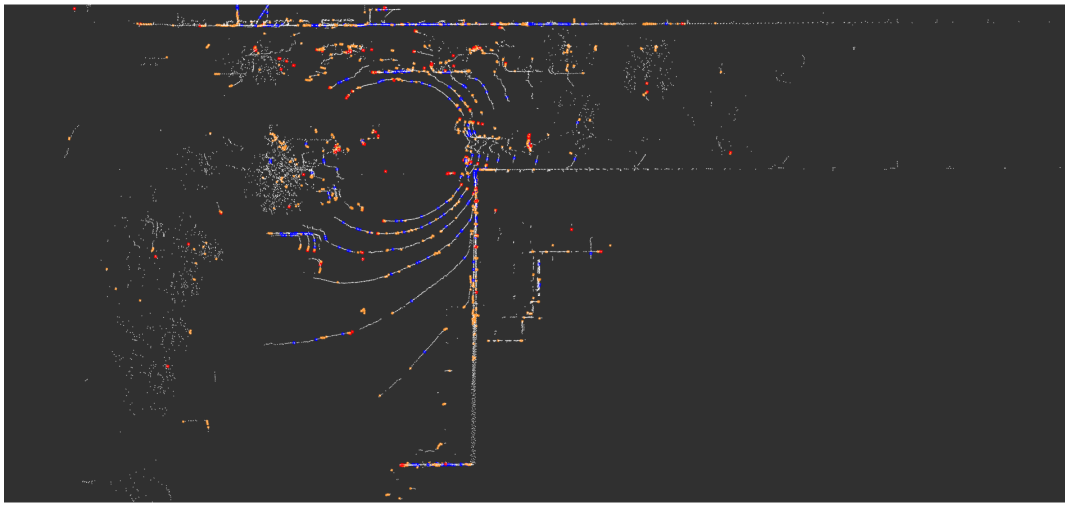

(b) 3D view.

Figure 3.11: Simulated LIDAR sweep of an empty room. Blue points are edge features and orange points are corner features.


(a) LIDAR measurement with features.

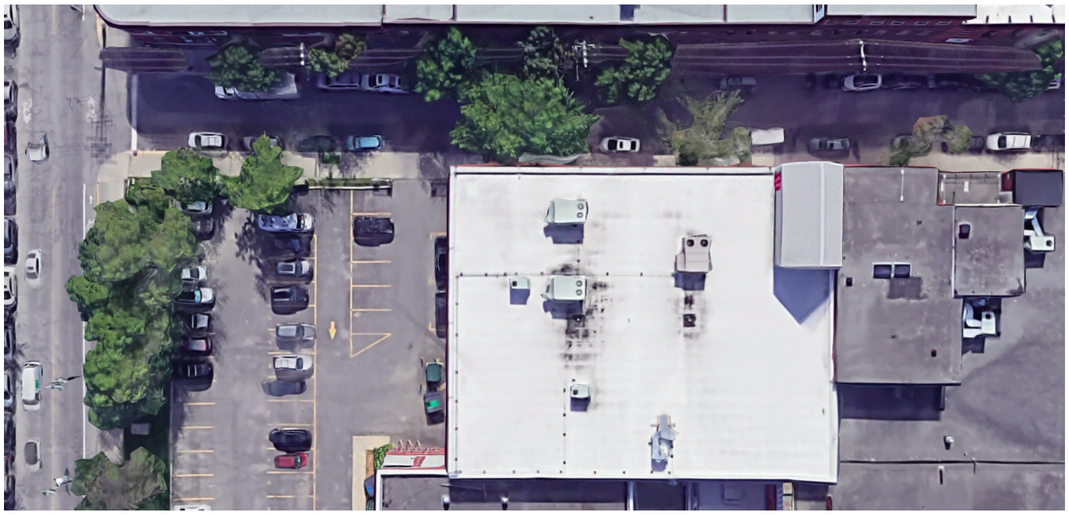

(b) Street from Google Map.

Figure 3.12: LIDAR sweep of a street collected using VLP-16. White points are surface features and blue points are flat surface features. Orange points are corner features and red points are sharp corner features. Flatness and sharpness are determined by the curvature values.

minimum out of the attraction basin of the true solution. This is due to wrong point correspondence.

3.3.3.2 Outlier Rejection

Outlier rejection of a LIDAR sweep is performed to remove unreliable points. Most of the unreliable points are filtered out using the methods in Section 3.3.1.3. One way to detect outliers is by setting a minimum threshold for the point-to-point distance. Corresponding points with point-to-point distances greater than this threshold are rejected.

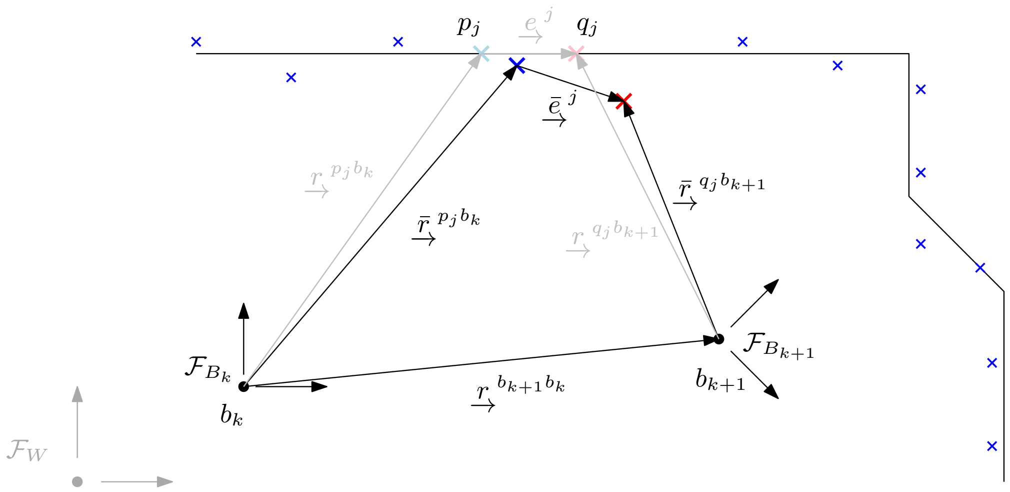
Another common method is by using RANSAC algorithm [24]. The first step of RANSAC would be to pick a random sample of consensus, namely N number of points along with their correspondences. Then, transform sampled points in the target point cloud to match the reference point cloud using the relative transformation, and compute the point-to-point residuals. This process is iterated until a relative transform whose consensus set exceeds some pre-determined threshold is found. One can choose the relative transformation which yielded the largest consensus set and use that consensus set to re-estimate the relative transform. However, usually a LIDAR feature cloud is much denser than camera features. Thus iterating through the LIDAR point cloud would be very computationally heavy.

Outlier rejection is already performed during the pre-processing step by filtering the unreliable points by occlusion and the angle of incidence. Therefore, in LIVO, no other outlier rejection method is applied. However, all the aforementioned outlier rejection methods can be applied to improve robustness of the odometry solution.

3.3.3.3 LIDAR Point Cloud Registration Factor

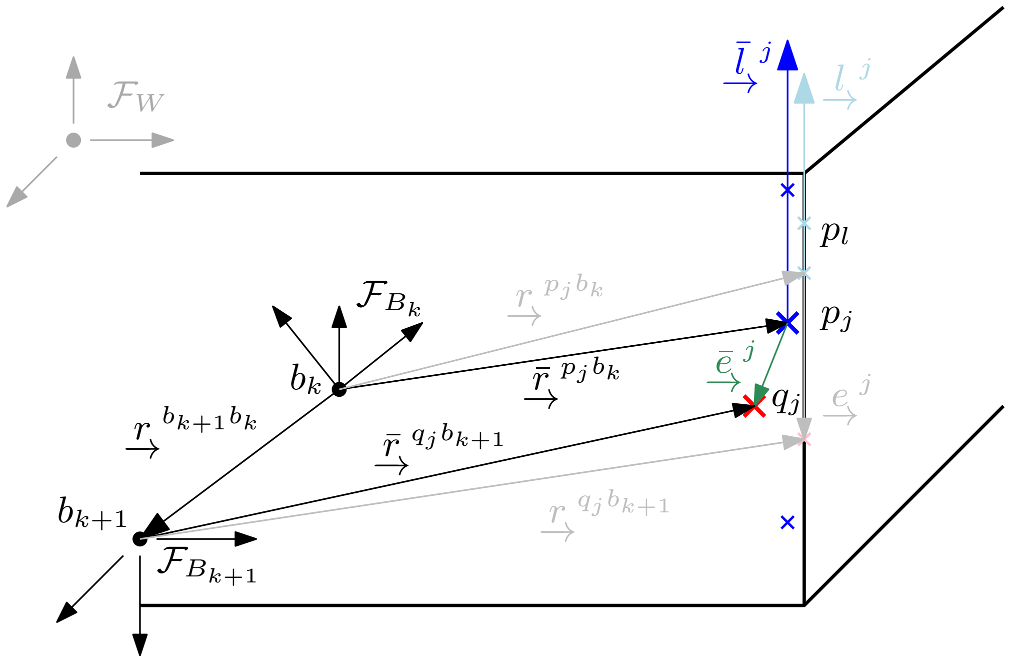
This section presents a derivation of the point-to-line and point-to-plane error metrics for an ICP algorithm. The conventional point-to-plane error metric proposed by [30] involves small angle approximations. In a recent publication [31], the solution is obtained via the weighted optimal linear attitude and translation estimator (WOLATE) using the Cayley transform. Here, the derivation is an extension of the work in Section 5.3 of [25].

Consider Figure 3.14. Let q_j be a point in a target point cloud, and p_j be the corresponding point in the reference point cloud. Recall that the corresponding point is *not the same point* in physical space, yet the closest point. This means that the distance between the two corresponding points $\underset{\rightarrow}{e}^{j}$ is not zero unless the points are identical. Note that these points



p_j
$\underrightarrow{e}^{j}$
q_j
$\underrightarrow{\bar{e}}^{j}$
$\underrightarrow{r}^{p_j b_k}$
$\underrightarrow{\bar{r}}^{p_j b_k}$
$\underrightarrow{\bar{r}}^{q_j b_{k+1}}$
$\underrightarrow{r}^{q_j b_{k+1}}$
$\mathcal{F}_{B_{k+1}}$
$\mathcal{F}_{B_k}$
b_k
$\underrightarrow{r}^{b_{k+1} b_k}$
b_{k+1}
$\mathcal{F}_W$

Figure 3.13: Geometrical representation of point to point registration problem. Blue points are measured relative to point b_k, and the red point is measured relative to point b_{k+1} [25].



$\underrightarrow{\bar{l}}^{j}$
$\underrightarrow{l}^{j}$
$\mathcal{F}_W$
p_l
$\mathcal{F}_{B_k}$
$\underrightarrow{r}^{p_j b_k}$
p_j
b_k
$\underrightarrow{\bar{r}}^{p_j b_k}$
$\underrightarrow{\bar{e}}^{j}$
q_j
$\underrightarrow{e}^{j}$
$\underrightarrow{r}^{b_{k+1} b_k}$
$\underrightarrow{\bar{r}}^{q_j b_{k+1}}$
b_{k+1}
$\underrightarrow{r}^{q_j b_{k+1}}$
$\mathcal{F}_{B_{k+1}}$

Figure 3.14: Geometrical representation of point to line registration problem. Blue points are measured relative to point b_k, and the red point is measured relative to point b_{k+1} [25].

are always subject to noise thus,

$$\tilde{\mathbf{r}}_{B_k}^{p_j b_k} = \mathbf{r}_{B_k}^{p_j b_k} + \boldsymbol{\eta}_{B_k}^{p_j}, \tag{3.144}$$

$$\tilde{\mathbf{r}}_{B_{k+1}}^{q_j b_{k+1}} = \mathbf{r}_{B_{k+1}}^{q_j b_{k+1}} + \boldsymbol{\eta}_{B_{k+1}}^{q_j}, \tag{3.145}$$

where $\boldsymbol{\eta}_{B_k}^{p_j} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}^{p_j}\right)$ and $\boldsymbol{\eta}_{B_{k+1}}^{q_j} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}^{q_j}\right)$ are the are the noises associated with the measurement of point p_j and q_j, respectively.. Then, the distance between the two points is

$$\underset{\rightarrow}{\bar{e}}^{\,j} = \underset{\rightarrow}{\bar{r}}^{\,q_j b_{k+1}} + \underset{\rightarrow}{r}^{\,b_{k+1} b_k} - \underset{\rightarrow}{\bar{r}}^{\,p_j b_k}. \tag{3.146}$$

Resolving each physical vector in their respective frames yields

$$\bar{\mathbf{e}}_{B_k}^{q_j p_j} = \mathbf{C}_{B_k B_{k+1}} \bar{\mathbf{r}}_{B_{k+1}}^{q_j b_{k+1}} + \mathbf{r}_{B_k}^{b_{k+1} b_k} - \bar{\mathbf{r}}_{B_k}^{p_j b_k}. \tag{3.147}$$

The relative position $\mathbf{r}_{B_k}^{b_{k+1} b_k}$ and rotation $\mathbf{C}_{B_k B_{k+1}}$ can be found by solving a least squares problem. If the cost function is defined to minimize the distance between the two points, $\underset{\rightarrow}{\bar{e}}^{\,j}$, this error metric is called *point-to-point.* Instead of minimizing the euclidean distance between the measured points p_j and q_j, if the distance between a point q_j in a target point cloud and the local line formed by a point p_j and some neighboring corner feature p_l is minimized, this error metric is called *point-to-line.* A line vector is extracted using corner points in Section 3.3.2. The closest distance between a point and a line is the difference between the distance between the two points and its projection on top of the line vector

$$d = \left\| \underset{\rightarrow}{\bar{e}}^{\,j} - \frac{\underset{\rightarrow}{\bar{l}}}{\left\| \underset{\rightarrow}{\bar{l}} \right\|} \underset{\rightarrow}{\bar{l}}^{\,j} \cdot \underset{\rightarrow}{\bar{e}}^{\,j} \right\|. \tag{3.148}$$

Given that the line vector is normalized, (3.148) can be simplified as

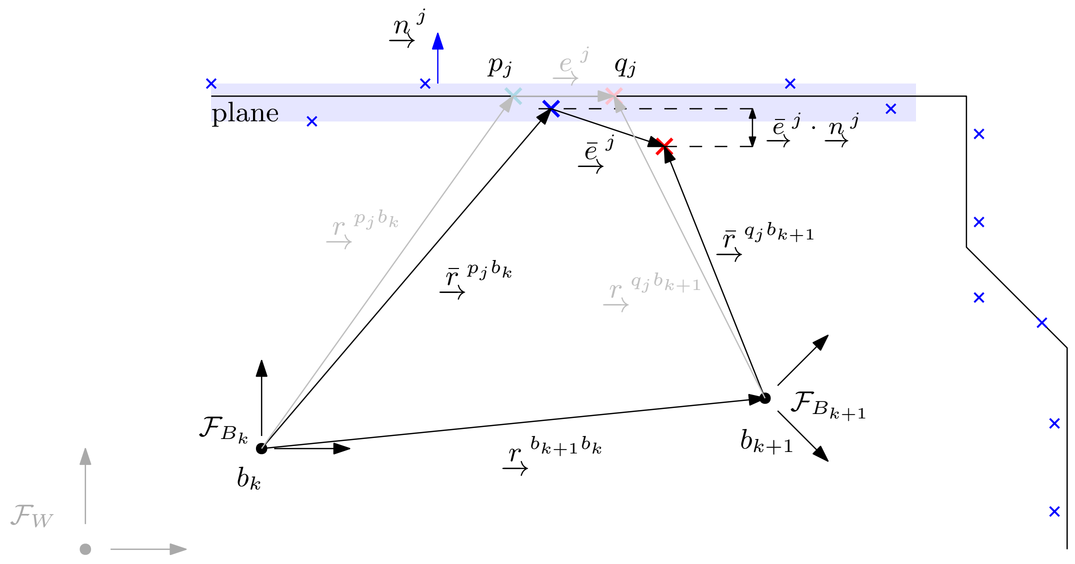
$$d = \left\| \underset{\rightarrow}{\bar{e}}^{\,j} - \underset{\rightarrow}{\bar{l}}^{\,j} \underset{\rightarrow}{\bar{l}}^{\,j} \cdot \underset{\rightarrow}{\bar{e}}^{\,j} \right\|. \tag{3.149}$$

Thus, by resolving the physical vectors in the their respective frames, the point-to-line residual $\mathbf{e}_l$ can be written as

$$\mathbf{e}_l^j = \bar{\mathbf{e}}_{B_k}^{q_j p_j} - \bar{\mathbf{l}}_{B_k}^j \bar{\mathbf{l}}_{B_k}^{j\,\mathsf{T}} \bar{\mathbf{e}}_{B_k}^{q_j p_j}, \tag{3.150}$$

where $\bar{\mathbf{e}}_{B_k}^{q_j p_j}$ is previously defined in (3.147).

Similarly, the *point-to-plane* error metric minimizes the distance between a point q_j and


$\underset{\rightarrow}{n}^{j}$
p_j
$\underset{\rightarrow}{e}^{j}$
q_j
plane
$\underset{\rightarrow}{\bar{e}}^{j} \cdot \underset{\rightarrow}{n}^{j}$
$\underset{\rightarrow}{\bar{e}}^{j}$
$\underset{\rightarrow}{r}^{p_j b_k}$
$\underset{\rightarrow}{\bar{r}}^{q_j b_{k+1}}$
$\underset{\rightarrow}{\bar{r}}^{p_j b_k}$
$\underset{\rightarrow}{r}^{q_j b_{k+1}}$
$\mathcal{F}_{B_k}$
b_k
$\mathcal{F}_{B_{k+1}}$
b_{k+1}
$\underset{\rightarrow}{r}^{b_{k+1} b_k}$
$\mathcal{F}_W$
Figure 3.15: Geometrical representation of point to plane registration problem. Blue points are measured relative to point b_k, and the red point is measured relative to point b_{k+1} [25].

the local surface found around the corresponding point p_j in the reference frame. The local surface is found using the method described in Section 3.3.2. The closest distance between a point and a plane is the projection of $\underset{\rightarrow}{\bar{e}}^{j}$ on to the normal vector of the plane $\underset{\rightarrow}{n}^{j}$ given as

$$d = \underset{\rightarrow}{n}^{j} \cdot \underset{\rightarrow}{\bar{e}}^{j}. \tag{3.151}$$

Thus, the point-to-plane residual is given as

$$\begin{aligned} e_{pl}^{j} &= \mathbf{n}_{B_k}^{j\,\mathsf{T}} \left(\mathbf{C}_{B_k B_{k+1}} \bar{\mathbf{r}}_{B_{k+1}}^{q_j b_{k+1}} + \mathbf{r}_{B_k}^{b_{k+1} b_k} - \bar{\mathbf{r}}_{B_k}^{p_j b_k} \right), &(3.152)\\ &= \mathbf{n}_{B_k}^{j\,\mathsf{T}} \left(\mathbf{C}_{W B_k}^{\mathsf{T}} \left(\mathbf{C}_{W B_{k+1}} \bar{\mathbf{r}}_{B_{k+1}}^{q_j b_{k+1}} + \mathbf{r}_{W}^{b_{k+1} w} - \mathbf{r}_{W}^{b_k w} \right) - \bar{\mathbf{r}}_{B_k}^{p_j b_k} \right). &(3.153) \end{aligned}$$

Lastly, the LIDAR point cloud registration factor can be made with the combination of these errors. Using point-to-line and point-to-plane error metrics together yields

$$\mathbf{e}_{\mathcal{L}} = \begin{bmatrix} \mathbf{e}_{pl}^{\mathsf{T}} & \mathbf{e}_{l}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}, \tag{3.154}$$

where

$$\mathbf{e}_{pl} = \begin{bmatrix} e_{pl}^{1} \\ e_{pl}^{2} \\ \vdots \\ e_{pl}^{N_{pl}} \end{bmatrix}, \qquad \mathbf{e}_{l} = \begin{bmatrix} \mathbf{e}_{l}^{1} \\ \mathbf{e}_{l}^{2} \\ \vdots \\ \mathbf{e}_{l}^{N_{l}} \end{bmatrix}. \tag{3.155}$$

The LIDAR point cloud registration factor constrains two robot poses using the point cloud registration error metric. In practice, a local map is used as the reference point cloud. A local map is built using multiple sweeps resolved in some common frame, a frame where a reliable robot pose is resolved in, to create a denser point cloud to match against. The usage of this factor in the back-end of SLAM is discussed in Chapter 4.

Chapter 4

LIVOM - Optimization and Results

The back-end of SLAM solves the constrained optimization problem established in the front-end. The first attempt at solving the SLAM back-end is using a nonlinear variant of the linear Kalman Filter (KF), the extended Kalman Filter (EKF), which makes use of a prediction-correction scheme based on a Markov process [32]. The filter-based methods perform well in real-time applications due to their simplicity and low computation requirement. However, Markov-based odometry solutions cannot update past estimates given new measurements, and this can cause problems in the mapping processes. Batch methods, also known as graph-based methods, can efficiently solve this problem by keeping a window of the robot's old poses. This is made possible thanks to recent advancements in hardware able to execute expensive computations and in the field of linear solvers. Graph-based unconstrained optimization problems are formulated as least squares problems, and can be solved using various optimization methods such as Gauss-Newton (GN) or Levenberg-Marquardt (LM). The Ceres solver [33] is used to solve the back-end least squares problem associated with SLAM. This chapter discusses novel tightly-coupled LIDAR-Inertial-Visual Odometry and Mapping by solving the factor graph established in Chapter 3. The proposed navigation solution comprises of odometry initialization in Section 4.1, the sliding window filter problem setup in Section 4.2, marginalization in Section 4.3, and finally experimental results and comparison to the state-of-the art solutions in Section 4.4.

4.1 Odometry Initialization

The initial robot pose estimation is essential as the successive trajectory estimate is relative to the robot's initial state. Sometimes, a poor initialization can affect the data association in the front-end. For example, LIDAR sweeps are *deskewed* using inertial sensor measurements and the robot kinematics. If the initial robot pose is not properly aligned with the gravity, the sweep may not be de-skewed thoroughly, and can cause drift in the odometry solution. For a case like monocular visual SLAM, the scale is not directly observable. The scale factor relates the physical dimensions of the objects to the corresponding dimensions in a camera image [34]. Wrong scale estimate hinders directly fusing the relative vision factor with other sensor data in monocular vision SLAM.

A loosely-coupled sensor fusion method is adopted by [1] to realize estimator initialization. In traditional vision-only SLAM, the Structure from Motion (SfM) method, a photogrammetric range imaging technique, is used to estimate the initial camera poses [1]. Relative motion can be estimated using eight-point [23] or five-point [23] algorithms. By aligning the relative camera poses from the SfM with the preintegrated IMU poses, the IMU sensor biases, velocities, and the camera scale factor can be recovered. Similarly, a LIDAR scan-to-scan matching is performed to provide LIDAR poses [35], and with sufficient motion of the sensor pair, the initial attitude of the robot with respect to the gravity vector as well as the IMU sensor biases can be obtained [2].

In this section, the initialization scheme proposed by [2] is discussed because LIDAR scan matching can provide more accurate relative poses, and the monocular camera's scale factor can be directly computed. The system initialization is done in four steps that include rate gyroscope bias estimation, gravity vector alignment, velocity estimation with gravity refinement, and accelerometer bias estimation.

4.1.1 Rate Gyroscope Bias Estimation

Consider two body poses at $t = t_k$ and $t = t_{k+1}$. Recall that the rotation relative motion increment has the form

$$\Delta \mathbf{C}_{kk+1} \triangleq \mathbf{C}_{WB_k}^{\mathsf{T}} \mathbf{C}_{WB_{k+1}}, \tag{4.1}$$

and its first order update is

$$\hat{\Delta \mathbf{C}}_{kk+1} \approx \bar{\Delta \mathbf{C}}_{kk+1} \mathrm{Exp}\left(\frac{\partial \Delta \mathbf{C}_{kk+1}}{\partial \boldsymbol{\beta}_B^g} \delta \boldsymbol{\beta}_B^g \right). \tag{4.2}$$

A window of the robot poses is used to initialize the system. Assuming constant bias across the window, the gyroscope bias can be estimated by minimizing the difference between the relative robot pose and the corresponding preintegrated IMU angular velocity measurements

(3.19),

$$\delta\boldsymbol{\beta}^g_B = \underset{\delta\boldsymbol{\beta}^g_B}{\arg\min} \sum_{k\in\mathcal{W}} J_k\left(\mathbf{C}_{WB_k}, \mathbf{C}_{WB_{k+1}}\right), \tag{4.3}$$

where $\mathcal{W}$ indexes the robot poses in the window, and

$$J_k\left(\mathbf{C}_{WB_k}, \mathbf{C}_{WB_{k+1}}\right) = \tfrac{1}{2}\left\|\operatorname{Log}\left(\Delta\hat{\mathbf{C}}^{\mathsf{T}}_{kk+1}\mathbf{C}^{\mathsf{T}}_{WB_k}\mathbf{C}_{WB_{k+1}}\right)\right\|^2. \tag{4.4}$$

The optimization problem can be formulated as a nonlinear least squares problem. The error residual between the rotation relative motion increment and the relative rotation from scan matching can be linearized as

$$\begin{aligned}
\mathbf{e}_k\left(\boldsymbol{\beta}^g_B\right) &= \mathbf{e}_k\left(\bar{\boldsymbol{\beta}}^g_B + \delta\boldsymbol{\beta}^g_B\right) &(4.5)\\
&= \operatorname{Log}\left(\left(\Delta\bar{\mathbf{C}}_{kk+1}\operatorname{Exp}\left(\left.\frac{\partial\Delta\mathbf{C}_{kk+1}}{\partial\boldsymbol{\beta}^g_B}\right|_{\bar{\boldsymbol{\beta}}^g_B}\delta\boldsymbol{\beta}^g_B\right)\right)^{\mathsf{T}}\mathbf{C}^{\mathsf{T}}_{WB_k}\mathbf{C}_{WB_{k+1}}\right) &(4.6)\\
&= \operatorname{Log}\left(\operatorname{Exp}\left(\left.\frac{\partial\Delta\mathbf{C}_{kk+1}}{\partial\boldsymbol{\beta}^g_B}\right|_{\bar{\boldsymbol{\beta}}^g_B}\delta\boldsymbol{\beta}^g_B\right)^{\mathsf{T}}\Delta\bar{\mathbf{C}}^{\mathsf{T}}_{kk+1}\mathbf{C}^{\mathsf{T}}_{WB_k}\mathbf{C}_{WB_{k+1}}\right), &(4.7)
\end{aligned}$$

where $\Delta\bar{\mathbf{C}}_{kk+1} = \Delta\mathbf{C}_{kk+1}\left(\bar{\boldsymbol{\beta}}^g_B\right)$, and $\bar{\boldsymbol{\beta}}^g_B = 0$. Using the first order approximation (2.36), the property of the exponential map give in (2.30), and by setting

$$\Delta\bar{\mathbf{C}}^{\mathsf{T}}_{kk+1}\mathbf{C}^{\mathsf{T}}_{WB_k}\mathbf{C}_{WB_{k+1}} = \delta\Delta\tilde{\mathbf{C}}_{kk+1}, \tag{4.8}$$

the above equation can be rewritten as

$$\begin{aligned}
\mathbf{e}_k\left(\bar{\boldsymbol{\beta}}^g_B + \delta\boldsymbol{\beta}^g_B\right) &= \operatorname{Log}\left(\operatorname{Exp}\left(-\left.\frac{\partial\Delta\mathbf{C}_{kk+1}}{\partial\boldsymbol{\beta}^g_B}\right|_{\bar{\boldsymbol{\beta}}^g_B}\delta\boldsymbol{\beta}^g_B\right)\delta\Delta\tilde{\mathbf{C}}_{kk+1}\right). &(4.9)\\
&= \operatorname{Log}\left(\delta\Delta\tilde{\mathbf{C}}_{kk+1}\operatorname{Exp}\left(-\delta\Delta\tilde{\mathbf{C}}^{\mathsf{T}}_{kk+1}\left.\frac{\partial\Delta\mathbf{C}_{kk+1}}{\partial\boldsymbol{\beta}^g_B}\right|_{\bar{\boldsymbol{\beta}}^g_B}\delta\boldsymbol{\beta}^g_B\right)\right) &(4.10)\\
&\approx \operatorname{Log}\left(\delta\Delta\tilde{\mathbf{C}}_{kk+1}\right) - \mathbf{J}_r^{-1}\left(\operatorname{Log}\left(\delta\Delta\tilde{\mathbf{C}}_{kk+1}\right)\right)\delta\Delta\tilde{\mathbf{C}}^{\mathsf{T}}_{kk+1}\left.\frac{\partial\Delta\mathbf{C}_{kk+1}}{\partial\boldsymbol{\beta}^g_B}\right|_{\bar{\boldsymbol{\beta}}^g_B}\delta\boldsymbol{\beta}^g_B &(4.11)\\
&= \bar{\mathbf{e}}_k + \left.\frac{\partial\mathbf{e}_k}{\partial\boldsymbol{\beta}^g_B}\right|_{\bar{\boldsymbol{\beta}}^g_B}\delta\boldsymbol{\beta}^g_B. &(4.12)
\end{aligned}$$

Having the linearized error residual, the objective function can be perturbed as

$$\begin{aligned}
J_k\left(\boldsymbol{\beta}_B^g\right) &= \tfrac{1}{2}\mathbf{e}_k\left(\boldsymbol{\beta}_B^g\right)^{\mathsf{T}}\mathbf{e}_k\left(\boldsymbol{\beta}_B^g\right) && (4.13)\\
&= \tfrac{1}{2}\mathbf{e}_k\left(\bar{\boldsymbol{\beta}}_B^g+\delta\boldsymbol{\beta}_B^g\right)^{\mathsf{T}}\mathbf{e}_k\left(\bar{\boldsymbol{\beta}}_B^g+\delta\boldsymbol{\beta}_B^g\right) && (4.14)\\
&\approx \tfrac{1}{2}\left(\mathbf{e}_k+\left.\frac{\partial \mathbf{e}_k}{\partial \boldsymbol{\beta}_B^g}\right|_{\bar{\boldsymbol{\beta}}_B^g}\delta\boldsymbol{\beta}_B^g\right)^{\mathsf{T}}\left(\mathbf{e}_k+\left.\frac{\partial \mathbf{e}_k}{\partial \boldsymbol{\beta}_B^g}\right|_{\bar{\boldsymbol{\beta}}_B^g}\delta\boldsymbol{\beta}_B^g\right) && (4.15)\\
&= \tfrac{1}{2}\bar{\mathbf{e}}_k^{\mathsf{T}}\bar{\mathbf{e}}_k+\left(\left.\frac{\partial \mathbf{e}_k}{\partial \boldsymbol{\beta}_B^g}\right|_{\bar{\boldsymbol{\beta}}_B^g}\right)^{\mathsf{T}}\bar{\mathbf{e}}_k\delta\boldsymbol{\beta}_B^g+\tfrac{1}{2}\delta\boldsymbol{\beta}_B^{g\,\mathsf{T}}\left(\left.\frac{\partial \mathbf{e}_k}{\partial \boldsymbol{\beta}_B^g}\right|_{\bar{\boldsymbol{\beta}}_B^g}\right)^{\mathsf{T}}\left(\left.\frac{\partial \mathbf{e}_k}{\partial \boldsymbol{\beta}_B^g}\right|_{\bar{\boldsymbol{\beta}}_B^g}\right)\delta\boldsymbol{\beta}_B^g, && (4.16)
\end{aligned}$$

so the Jacobian and the Hessian matrices of each objective function are

$$\begin{aligned}
\left.\frac{\partial J_k}{\partial \boldsymbol{\beta}_B^g}\right|_{\delta\bar{\boldsymbol{\beta}}_B^g} &= \left(\left.\frac{\partial \mathbf{e}_k}{\partial \boldsymbol{\beta}_B^g}\right|_{\bar{\boldsymbol{\beta}}_B^g}\right)^{\mathsf{T}}\bar{\mathbf{e}}_k, && (4.17)\\
\left.\frac{\partial^2 J_k}{\partial {\boldsymbol{\beta}_B^g}^2}\right|_{\delta\bar{\boldsymbol{\beta}}_B^g} &= \left(\left.\frac{\partial \mathbf{e}_k}{\partial \boldsymbol{\beta}_B^g}\right|_{\bar{\boldsymbol{\beta}}_B^g}\right)^{\mathsf{T}}\left(\left.\frac{\partial \mathbf{e}_k}{\partial \boldsymbol{\beta}_B^g}\right|_{\bar{\boldsymbol{\beta}}_B^g}\right). && (4.18)
\end{aligned}$$

The complete Jacobian and Hessian matrices are simply the summation of all the Jacobian and Hessian in the window,

$$\begin{aligned}
\left.\frac{\partial J}{\partial \boldsymbol{\beta}_B^g}\right|_{\delta\bar{\boldsymbol{\beta}}_B^g} &= \sum_{k\in\mathcal{W}}\left.\frac{\partial J_k}{\partial \boldsymbol{\beta}_B^g}\right|_{\bar{\boldsymbol{\beta}}_B^g}, && (4.19)\\
\left.\frac{\partial^2 J}{\partial {\boldsymbol{\beta}_B^g}^2}\right|_{\delta\bar{\boldsymbol{\beta}}_B^g} &= \sum_{k\in\mathcal{W}}\left.\frac{\partial^2 J_k}{\partial {\boldsymbol{\beta}_B^g}^2}\right|_{\bar{\boldsymbol{\beta}}_B^g}. && (4.20)
\end{aligned}$$

Finally, the gyroscope bias can be estimated using the Gauss-Newton method,

$$\begin{aligned}
\delta\boldsymbol{\beta}_B^g &= -\left(\left.\frac{\partial^2 J}{\partial {\boldsymbol{\beta}_B^g}^2}\right|_{\delta\bar{\boldsymbol{\beta}}_B^g}\right)^{-1}\left.\frac{\partial J}{\partial \boldsymbol{\beta}_B^g}\right|_{\delta\bar{\boldsymbol{\beta}}_B^g}, && (4.21)\\
\hat{\boldsymbol{\beta}}_B^g &= \bar{\boldsymbol{\beta}}_B^g+\delta\boldsymbol{\beta}_B^g. && (4.22)
\end{aligned}$$

4.1.2 Gravity Vector Initialization

The pose estimates of the initial window obtained by the LIDAR scan matching assumes that the gravity in the initial body frame is $\mathbf{g}_{B_0}=\begin{bmatrix}0 & 0 & -9.81\end{bmatrix}^{\mathsf{T}}$. However, the initial orientation of the robot is not always aligned with the Earth's gravity vector. This section shows how the gravity can be estimated using the inertial sensor measurements and the estimated poses. Consider the preintegrated IMU measurements between three consecutive sweeps, namely sweep at $t = t_k$, $t = t_{k+1}$, and $t = t_{k+2}$, or simply 1, 2, 3 for short,

$$\mathbf{r}_{B_0}^{b_2 b_0} = \mathbf{r}_{B_0}^{b_1 b_0} + \mathbf{v}_{B_0}^{b_1 b_0/B_0} \Delta t_{12} + \tfrac{1}{2} \mathbf{g}_{B_0} \Delta t_{12}^2 + \mathbf{C}_{B_0 B_1} \Delta \mathbf{r}_{12}, \tag{4.23}$$

$$\mathbf{r}_{B_0}^{b_3 b_0} = \mathbf{r}_{B_0}^{b_2 b_0} + \mathbf{v}_{B_0}^{b_2 b_0/B_0} \Delta t_{23} + \tfrac{1}{2} \mathbf{g}_{B_0} \Delta t_{23}^2 + \mathbf{C}_{B_0 B_2} \Delta \mathbf{r}_{23}, \tag{4.24}$$

$$\mathbf{v}_{B_0}^{b_2 b_0/B_0} = \mathbf{v}_{B_0}^{b_1 b_0/B_0} + \mathbf{g}_{B_0} \Delta t_{12} + \mathbf{C}_{B_0 B_1} \Delta \mathbf{v}_{12}. \tag{4.25}$$

Because the scan matching does not provide any velocity estimates, the velocities are eliminated using (4.1), (4.2), and (4.3) via a parametrization. First, substituting (4.25) into (4.24) yields

$$\mathbf{r}_{B_0}^{b_3 b_0} = \mathbf{r}_{B_0}^{b_2 b_0} + \left(\mathbf{v}_{B_0}^{b_1 b_0/B_0} + \mathbf{g}_{B_0} \Delta t_{12} + \mathbf{C}_{B_0 B_1} \Delta \mathbf{v}_{12} \right) \Delta t_{23} + \tfrac{1}{2} \mathbf{g}_{B_0} \Delta t_{23}^2 + \mathbf{C}_{B_0 B_2} \Delta \mathbf{r}_{23} \tag{4.26}$$

$$= \mathbf{r}_{B_0}^{b_2 b_0} + \mathbf{v}_{B_0}^{b_1 b_0/B_0} \Delta t_{23} + \mathbf{g}_{B_0} \Delta t_{12} \Delta t_{23} + \mathbf{C}_{B_0 B_1} \Delta \mathbf{v}_{12} \Delta t_{23} + \tfrac{1}{2} \mathbf{g}_{B_0} \Delta t_{23}^2 + \mathbf{C}_{B_0 B_2} \Delta \mathbf{r}_{23}. \tag{4.27}$$

By pre-multiplying (4.23) and (4.27) by Δt_{23} and Δt_{12}, respectively, results

$$\mathbf{r}_{B_0}^{b_2 b_0} \Delta t_{23} = \mathbf{r}_{B_0}^{b_1 b_0} \Delta t_{23} + \mathbf{v}_{B_0}^{b_1 b_0/B_0} \Delta t_{12} \Delta t_{23} + \tfrac{1}{2} \mathbf{g}_{B_0} \Delta t_{12}^2 \Delta t_{23} + \mathbf{C}_{B_0 B_1} \Delta \mathbf{r}_{12} \Delta t_{23}, \tag{4.28}$$

$$\mathbf{r}_{B_0}^{b_3 b_0} \Delta t_{12} = \mathbf{r}_{B_0}^{b_2 b_0} \Delta t_{12} + \mathbf{v}_{B_0}^{b_1 b_0/B_0} \Delta t_{23} \Delta t_{12} + \mathbf{g}_{B_0} \Delta t_{12}^2 \Delta t_{23} + \tag{4.29}$$

$$\mathbf{C}_{B_0 B_1} \Delta \mathbf{v}_{12} \Delta t_{23} \Delta t_{12} + \tfrac{1}{2} \mathbf{g}_{B_0} \Delta t_{23}^2 \Delta t_{12} + \mathbf{C}_{B_0 B_2} \Delta \mathbf{r}_{23} \Delta t_{12}. \tag{4.30}$$

Subtracting and isolating the gravity terms on the left hand side yields

$$a_{123} \mathbf{g}_{B_0} = \mathbf{b}_{123}, \tag{4.31}$$

where

$$a_{123} = \tfrac{1}{2} \left(\Delta t_{12}^2 \Delta t_{23} + \Delta t_{12} \Delta t_{23}^2 \right) \mathbf{g}_{B_0}, \tag{4.32}$$

and

$$\mathbf{b}_{123} = \left(\mathbf{r}_{B_0}^{b_1 b_0} - \mathbf{r}_{B_0}^{b_2 b_0} \right) \Delta t_{23} - \left(\mathbf{r}_{B_0}^{b_2 b_0} - \mathbf{r}_{B_0}^{b_3 b_0} \right) \Delta t_{12} + \tag{4.33}$$

$$\mathbf{C}_{B_0 B_1} \Delta \mathbf{r}_{12} \Delta t_{23} - \mathbf{C}_{B_0 B_2} \Delta \mathbf{r}_{23} \Delta t_{12} - \mathbf{C}_{B_0 B_1} \Delta \mathbf{v}_{12} \Delta t_{12} \Delta t_{23}. \tag{4.34}$$

Thus, the gravity term can be computed by the system of the linear expressions obtained via the remaining consecutive sweep triplets in the window such that

$$\mathbf{A} \mathbf{g}_{B_0} = \mathbf{B}, \tag{4.35}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{123} \\ a_{234} \\ \vdots \\ a_{k-2k-1k} \end{bmatrix}, \tag{4.36}$$

and

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_{123} \\ \mathbf{b}_{234} \\ \vdots \\ \mathbf{b}_{k-2k-1k} \end{bmatrix}, \tag{4.37}$$

for $k \in \mathcal{W}$.

4.1.3 Velocity Estimation and Gravity Refinement

The estimated gravity is further refined in this section by solving a nonlinear least squares problem. Consider the equations (4.23) and (4.25). Note that the gravity vector is constrained with its magnitude $g = 9.81\text{m s}^{-2}$. Let the gravity direction resolved in $\mathcal{F}_{B_0}$ be $\mathbf{g}'_{B_0} = \frac{\mathbf{g}_{B_0}}{g}$. Then, the gravity direction in the inertial frame $\mathcal{F}_W$ is known to be $\mathbf{g}'_W = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^\mathsf{T}$. The attitude of the initial body frame $\mathcal{F}_{B_0}$ with respect to $\mathcal{F}_W$ can be defined as

$$\mathbf{C}_{WB_0} = \operatorname{Exp}\left(\boldsymbol{\phi}_{B_0}\right) = \operatorname{Exp}\left(\phi\mathbf{a}\right), \tag{4.38}$$

where

$$\mathbf{a} = \frac{\mathbf{g}'_{B_0}{}^{\times}\mathbf{g}'_{B_0}}{\left\|\mathbf{g}'_{B_0}{}^{\times}\mathbf{g}'_{B_0}\right\|}, \tag{4.39}$$

$$\phi = \arctan\left(\frac{\left\|\mathbf{g}'_{B_0}{}^{\times}\mathbf{g}'_{B_0}\right\|}{\mathbf{g}'_{B_0}{}^{\mathsf{T}}\mathbf{g}'_{B_0}}\right). \tag{4.40}$$

Thus the gravity in the initial body frame can be rewritten as

$$\mathbf{g}_{B_0} = g\mathbf{C}_{WB_0}^{\mathsf{T}}\mathbf{g}_W. \tag{4.41}$$

However, any rotation around the gravity direction will not be observable. Hence, the perturbation around this initial attitude is going to have a form

$$\mathbf{C}_{WB_0} = \bar{\mathbf{C}}_{WB_0}\mathrm{Exp}\left(\delta\boldsymbol{\phi}_{B_0}\right), \tag{4.42}$$

where $\delta\boldsymbol{\phi}_{B_0} = \begin{bmatrix}\delta\boldsymbol{\phi}_{xy}^{\mathsf{T}} & 0\end{bmatrix}^{\mathsf{T}}$. Thus the first order approximation of the gravity in the initial body frame is written as

$$\begin{aligned}
\mathbf{g}_{B_0} &= g\mathrm{Exp}\left(-\delta\boldsymbol{\phi}_{B_0}\right)\bar{\mathbf{C}}_{WB_0}^{\mathsf{T}}\mathbf{g}_W' && (4.43)\\
&= g\left(\mathbf{1} - \delta\boldsymbol{\phi}_{B_0}^{\times}\right)\bar{\mathbf{C}}_{WB_0}^{\mathsf{T}}\mathbf{g}_W' && (4.44)\\
&= g\bar{\mathbf{C}}_{WB_0}^{\mathsf{T}}\mathbf{g}_W' + g\left(\bar{\mathbf{C}}_{WB_0}^{\mathsf{T}}\mathbf{g}_W'\right)^{\times}\delta\boldsymbol{\phi}_{B_0}. && (4.45)
\end{aligned}$$

Substituting this gravity into (4.23) and (4.25) yields

$$\mathbf{r}_{B_0}^{b_2b_0} = \mathbf{r}_{B_0}^{b_1b_0} + \mathbf{v}_{B_0}^{b_1b_0/B_0}\Delta t_{12} + \tfrac{1}{2}\left(g\bar{\mathbf{C}}_{WB_0}^{\mathsf{T}}\mathbf{g}_W' + g\left(\bar{\mathbf{C}}_{WB_0}^{\mathsf{T}}\mathbf{g}_W'\right)^{\times}\delta\boldsymbol{\phi}_{B_0}\right)\Delta t_{12}^2 + \mathbf{C}_{B_0B_1}\Delta\mathbf{r}_{12}, \tag{4.46}$$

$$\mathbf{v}_{B_0}^{b_2b_0/B_0} = \mathbf{v}_{B_0}^{b_1b_0/B_0} + \left(g\bar{\mathbf{C}}_{WB_0}^{\mathsf{T}}\mathbf{g}_W' + g\left(\bar{\mathbf{C}}_{WB_0}^{\mathsf{T}}\mathbf{g}_W'\right)^{\times}\delta\boldsymbol{\phi}_{B_0}\right)\Delta t_{12} + \mathbf{C}_{B_0B_1}\Delta\mathbf{v}_{12}. \tag{4.47}$$

Isolating the $\delta\boldsymbol{\phi}_{B_0}$ and the velocities on the left hand side results

$$\mathbf{a}_{12}\begin{bmatrix}\mathbf{v}_{B_0}^{b_1b_0/B_0}\\ \mathbf{v}_{B_0}^{b_2b_0/B_0}\\ \delta\boldsymbol{\phi}_{xy}\end{bmatrix} = \mathbf{b}_{12}, \tag{4.48}$$

where

$$\mathbf{a}_{12} = \begin{bmatrix}\mathbf{1}\Delta t_{12} & \mathbf{0} & \left(\frac{1}{2}g\left(\bar{\mathbf{C}}_{WB_0}^{\mathsf{T}}\mathbf{g}_W'\right)^{\times}\Delta t_{12}^2\right)_{(:,1,2)}\\ \mathbf{1} & -\mathbf{1} & \left(g\left(\bar{\mathbf{C}}_{WB_0}^{\mathsf{T}}\mathbf{g}_W'\right)^{\times}\Delta t_{12}\right)_{(:,1,2)}\end{bmatrix}, \tag{4.49}$$

and

$$\mathbf{b}_{12} = \begin{bmatrix}\mathbf{r}_{B_0}^{b_2b_0} - \mathbf{r}_{B_0}^{b_1b_0} - \mathbf{C}_{B_0B_1}\Delta\mathbf{r}_{12} - \frac{1}{2}\bar{\mathbf{g}}_{B_0}\Delta t_{12}^2\\ -\mathbf{C}_{B_0B_1}\Delta\mathbf{v}_{12} - \bar{\mathbf{g}}_{B_0}\Delta t_{12}\end{bmatrix}. \tag{4.50}$$

The subscript $(\cdot)_{(:,1,2)}$ indicates that only the first two columns of the matrix are kept. Again, the system is expanded to every doublet in the window. However, in this case, every doublet has different velocities to be estimated. The normal equations matrices are used

$$\mathbf{A}_{12} = \mathbf{a}_{12}^{\mathsf{T}}\mathbf{a}_{12} = \begin{bmatrix} \mathbf{A}_{v_{11}} & \mathbf{A}_{v_{12}} & \mathbf{A}_{v_{1}g} \\ \mathbf{A}_{v_{21}} & \mathbf{A}_{v_{22}} & \mathbf{A}_{v_{2}g} \\ \mathbf{A}_{gv_{1}} & \mathbf{A}_{gv_{2}} & \mathbf{A}_{g} \end{bmatrix} \in \mathbb{R}^{8\times 8}, \tag{4.51}$$

$$\mathbf{B}_{12} = \mathbf{a}_{12}^{\mathsf{T}}\mathbf{b}_{12} = \begin{bmatrix} \mathbf{b}_{v_1} \\ \mathbf{b}_{v_2} \\ \mathbf{b}_{g} \end{bmatrix} \in \mathbb{R}^{8\times 1}. \tag{4.52}$$

The normal equations matrices are concatenated into a single matrix,

$$\begin{bmatrix} \sum \mathbf{A}_{v_{11}} & \sum \mathbf{A}_{v_{12}} & \cdots & \sum \mathbf{A}_{v_{1N}} & \sum \mathbf{A}_{v_{1}g} \\ \sum \mathbf{A}_{v_{21}} & \sum \mathbf{A}_{v_{22}} & \cdots & \sum \mathbf{A}_{v_{2N}} & \sum \mathbf{A}_{v_{2}g} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum \mathbf{A}_{gv_{1}} & \sum \mathbf{A}_{gv_{2}} & \cdots & \sum \mathbf{A}_{v_{NN}} & \sum \mathbf{A}_{g} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{B_0}^{b_1 b_0/B_0} \\ \mathbf{v}_{B_0}^{b_2 b_0/B_0} \\ \vdots \\ \mathbf{v}_{B_0}^{b_N b_0/B_0} \\ \delta\boldsymbol{\phi}_{xy} \end{bmatrix} = \begin{bmatrix} \sum \mathbf{b}_{v_1} \\ \sum \mathbf{b}_{v_2} \\ \vdots \\ \sum \mathbf{b}_{v_N} \\ \sum \mathbf{b}_{g} \end{bmatrix}. \tag{4.53}$$

This linear system is solved iteratively. Once the gravity vector is refined in the initial body frame, $\mathbf{C}_{WB_0}$ can be used to transform all the robot poses relative to the gravity-aligned inertial frame.

4.1.4 Accelerometer Bias Estimation

Once the gravity vector is refined, substituting the first order bias update (3.101) into (4.34) results in

$$\mathbf{b}_{123} = \left(\mathbf{r}_{B_0}^{b_1 b_0} - \mathbf{r}_{B_0}^{b_2 b_0}\right)\Delta t_{23} - \left(\mathbf{r}_{B_0}^{b_2 b_0} - \mathbf{r}_{B_0}^{b_3 b_0}\right)\Delta t_{12} + \tag{4.54}$$

$$\mathbf{C}_{B_0 B_1}\left(\bar{\Delta}\mathbf{r}_{12} + \frac{\partial \Delta \mathbf{r}_{12}}{\partial \boldsymbol{\beta}_B^a}\delta\boldsymbol{\beta}_B^a\right)\Delta t_{23} - \tag{4.55}$$

$$\mathbf{C}_{B_0 B_2}\left(\bar{\Delta}\mathbf{r}_{23} + \frac{\partial \Delta \mathbf{r}_{23}}{\partial \boldsymbol{\beta}_B^a}\delta\boldsymbol{\beta}_B^a\right)\Delta t_{12} - \tag{4.56}$$

$$\mathbf{C}_{B_0 B_1}\left(\bar{\Delta}\mathbf{v}_{12} + \frac{\partial \Delta \mathbf{v}_{12}}{\partial \boldsymbol{\beta}_B^a}\delta\boldsymbol{\beta}_B^a\right)\Delta t_{12}\Delta t_{23}. \tag{4.57}$$

Again, assuming constant bias over the initial window, rearranging (4.54) such that the $\delta\boldsymbol{\beta}_B^a$ is on the left hand side yields

$$\tilde{a}_{123}\delta\boldsymbol{\beta}_B^a = \tilde{\mathbf{b}}_{123}, \tag{4.58}$$

where

$$\tilde{a}_{123} = \left(-\mathbf{C}_{B_0B_1} \frac{\partial \Delta \mathbf{r}_{12}}{\partial \boldsymbol{\beta}^a_B} \Delta t_{23} + \mathbf{C}_{B_0B_2} \frac{\partial \Delta \mathbf{r}_{23}}{\partial \boldsymbol{\beta}^a_B} \Delta t_{12} + \mathbf{C}_{B_0B_1} \frac{\partial \Delta \mathbf{v}_{12}}{\partial \boldsymbol{\beta}^a_B} \Delta t_{12} \Delta_{23} \right), \tag{4.59}$$

and

$$\tilde{\mathbf{b}}_{123} = -\tfrac{1}{2}\mathbf{g}_{B_0} \left(\Delta t^2_{12} \Delta t_{23} + \Delta t_{12} \Delta t^2_{23}\right) + \left(\mathbf{r}^{b_1b_0}_{B_0} - \mathbf{r}^{b_2b_0}_{B_0}\right) \Delta t_{23} - \left(\mathbf{r}^{b_2b_0}_{B_0} - \mathbf{r}^{b_3b_0}_{B_0}\right) \Delta t_{12} + \tag{4.60}$$

$$\mathbf{C}_{B_0B_1} \bar{\Delta} \mathbf{r}_{12} \Delta t_{23} - \mathbf{C}_{B_0B_2} \bar{\Delta} \mathbf{r}_{23} \Delta t_{12} - \mathbf{C}_{B_0B_1} \bar{\Delta} \mathbf{v}_{12} \Delta t_{12} \Delta t_{23}. \tag{4.61}$$

Similar to solving gravity vector estimation, to estimate $\delta \boldsymbol{\beta}^a_B$, linear expressions for the remaining consecutive sweep triplets in the window are concatenated in a matrix.

4.2 Sliding Window Filter

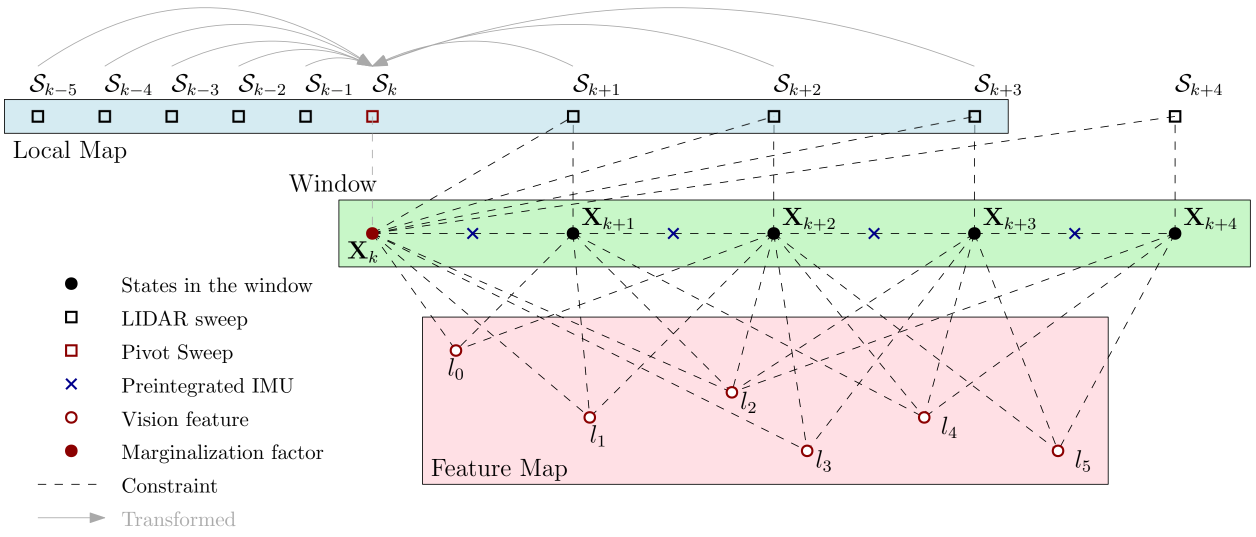


Figure 4.1: Illustration of proposed sliding window filter.

In this section, the sliding window-based tightly-coupled LIDAR-Visual-Inertial Odometry and Mapping (LIVOM) solution is presented for high accuracy and robust state estimation. After estimator initialization, the robot states are estimated using sliding window filter (SWF) as shown in Figure 4.1. The filter uses the sensor factors that constrains poses in the window as well as the marginalization factor. The states in the window are

$$\mathcal{X} = \{\mathbf{X}_k, \mathbf{X}_{k+1}, \ldots, \mathbf{X}_{k+N}, \lambda_j, \lambda_{j+1}, \ldots, \lambda_{j+M}\}, \tag{4.62}$$

where

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{C}_{WB_k} & \mathbf{v}_W^{b_k w/W} & \mathbf{r}_W^{b_k w} & & & \\ \mathbf{0} & 1 & & & & \\ \mathbf{0} & & 1 & & & \\ & & & \mathbf{1} & \boldsymbol{\beta}_{B_k}^a & \boldsymbol{\beta}_{B_k}^g \\ & & & \mathbf{0} & 1 & 0 \\ & & & \mathbf{0} & 0 & 1 \end{bmatrix} \tag{4.63}$$

is the state in a matrix Lie group that contains position, velocity and orientation of the robot as well as the IMU accelerometer and gyroscope biases. The states in vector space can be obtained via

$$\mathbf{x}_k = \begin{bmatrix} \boldsymbol{\phi}_{WB_k}{}^{\mathsf{T}} & \mathbf{v}_W^{b_k w/W\mathsf{T}} & \mathbf{r}_W^{b_k w\mathsf{T}} & \boldsymbol{\beta}_{B_k}^a{}^{\mathsf{T}} & \boldsymbol{\beta}_{B_k}^g{}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{15\times 1}, \tag{4.64}$$

where $\boldsymbol{\phi}_{WB_k} = \mathrm{Log}\left(\mathbf{C}_{WB_k}\right)$. Similar to errors defined in Section 3.1.3.2, the state error is defined by matrix multiplication $\delta\mathbf{X} = \bar{\mathbf{X}}^{-1}\mathbf{X}$, and each element of the matrix is given as

$$\delta\mathbf{C}_B = \bar{\mathbf{C}}_{WB}^{\mathsf{T}}\mathbf{C}_{WB}, \tag{4.65}$$

$$\delta\mathbf{v}_B = \bar{\mathbf{C}}_{WB}^{\mathsf{T}}\left(\mathbf{v}_W^{bw/W} - \bar{\mathbf{v}}_W^{bw/W}\right), \tag{4.66}$$

$$\delta\mathbf{r}_B = \bar{\mathbf{C}}_{WB}^{\mathsf{T}}\left(\mathbf{r}_W^{bw} - \bar{\mathbf{r}}_W^{bw}\right), \tag{4.67}$$

$$\delta\boldsymbol{\beta}_B^g = \boldsymbol{\beta}_B^g - \bar{\boldsymbol{\beta}}_B^g, \tag{4.68}$$

$$\delta\boldsymbol{\beta}_B^a = \boldsymbol{\beta}_B^a - \bar{\boldsymbol{\beta}}_B^a. \tag{4.69}$$

Additionally, λ_j is the inverse depth value of j^{th} vision feature which is optimized along with the robot states. N is the total number of states, and M is the total number of visual features in the window. The factor graph is optimized by forming a least squares problem

$$\underset{\mathcal{X}}{\arg\min}\ J\left(\mathcal{X}\right), \tag{4.70}$$

where

$$J\left(\mathcal{X}\right) = \tfrac{1}{2}\left\|\mathbf{b}_p - \mathbf{A}_p\mathbf{X}_k\right\|^2 + \tfrac{1}{2}\left\|\mathbf{e}_{\mathcal{I}}\left(\mathcal{X}\right)\right\|_{\boldsymbol{\Sigma}_{\mathcal{I}}}^2 + \tfrac{1}{2}\left\|\mathbf{e}_{\mathcal{C}}\left(\mathcal{X}\right)\right\|_{\boldsymbol{\Sigma}_{\mathcal{C}}}^2 + \tfrac{1}{2}\left\|\mathbf{e}_{\mathcal{L}}\left(\mathcal{X}\right)\right\|_{\boldsymbol{\Sigma}_{\mathcal{L}}}^2, \tag{4.71}$$

and $\mathbf{b}_p$ and $\mathbf{A}_p$ are the prior information from marginalization. Also, $\mathbf{e}_{\mathcal{I}}\left(\mathcal{X}\right)$, $\mathbf{e}_{\mathcal{C}}\left(\mathcal{X}\right)$, and $\mathbf{e}_{\mathcal{L}}\left(\mathcal{X}\right)$ are preintegrated IMU measurement factor, depth-based vision factor, and LIDAR point cloud registration factor respectively. The optimization problem is solved using the

Gauss-Newton method. The linearized objective function is given as

$$J\left(\bar{\mathcal{X}} + \delta\mathcal{X}\right) = \tfrac{1}{2}\left\|\mathbf{b}_p - \mathbf{A}_p\left(\mathbf{X}_k + \delta\mathbf{X}_k\right)\right\|^2 + \tfrac{1}{2}\mathbf{e}_{\mathcal{I}}^{\mathsf{T}}\left(\bar{\mathcal{X}} + \delta\mathcal{X}\right)\mathbf{\Sigma}_{\mathcal{I}}^{-1}\mathbf{e}_{\mathcal{I}}\left(\bar{\mathcal{X}} + \delta\mathcal{X}\right) + \tag{4.72}$$

$$\tfrac{1}{2}\mathbf{e}_{\mathcal{C}}^{\mathsf{T}}\left(\bar{\mathcal{X}} + \delta\mathcal{X}\right)\mathbf{\Sigma}_{\mathcal{C}}^{-1}\mathbf{e}_{\mathcal{C}}\left(\bar{\mathcal{X}} + \delta\mathcal{X}\right) + \tfrac{1}{2}\mathbf{e}_{\mathcal{L}}^{\mathsf{T}}\left(\bar{\mathcal{X}} + \delta\mathcal{X}\right)\mathbf{\Sigma}_{\mathcal{L}}^{-1}\mathbf{e}_{\mathcal{L}}\left(\bar{\mathcal{X}} + \delta\mathcal{X}\right). \tag{4.73}$$

Linearization of the objective function requires the linearization of the error residuals of each factor. In the following sections, linearization of each factor is discussed. For brevity of notation, $\mathbf{C}_j = \mathbf{C}_{WB_j}$, $\mathbf{v}_j = \mathbf{v}_W^{b_jw/W}$, $\mathbf{r}_j = \mathbf{r}_W^{b_jw}$ are used.

4.2.1 Preintegrated IMU Factor

The IMU cost function is given as

$$J_{\mathcal{I}}\left(\mathcal{X}\right) = \sum_{k\in\mathcal{B}} J_{\mathcal{I},k}\left(\mathbf{X}_k, \mathbf{X}_{k+1}\right), \tag{4.74}$$

where

$$J_{\mathcal{I},k}\left(\mathbf{X}_k, \mathbf{X}_{k+1}\right) = \tfrac{1}{2}\mathbf{e}_{\mathcal{I},k}^{\mathsf{T}}\left(\mathbf{X}_k, \mathbf{X}_{k+1}\right)\mathbf{\Sigma}_{\mathcal{I},kk+1}^{-1}\mathbf{e}_{\mathcal{I},k}\left(\mathbf{X}_k, \mathbf{X}_{k+1}\right), \tag{4.75}$$

and $\mathcal{B}$ indexes body states in the window. Recall the preintegrated IMU factor residual that links two states $\mathbf{X}_i$ and $\mathbf{X}_j$ from Section 3.1.6 is given by

$$\mathbf{e}_{\mathcal{I},i} = \begin{bmatrix}\mathbf{e}_{\Delta\mathbf{C}_{ij}}^{\mathsf{T}} & \mathbf{e}_{\Delta\mathbf{v}_{ij}}^{\mathsf{T}} & \mathbf{e}_{\Delta\mathbf{r}_{ij}}^{\mathsf{T}} & \mathbf{e}_{\boldsymbol{\beta}_{ij}^{a}}^{\mathsf{T}} & \mathbf{e}_{\boldsymbol{\beta}_{ij}^{g}}^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}}. \tag{4.76}$$

The preintegrated IMU factor residual is perturbed using the error defined by matrix multiplication. The analytic expressions for the Jacobian matrices are derived in the following sections.

4.2.1.1 Rotation Jacobian

The residual associated with the rotation was earlier defined as

$$\mathbf{e}_{\Delta\mathbf{C}_{ij}} = \operatorname{Log}\left(\Delta\hat{\mathbf{C}}_{ij}^{\mathsf{T}}\mathbf{C}_i^{\mathsf{T}}\mathbf{C}_j\right). \tag{4.77}$$

By substituting the rotation error (4.65) and gyroscope bias error (4.69), and by defining the nominal error in a matrix Lie group $\bar{\mathbf{E}}_{\Delta\mathbf{C}_{ij}} = \Delta\bar{\mathbf{C}}_{ij}^{\mathsf{T}}\bar{\mathbf{C}}_i^{\mathsf{T}}\bar{\mathbf{C}}_j$ defined earlier, (4.77) can be rewritten as

$$\mathbf{e}_{\Delta\mathbf{C}_{ij}} = \operatorname{Log}\left(\left(\bar{\Delta\mathbf{C}}_{ij}\operatorname{Exp}\left(\frac{\partial\Delta\mathbf{C}_{ij}}{\partial\boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i}\right)\right)^{\mathsf{T}}\left(\bar{\mathbf{C}}_i\delta\mathbf{C}_i\right)^{\mathsf{T}}\bar{\mathbf{C}}_j\delta\mathbf{C}_j\right) \tag{4.78}$$

$$= \operatorname{Log}\left(\operatorname{Exp}\left(\frac{\partial\Delta\mathbf{C}_{ij}}{\partial\boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i}\right)^{\mathsf{T}}\bar{\Delta\mathbf{C}}^{\mathsf{T}}_{ij}\delta\mathbf{C}^{\mathsf{T}}_i\bar{\mathbf{C}}^{\mathsf{T}}_i\bar{\mathbf{C}}_j\delta\mathbf{C}_j\right) \tag{4.79}$$

$$= \operatorname{Log}\left(\operatorname{Exp}\left(-\frac{\partial\Delta\mathbf{C}_{ij}}{\partial\boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i}\right)\bar{\Delta\mathbf{C}}^{\mathsf{T}}_{ij}\bar{\mathbf{C}}^{\mathsf{T}}_i\bar{\mathbf{C}}_j\operatorname{Exp}\left(-\left(\bar{\mathbf{C}}^{\mathsf{T}}_i\bar{\mathbf{C}}_j\right)^{\mathsf{T}}\delta\boldsymbol{\phi}_i\right)\operatorname{Exp}\left(\delta\boldsymbol{\phi}_j\right)\right), \tag{4.80}$$

$$= \operatorname{Log}\left(\bar{\mathbf{E}}_{\Delta\mathbf{C}_{ij}}\operatorname{Exp}\left(-\bar{\mathbf{E}}^{\mathsf{T}}_{\Delta\mathbf{C}_{ij}}\frac{\partial\Delta\mathbf{C}_{ij}}{\partial\boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i}\right)\operatorname{Exp}\left(-\left(\bar{\mathbf{C}}^{\mathsf{T}}_i\bar{\mathbf{C}}_j\right)^{\mathsf{T}}\delta\boldsymbol{\phi}_i\right)\operatorname{Exp}\left(\delta\boldsymbol{\phi}_j\right)\right). \tag{4.81}$$

By defining the nominal error in a matrix Lie algebra $\bar{\mathbf{e}}_{\Delta\mathbf{C}_{ij}} = \operatorname{Log}\left(\bar{\mathbf{E}}_{\Delta\mathbf{C}_{ij}}\right)$ and using the BCH formula from Definition 2.2.1, (4.81) can be approximated as

$$\mathbf{e}_{\Delta\mathbf{C}_{ij}} \approx \bar{\mathbf{e}}_{\Delta\mathbf{C}_{ij}} + \tag{4.82}$$

$$\mathbf{J}_r^{-1}\left(\bar{\mathbf{e}}_{\Delta\mathbf{C}_{ij}}\right)\operatorname{Log}\left(\operatorname{Exp}\left(-\bar{\mathbf{E}}^{\mathsf{T}}_{\Delta\mathbf{C}_{ij}}\frac{\partial\Delta\mathbf{C}_{ij}}{\partial\boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i}\right)\operatorname{Exp}\left(-\left(\bar{\mathbf{C}}^{\mathsf{T}}_i\bar{\mathbf{C}}_j\right)^{\mathsf{T}}\delta\boldsymbol{\phi}_{B_i}\right)\operatorname{Exp}\left(\delta\boldsymbol{\phi}_j\right)\right) \tag{4.83}$$

$$= \bar{\mathbf{e}}_{\Delta\mathbf{C}_{ij}} + \mathbf{J}_r^{-1}\left(\bar{\mathbf{e}}_{\Delta\mathbf{C}_{ij}}\right)\left(-\bar{\mathbf{E}}^{\mathsf{T}}_{\Delta\mathbf{C}_{ij}}\frac{\partial\Delta\mathbf{C}_{ij}}{\partial\boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i} - \bar{\mathbf{C}}^{\mathsf{T}}_j\bar{\mathbf{C}}_i\delta\boldsymbol{\phi}_i + \delta\boldsymbol{\phi}_j\right) \tag{4.84}$$

The matrix logarithm of the perturbations are assumed very small. Further, the nominal residual is assumed to be very close to zero. Thus, by neglecting the higher order terms, $\mathbf{J}_r^{-1}\left(\bar{\mathbf{e}}_{\Delta\mathbf{C}_{ij}}\right) \approx \mathbf{1}$. Therefore, the linearized rotation residual is

$$\mathbf{e}_{\Delta\mathbf{C}_{ij}} \approx \bar{\mathbf{e}}_{\Delta\mathbf{C}_{ij}} - \bar{\mathbf{C}}^{\mathsf{T}}_j\bar{\mathbf{C}}_i\bar{\Delta\mathbf{C}}_{ij}\frac{\partial\Delta\mathbf{C}_{ij}}{\partial\boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i} - \bar{\mathbf{C}}^{\mathsf{T}}_j\bar{\mathbf{C}}_i\delta\boldsymbol{\phi}_i + \delta\boldsymbol{\phi}_j. \tag{4.85}$$

4.2.1.2 Velocity Jacobian

The residual associated with the velocity was earlier defined as

$$\mathbf{e}_{\Delta\mathbf{v}_{ij}} = \mathbf{C}^{\mathsf{T}}_i\left(\mathbf{v}_j - \mathbf{v}_i - \Delta t_{ij}\mathbf{g}_W\right) - \hat{\Delta\mathbf{v}}_{ij}. \tag{4.86}$$

Substituting the velocity error (4.66) as well as the rotation error (4.65) into (4.86) yields

$$\mathbf{e}_{\Delta\mathbf{v}_{ij}} = \left(\bar{\mathbf{C}}_i\delta\mathbf{C}_i\right)^{\mathsf{T}}\left(\bar{\mathbf{v}}_j + \bar{\mathbf{C}}_j\delta\mathbf{v}_j - \bar{\mathbf{v}}_i - \bar{\mathbf{C}}_i\delta\mathbf{v}_i - \Delta t_{ij}\mathbf{g}_W\right) - \hat{\Delta\mathbf{v}}_{ij} \tag{4.87}$$

$$= \operatorname{Exp}\left(-\delta\boldsymbol{\phi}_i\right)\bar{\mathbf{C}}^{\mathsf{T}}_i\left(\bar{\mathbf{v}}_j - \bar{\mathbf{v}}_i - \Delta t_{ij}\mathbf{g}_W + \bar{\mathbf{C}}_j\delta\mathbf{v}_j - \bar{\mathbf{C}}_i\delta\mathbf{v}_i\right) - \hat{\Delta\mathbf{v}}_{ij}, \tag{4.88}$$

To linearize (4.88), let $\operatorname{Exp}(-\delta\boldsymbol{\phi}_i) \approx \mathbf{1} - \delta\boldsymbol{\phi}_i^\times$ and use the first order update of the biases

$$\hat{\Delta\mathbf{v}}_{ij} \approx \bar{\Delta\mathbf{v}}_{ij} + \frac{\partial \Delta\mathbf{v}_{ij}}{\partial \boldsymbol{\beta}^a_{B_i}}\delta\boldsymbol{\beta}^a_{B_i} + \frac{\partial \Delta\mathbf{v}_{ij}}{\partial \boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i}. \tag{4.89}$$

Neglecting the higher order terms, (4.88) becomes

$$\begin{aligned}
\mathbf{e}_{\Delta\mathbf{v}_{ij}} &\approx \left(\mathbf{1} - \delta\boldsymbol{\phi}_i^\times\right)\bar{\mathbf{C}}_i^\mathsf{T}\left(\bar{\mathbf{v}}_j - \bar{\mathbf{v}}_i - \Delta t_{ij}\mathbf{g}_W + \bar{\mathbf{C}}_j\delta\mathbf{v}_j - \bar{\mathbf{C}}_i\delta\mathbf{v}_i\right) - &(4.90)\\
&\qquad\left(\bar{\Delta\mathbf{v}}_{ij} + \frac{\partial \Delta\mathbf{v}_{ij}}{\partial \boldsymbol{\beta}^a_{B_i}}\delta\boldsymbol{\beta}^a_{B_i} + \frac{\partial \Delta\mathbf{v}_{ij}}{\partial \boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i}\right) &(4.91)\\
&= \bar{\mathbf{C}}_i^\mathsf{T}\left(\bar{\mathbf{v}}_j - \bar{\mathbf{v}}_i - \Delta t_{ij}\mathbf{g}_W\right) - \bar{\Delta\mathbf{v}}_{ij} + \bar{\mathbf{C}}_i^\mathsf{T}\bar{\mathbf{C}}_j\delta\mathbf{v}_j - \delta\mathbf{v}_i - &(4.92)\\
&\qquad\delta\boldsymbol{\phi}_i^\times\bar{\mathbf{C}}_i^\mathsf{T}\left(\bar{\mathbf{v}}_j - \bar{\mathbf{v}}_i - \Delta t_{ij}\mathbf{g}_W\right) - \frac{\partial \Delta\mathbf{v}_{ij}}{\partial \boldsymbol{\beta}^a_{B_i}}\delta\boldsymbol{\beta}^a_{B_i} - \frac{\partial \Delta\mathbf{v}_{ij}}{\partial \boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i} &(4.93)\\
&= \bar{\mathbf{e}}_{\Delta\mathbf{v}_{ij}} + \bar{\mathbf{C}}_i^\mathsf{T}\bar{\mathbf{C}}_j\delta\mathbf{v}_j - \delta\mathbf{v}_i + \left(\bar{\mathbf{C}}_i^\mathsf{T}\left(\bar{\mathbf{v}}_j - \bar{\mathbf{v}}_i - \Delta t_{ij}\mathbf{g}_W\right)\right)^\times\delta\boldsymbol{\phi}_i - \frac{\partial \Delta\mathbf{v}_{ij}}{\partial \boldsymbol{\beta}^a_{B_i}}\delta\boldsymbol{\beta}^a_{B_i} - \frac{\partial \Delta\mathbf{v}_{ij}}{\partial \boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i}. &(4.94)
\end{aligned}$$

4.2.1.3 Position Jacobian

Similar to velocity Jacobian derivation, recall the residual associated with the position defined earlier

$$\mathbf{e}_{\Delta\mathbf{r}_{ij}} = \mathbf{C}_i^\mathsf{T}\left(\mathbf{r}_j - \mathbf{r}_i - \Delta t_{ij}\mathbf{v}_i - \tfrac{1}{2}\Delta t_{ij}^2\mathbf{g}_W\right) - \hat{\Delta\mathbf{r}}_{ij}. \tag{4.95}$$

Substituting the position error defined by the matrix multiplication (4.67) as well as the rotation and velocity errors into (4.95) yields

$$\begin{aligned}
\mathbf{e}_{\Delta\mathbf{r}_{ij}} &= \left(\bar{\mathbf{C}}_i\delta\mathbf{C}_i\right)^\mathsf{T}\left(\bar{\mathbf{r}}_j + \bar{\mathbf{C}}_j\delta\mathbf{r}_j - \bar{\mathbf{r}}_i - \bar{\mathbf{C}}_i\delta\mathbf{r}_i - \Delta t_{ij}\left(\bar{\mathbf{v}}_i + \bar{\mathbf{C}}_i\delta\mathbf{v}_i\right) - \tfrac{1}{2}\Delta t_{ij}^2\mathbf{g}_W\right) - \hat{\Delta\mathbf{r}}_{ij} &(4.96)\\
&= \operatorname{Exp}(-\delta\boldsymbol{\phi}_i)\,\bar{\mathbf{C}}_i^\mathsf{T}\left(\bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i - \Delta t_{ij}\bar{\mathbf{v}}_i - \tfrac{1}{2}\Delta t_{ij}^2\mathbf{g}_W\right) + &(4.97)\\
&\qquad\operatorname{Exp}(-\delta\boldsymbol{\phi}_i)\,\bar{\mathbf{C}}_i^\mathsf{T}\left(\bar{\mathbf{C}}_j\delta\mathbf{r}_j - \bar{\mathbf{C}}_i\delta\mathbf{r}_i - \Delta t_{ij}\bar{\mathbf{C}}_i\delta\mathbf{v}_i\right) - \hat{\Delta\mathbf{r}}_{ij}. &(4.98)
\end{aligned}$$

To linearize (4.97), let $\operatorname{Exp}(-\delta\boldsymbol{\phi}_i) \approx \mathbf{1} - \delta\boldsymbol{\phi}_i^\times$ and use the first order update of the biases

$$\hat{\Delta\mathbf{r}}_{ij} \approx \bar{\Delta\mathbf{r}}_{ij} + \frac{\partial \Delta\mathbf{r}_{ij}}{\partial \boldsymbol{\beta}^a_{B_i}}\delta\boldsymbol{\beta}^a_{B_i} + \frac{\partial \Delta\mathbf{r}_{ij}}{\partial \boldsymbol{\beta}^g_{B_i}}\delta\boldsymbol{\beta}^g_{B_i}. \tag{4.99}$$

Neglecting the higher order terms, (4.97) becomes

$$\mathbf{e}_{\Delta \mathbf{r}_{ij}} \approx \left(\mathbf{1} - \delta\boldsymbol{\phi}_i^\times\right) \bar{\mathbf{C}}_i^\mathsf{T} \left(\bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i - \Delta t_{ij}\bar{\mathbf{v}}_i - \tfrac{1}{2}\Delta t_{ij}^2 \mathbf{g}_W\right) + \tag{4.100}$$

$$\left(\mathbf{1} - \delta\boldsymbol{\phi}_i^\times\right) \bar{\mathbf{C}}_i^\mathsf{T} \left(\bar{\mathbf{C}}_j \delta\mathbf{r}_j - \bar{\mathbf{C}}_i \delta\mathbf{r}_i - \Delta t_{ij}\bar{\mathbf{C}}_i \delta\mathbf{v}_i\right) - \bar{\Delta\mathbf{r}}_{ij} - \frac{\partial \Delta\mathbf{r}_{ij}}{\partial \boldsymbol{\beta}_{B_i}^a}\delta\boldsymbol{\beta}_{B_i}^a - \frac{\partial \Delta\mathbf{r}_{ij}}{\partial \boldsymbol{\beta}_{B_i}^g}\delta\boldsymbol{\beta}_{B_i}^g \tag{4.101}$$

$$= \bar{\mathbf{C}}_i^\mathsf{T} \left(\bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i - \Delta t_{ij}\bar{\mathbf{v}}_i - \tfrac{1}{2}\Delta t_{ij}^2 \mathbf{g}_W\right) - \bar{\Delta\mathbf{r}}_{ij} + \bar{\mathbf{C}}_i^\mathsf{T}\bar{\mathbf{C}}_j \delta\mathbf{r}_j - \delta\mathbf{r}_i - \Delta t_{ij}\delta\mathbf{v}_i - \tag{4.102}$$

$$\delta\boldsymbol{\phi}_i^\times \left(\bar{\mathbf{C}}_i^\mathsf{T} \left(\bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i - \Delta t_{ij}\bar{\mathbf{v}}_i - \tfrac{1}{2}\Delta t_{ij}^2 \mathbf{g}_W\right)\right) - \frac{\partial \Delta\mathbf{r}_{ij}}{\partial \boldsymbol{\beta}_{B_i}^a}\delta\boldsymbol{\beta}_{B_i}^a - \frac{\partial \Delta\mathbf{r}_{ij}}{\partial \boldsymbol{\beta}_{B_i}^g}\delta\boldsymbol{\beta}_{B_i}^g \tag{4.103}$$

$$= \bar{\mathbf{e}}_{\Delta \mathbf{r}_{ij}} + \bar{\mathbf{C}}_i^\mathsf{T}\bar{\mathbf{C}}_j \delta\mathbf{r}_j - \delta\mathbf{r}_i - \Delta t_{ij}\delta\mathbf{v}_i + \tag{4.104}$$

$$\left(\bar{\mathbf{C}}_i^\mathsf{T} \left(\bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i - \Delta t_{ij}\bar{\mathbf{v}}_i - \tfrac{1}{2}\Delta t_{ij}^2 \mathbf{g}_W\right)\right)^\times \delta\boldsymbol{\phi}_i - \frac{\partial \Delta\mathbf{r}_{ij}}{\partial \boldsymbol{\beta}_{B_i}^a}\delta\boldsymbol{\beta}_{B_i}^a - \frac{\partial \Delta\mathbf{r}_{ij}}{\partial \boldsymbol{\beta}_{B_i}^g}\delta\boldsymbol{\beta}_{B_i}^g. \tag{4.105}$$

4.2.1.4 Bias Jacobian

The bias residuals defined earlier are

$$\mathbf{e}_{\boldsymbol{\beta}_{ij}^a} = \boldsymbol{\beta}_{B_j}^a - \boldsymbol{\beta}_{B_i}^a, \tag{4.106}$$

$$\mathbf{e}_{\boldsymbol{\beta}_{ij}^g} = \boldsymbol{\beta}_{B_j}^g - \boldsymbol{\beta}_{B_i}^g. \tag{4.107}$$

Perturbing bias residual using the bias errors defined by the matrix multiplication (4.68) and (4.69) yields

$$\mathbf{e}_{\boldsymbol{\beta}_{ij}^a} = \bar{\boldsymbol{\beta}}_{B_j}^a + \delta\boldsymbol{\beta}_{B_j}^a - \bar{\boldsymbol{\beta}}_{B_i}^a - \delta\boldsymbol{\beta}_{B_i}^a \tag{4.108}$$

$$= \bar{\mathbf{e}}_{\boldsymbol{\beta}_{ij}^a} + \delta\boldsymbol{\beta}_{B_j}^a - \delta\boldsymbol{\beta}_{B_i}^a, \tag{4.109}$$

$$\mathbf{e}_{\boldsymbol{\beta}_{ij}^g} = \bar{\boldsymbol{\beta}}_{B_j}^g + \delta\boldsymbol{\beta}_{B_j}^g - \bar{\boldsymbol{\beta}}_{B_i}^g - \delta\boldsymbol{\beta}_{B_i}^g \tag{4.110}$$

$$= \bar{\mathbf{e}}_{\boldsymbol{\beta}_{ij}^g} + \delta\boldsymbol{\beta}_{B_j}^g - \delta\boldsymbol{\beta}_{B_i}^g. \tag{4.111}$$

4.2.1.5 Summary

The preintegrated IMU factor residual (4.76) is linearized such that

$$\mathbf{e}_{\mathcal{I},k}\left(\mathbf{X}_k, \mathbf{X}_{k+1}\right) = \mathbf{e}_{\mathcal{I},k}\left(\bar{\mathbf{X}}_k, \bar{\mathbf{X}}_{k+1}\right) + \frac{\partial \mathbf{e}_{\mathcal{I},k}}{\partial \mathbf{X}_k}\delta\boldsymbol{\xi}_k + \frac{\partial \mathbf{e}_{\mathcal{I},k}}{\partial \mathbf{X}_{k+1}}\delta\boldsymbol{\xi}_{k+1}, \tag{4.112}$$

where

$$\frac{\partial \mathbf{e}_{\mathcal{I},k}}{\partial \mathbf{X}_k} = \tag{4.113}$$

$$\begin{bmatrix} -\bar{\mathbf{C}}_j^{\mathsf{T}}\bar{\mathbf{C}}_i & & & & -\bar{\mathbf{E}}_{\Delta \mathbf{C}_{ij}}^{\mathsf{T}} \frac{\partial \Delta \mathbf{C}_{ij}}{\partial \boldsymbol{\beta}_{B_i}^{g}} \\ \left(\bar{\mathbf{C}}_i^{\mathsf{T}}\left(\bar{\mathbf{v}}_j - \bar{\mathbf{v}}_i - \Delta t_{ij}\mathbf{g}_W\right)\right)^{\times} & -\mathbf{1} & & -\frac{\partial \Delta \mathbf{v}_{ij}}{\partial \boldsymbol{\beta}_{B_i}^{a}} & -\frac{\partial \Delta \mathbf{v}_{ij}}{\partial \boldsymbol{\beta}_{B_i}^{g}} \\ \left(\bar{\mathbf{C}}_i^{\mathsf{T}}\left(\bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i - \Delta t_{ij}\bar{\mathbf{v}}_i - \frac{1}{2}\Delta t_{ij}^2\mathbf{g}_W\right)\right)^{\times} & -\Delta t_{ij}\mathbf{1} & -\mathbf{1} & -\frac{\partial \Delta \mathbf{r}_{ij}}{\partial \boldsymbol{\beta}_{B_i}^{a}} & -\frac{\partial \Delta \mathbf{r}_{ij}}{\partial \boldsymbol{\beta}_{B_i}^{g}} \\ & & & -\mathbf{1} & \\ & & & & -\mathbf{1} \end{bmatrix}, \tag{4.114}$$

$$\frac{\partial \mathbf{e}_{\mathcal{I},k}}{\partial \mathbf{X}_{k+1}} = \begin{bmatrix} \mathbf{1} & & & & \\ & \bar{\mathbf{C}}_i^{\mathsf{T}}\bar{\mathbf{C}}_j & & & \\ & & \bar{\mathbf{C}}_i^{\mathsf{T}}\bar{\mathbf{C}}_j & & \\ & & & \mathbf{1} & \\ & & & & \mathbf{1} \end{bmatrix}, \tag{4.115}$$

and $\delta\boldsymbol{\xi}_k = \mathrm{Log}\left(\delta\mathbf{X}_k\right)$. In addition, the weight associated with the preintegrated IMU factor $\boldsymbol{\Sigma}_{\mathcal{I},kk+1}$ is the covariance matrix obtained by (3.85) in Section 3.1.4.

4.2.2 Depth-based Vision Factor

The visual cost function is given as

$$J_{\mathcal{C}}\left(\mathcal{X}\right) = \sum_{p \in \mathcal{V}} J_{\mathcal{C},p}\left(\mathcal{X}\right), \tag{4.116}$$

where

$$J_{\mathcal{C},p}\left(\mathcal{X}\right) = \tfrac{1}{2}\mathbf{e}_{\mathcal{C},p}^{\mathsf{T}}\left(\mathcal{X}\right)\boldsymbol{\Sigma}_{\mathcal{C},k}^{-1}\mathbf{e}_{\mathcal{C},p}\left(\mathcal{X}\right), \tag{4.117}$$

and $\mathcal{V}$ indexes the vision feature points in the window. Suppose a k^{th} feature l_k is tracked in frames i and j. The depth-based vision factor residual is given as

$$\mathbf{e}_{\mathcal{C},k}\left(\mathbf{X}_i, \mathbf{X}_j\right) = \begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2\end{bmatrix}^{\mathsf{T}} \left(\frac{\check{\mathbf{r}}_{C_j}^{l_k c_j}}{r_{C_j,3}^{l_k c_j}} - \pi_c^{-1}\left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}_j\right)\right), \tag{4.118}$$

where

$$\check{\mathbf{r}}_{C_j}^{l_k c_j} = \mathbf{C}_{BC}^{\mathsf{T}} \left(\mathbf{C}_j^{\mathsf{T}} \left(\mathbf{C}_i \left(\mathbf{C}_{BC} \frac{1}{\lambda_k} \pi_c^{-1} \left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}_i \right) + \mathbf{r}_B^{bc} \right) + \mathbf{r}_i - \mathbf{r}_j \right) - \mathbf{r}_B^{bc} \right). \tag{4.119}$$

Note that only the estimated measurement $\check{\mathbf{r}}_{C_j}^{l_k c_j}$ is state dependent in the vision factor residual. Thus, to linearize the innovation equation (4.118), the chain rule is applied

$$\frac{\partial \mathbf{e}_{\mathcal{C},k}}{\partial \mathcal{X}} = \frac{\partial \mathbf{e}_{\mathcal{C}}}{\partial \check{\mathbf{r}}_{C_j}^{l_k c_j}} \frac{\partial \check{\mathbf{r}}_{C_j}^{l_k c_j}}{\partial \mathcal{X}}. \tag{4.120}$$

For simplicity, $\check{\mathbf{r}}_{C_j}^{l_k c_j} = \begin{bmatrix} x^{l_k} & y^{l_k} & z^{l_k} \end{bmatrix}^{\mathsf{T}}$. The first partial derivative is given as

$$\frac{\partial \mathbf{e}_{\mathcal{C}}}{\partial \check{\mathbf{r}}_{C_j}^{l_k c_j}} = \begin{bmatrix} \frac{1}{z^{l_k}} & 0 & -\frac{x^{l_k}}{\left(z^{l_k}\right)^2} \\ 0 & \frac{1}{z^{l_k}} & -\frac{y^{l_k}}{\left(z^{l_k}\right)^2} \end{bmatrix}. \tag{4.121}$$

The next partial derivative is obtained by perturbing $\check{\mathbf{r}}_{C_j}^{l_k c_j}$. Let the features in frame i be resolved in the body frame such that $\mathbf{r}_{B_i}^{l_k b_i} = \mathbf{C}_{BC} \frac{1}{\lambda_k} \pi_c^{-1} \left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}_i \right) + \mathbf{r}_B^{bc}$. Then, the estimated measurement now becomes

$$\check{\mathbf{r}}_{C_j}^{l_k c_j} = \mathbf{C}_{BC}^{\mathsf{T}} \left(\mathbf{C}_j^{\mathsf{T}} \left(\mathbf{C}_i \mathbf{r}_{B_i}^{l_k b_i} + \mathbf{r}_i - \mathbf{r}_j \right) - \mathbf{r}_B^{bc} \right). \tag{4.122}$$

Recall that the inverse depth λ_k is optimized along with other robot navigation states. Thus linearizing (4.122) with respect to λ_k is

$$\frac{\partial \check{\mathbf{r}}_{C_j}^{l_k c_j}}{\partial \lambda_k} = -\mathbf{C}_{BC}^{\mathsf{T}} \mathbf{C}_j^{\mathsf{T}} \mathbf{C}_i \mathbf{C}_{BC} \frac{1}{\lambda_k^2} \pi_c^{-1} \left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}_i \right). \tag{4.123}$$

Further, (4.122) is perturbed using the pose errors defined by the matrix multiplication (4.65) and (4.67),

$$\check{\mathbf{r}}_{C_j}^{l_k c_j} = \mathbf{C}_{BC}^{\mathsf{T}} \left(\left(\bar{\mathbf{C}}_j \delta \mathbf{C}_j \right)^{\mathsf{T}} \left(\bar{\mathbf{C}}_i \delta \mathbf{C}_i \mathbf{r}_{B_i}^{l_k b_i} + \bar{\mathbf{r}}_i + \bar{\mathbf{C}}_i \delta \mathbf{r}_i - \bar{\mathbf{r}}_j - \bar{\mathbf{C}}_j \delta \mathbf{r}_j \right) - \mathbf{r}_B^{bc} \right). \tag{4.124}$$

To linearize (4.124), let $\delta \mathbf{C}_j \approx \mathbf{1} + \delta \boldsymbol{\phi}_j^{\times}$. Neglecting the second order terms, (4.124) is now approximated as

$$\begin{aligned}
\check{\mathbf{r}}_{C_j}^{l_k c_j} &\approx \mathbf{C}_{BC}^{\mathsf{T}} \left(\left(\mathbf{1} - \delta \boldsymbol{\phi}_j^{\times} \right) \bar{\mathbf{C}}_j^{\mathsf{T}} \left(\bar{\mathbf{C}}_i \left(\mathbf{1} + \delta \boldsymbol{\phi}_i^{\times} \right) \mathbf{r}_{B_i}^{l_k b_i} + \bar{\mathbf{r}}_i + \bar{\mathbf{C}}_i \delta \mathbf{r}_i - \bar{\mathbf{r}}_j - \bar{\mathbf{C}}_j \delta \mathbf{r}_j \right) - \mathbf{r}_B^{bc} \right) && (4.125) \\
&= \mathbf{C}_{BC}^{\mathsf{T}} \left(\bar{\mathbf{C}}_j^{\mathsf{T}} \left(\bar{\mathbf{C}}_i \mathbf{r}_{B_i}^{l_k b_i} + \bar{\mathbf{r}}_i - \bar{\mathbf{r}}_j \right) - \mathbf{r}_B^{bc} \right) + && (4.126) \\
&\qquad \mathbf{C}_{BC}^{\mathsf{T}} \left(-\delta \boldsymbol{\phi}_j^{\times} \bar{\mathbf{C}}_j^{\mathsf{T}} \left(\bar{\mathbf{C}}_i \mathbf{r}_{B_i}^{l_k b_i} + \bar{\mathbf{r}}_i - \bar{\mathbf{r}}_j \right) + \bar{\mathbf{C}}_j^{\mathsf{T}} \bar{\mathbf{C}}_i \delta \boldsymbol{\phi}_i^{\times} \mathbf{r}_{B_i}^{l_k b_i} + \bar{\mathbf{C}}_j^{\mathsf{T}} \bar{\mathbf{C}}_i \delta \mathbf{r}_i - \delta \mathbf{r}_j \right) && (4.127) \\
&= \bar{\mathbf{r}}_{C_j}^{l_k c_j} + \mathbf{C}_{BC}^{\mathsf{T}} \left(\left(\bar{\mathbf{C}}_j^{\mathsf{T}} \left(\bar{\mathbf{C}}_i \mathbf{r}_{B_i}^{l_k b_i} + \bar{\mathbf{r}}_i - \bar{\mathbf{r}}_j \right) \right)^{\times} \delta \boldsymbol{\phi}_j - \bar{\mathbf{C}}_j^{\mathsf{T}} \bar{\mathbf{C}}_i \mathbf{r}_{B_i}^{l_k b_i \times} \delta \boldsymbol{\phi}_i + \bar{\mathbf{C}}_j^{\mathsf{T}} \bar{\mathbf{C}}_i \delta \mathbf{r}_i - \delta \mathbf{r}_j \right). && (4.128)
\end{aligned}$$

In summary, the depth-based vision factor (4.118) is linearized such that

$$\mathbf{e}_{\mathcal{C},k} \left(\bar{\mathcal{X}} + \delta \mathcal{X} \right) = \mathbf{e}_{\mathcal{C},k} \left(\bar{\mathcal{X}} \right) + \frac{\partial \mathbf{e}_{\mathcal{C},k}}{\partial \mathbf{X}_i} \delta \boldsymbol{\xi}_i + \frac{\partial \mathbf{e}_{\mathcal{C},k}}{\partial \mathbf{X}_j} \delta \boldsymbol{\xi}_j + \sum_{k \in \mathcal{Z}} \frac{\partial \mathbf{e}_{\mathcal{C},k}}{\partial \lambda_k} \delta \lambda_k, \tag{4.129}$$

where,

$$\frac{\partial \mathbf{e}_{\mathcal{C},k}}{\partial \mathbf{X}_i} = \begin{bmatrix} \frac{1}{z^{l_k}} & 0 & -\frac{x^{l_k}}{\left(z^{l_k}\right)^2} \\ 0 & \frac{1}{z^{l_k}} & -\frac{y^{l_k}}{\left(z^{l_k}\right)^2} \end{bmatrix} \begin{bmatrix} -\mathbf{C}_{BC}^{\mathsf{T}} \bar{\mathbf{C}}_j^{\mathsf{T}} \bar{\mathbf{C}}_i \mathbf{r}_{B_i}^{l_k b_i \times} & \mathbf{0} & \mathbf{C}_{BC}^{\mathsf{T}} \bar{\mathbf{C}}_j^{\mathsf{T}} \bar{\mathbf{C}}_i & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{4.130}$$

$$\frac{\partial \mathbf{e}_{\mathcal{C},k}}{\partial \mathbf{X}_j} = \begin{bmatrix} \frac{1}{z^{l_k}} & 0 & -\frac{x^{l_k}}{\left(z^{l_k}\right)^2} \\ 0 & \frac{1}{z^{l_k}} & -\frac{y^{l_k}}{\left(z^{l_k}\right)^2} \end{bmatrix} \begin{bmatrix} \mathbf{C}_{BC}^{\mathsf{T}} \left(\bar{\mathbf{C}}_j^{\mathsf{T}} \left(\bar{\mathbf{C}}_i \mathbf{r}_{B_i}^{l_k b_i} + \bar{\mathbf{r}}_i - \bar{\mathbf{r}}_j \right) \right)^{\times} & \mathbf{0} & -\mathbf{C}_{BC}^{\mathsf{T}} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{4.131}$$

$$\frac{\partial \mathbf{e}_{\mathcal{C},k}}{\partial \lambda_k} = \begin{bmatrix} \frac{1}{z^{l_k}} & 0 & -\frac{x^{l_k}}{\left(z^{l_k}\right)^2} \\ 0 & \frac{1}{z^{l_k}} & -\frac{y^{l_k}}{\left(z^{l_k}\right)^2} \end{bmatrix} - \mathbf{C}_{BC}^{\mathsf{T}} \bar{\mathbf{C}}_j^{\mathsf{T}} \bar{\mathbf{C}}_i \bar{\mathbf{C}}_{BC} \frac{1}{\lambda_k^2} \pi_c^{-1} \left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}_i \right), \tag{4.132}$$

and $\mathcal{Z}$ indexes all the features seen at both frames i and j. The weight associated with the camera factor $\boldsymbol{\Sigma}_{\mathcal{C},k}$ has a size of 2×2. In fact, the image coordinates of the feature points are affected by noise, and the effect is carried on when tracking is performed in the 3D reconstruction equations. The noise model by [36] denotes the covariance associated with a point in camera frame and allows weighting each point accordingly. In practice, computing weights for each point may be computationally heavy. Given that the outlier rejection has removed most of the non reliable points, some constant weight is given to each feature.

4.2.3 LIDAR Point Cloud Registration Factor

The LIDAR PCR factor is given as

$$J_{\mathcal{L}}(\mathcal{X}) = \sum_{\mathcal{S}\in\mathcal{M}} \sum_{p\in\mathcal{S}} J^{pl}_{\mathcal{L},\mathcal{S},p}(\mathbf{X}_i, \mathbf{X}_{\mathcal{S}}) + J^{l}_{\mathcal{L},\mathcal{S},p}(\mathbf{X}_i, \mathbf{X}_{\mathcal{S}}), \tag{4.133}$$

where $\mathcal{S}$ indexes all the points in a sweep and $\mathcal{M}$ indexes the LIDAR sweeps in the window. $\mathbf{X}_{\mathcal{S}}$ denotes the robot state at sweep $\mathcal{S}$, and $\mathbf{X}_i$ denotes the robot state where the reference point cloud is resolved in. Thus, the point-to-plane and the point-to-line cost associated with j^{th} point $\mathbf{q}_k$ in j^{th} sweep are given respectively as

$$J^{pl}_{\mathcal{L},j,k}(\mathbf{X}_i, \mathbf{X}_j) = \tfrac{1}{2}{\Sigma^{pl}_{\mathcal{L},j,k}}^{-1} {e^{pl}_{\mathcal{L},j,k}}^{2}(\mathbf{X}_i, \mathbf{X}_j), \tag{4.134}$$

$$J^{l}_{\mathcal{L},j,k}(\mathbf{X}_i, \mathbf{X}_j) = \tfrac{1}{2}{\mathbf{e}^{l}_{\mathcal{L},j,k}}^{\mathsf{T}}(\mathbf{X}_i, \mathbf{X}_j)\, {\Sigma^{pl}_{\mathcal{L},j,k}}^{-1} \mathbf{e}^{l}_{\mathcal{L},j,k}(\mathbf{X}_i, \mathbf{X}_j). \tag{4.135}$$

In this section, the corresponding point of point $\mathbf{q}_k$ is denoted as $\mathbf{p}_k$.

4.2.3.1 Point-to-line Factor

LIDAR PCR point-to-line factor residual is given as

$$\mathbf{e}^{l}_{\mathcal{L},j,k}(\mathbf{X}_i, \mathbf{X}_j) = \mathbf{e}^{p}_{\mathcal{L},j,k}(\mathbf{X}_i, \mathbf{X}_j) - \mathbf{l}^{k}_{B_i}{\mathbf{l}^{k}_{B_i}}^{\mathsf{T}} \mathbf{e}^{p}_{\mathcal{L},j,k}(\mathbf{X}_i, \mathbf{X}_j), \tag{4.136}$$

where

$$\mathbf{e}^{p}_{\mathcal{L},j,k}(\mathbf{X}_i, \mathbf{X}_j) = \mathbf{C}_i^{\mathsf{T}}\left(\mathbf{C}_j \mathbf{r}^{q_k b_j}_{B_j} + \mathbf{r}_j - \mathbf{r}_i\right) - \mathbf{r}^{p_k b_i}_{B_i} \tag{4.137}$$

is the point-to-point error. Note that only the point-to-point error is a function of states. Thus, a first order Taylor-series approximation is used in (4.137). Substituting the state error definitions (4.65) and (4.67) in (4.137) results in

$$\mathbf{e}^{p}_{\mathcal{L},j,k}(\mathbf{X}_i, \mathbf{X}_j) = \left(\bar{\mathbf{C}}_i \delta\mathbf{C}_i\right)^{\mathsf{T}}\left(\bar{\mathbf{C}}_j \delta\mathbf{C}_j \mathbf{r}^{q_k b_j}_{B_j} + \bar{\mathbf{r}}_j + \bar{\mathbf{C}}_j \delta\mathbf{r}_j - \bar{\mathbf{r}}_i - \bar{\mathbf{C}}_i \delta\mathbf{r}_i\right) - \mathbf{r}^{p_k b_i}_{B_i}. \tag{4.138}$$

To linearize (4.138), let $\delta \mathbf{C}_j \approx \mathbf{1} + \delta \boldsymbol{\phi}_j^\times$. By setting $\bar{\mathbf{e}}^p_{\mathcal{L},j,k} = \mathbf{e}^p_{\mathcal{L},j,k}\left(\bar{\mathbf{X}}_i, \bar{\mathbf{X}}_j\right)$ and neglecting the second order terms, (4.138) is approximated as

$$\mathbf{e}^p_{\mathcal{L},j,k}\left(\mathbf{X}_i, \mathbf{X}_j\right) \approx \left(\mathbf{1} - \delta\boldsymbol{\phi}_i^\times\right) \bar{\mathbf{C}}_i^\mathsf{T} \left(\bar{\mathbf{C}}_j \left(\mathbf{1} + \delta\boldsymbol{\phi}_j^\times\right) \mathbf{r}_{B_j}^{q_k b_j} + \bar{\mathbf{r}}_j + \bar{\mathbf{C}}_j \delta\mathbf{r}_j - \bar{\mathbf{r}}_i - \bar{\mathbf{C}}_i \delta \mathbf{r}_i\right) - \mathbf{r}_{B_i}^{p_k b_i} \tag{4.139}$$

$$= \bar{\mathbf{C}}_i^\mathsf{T} \left(\bar{\mathbf{C}}_j \mathbf{r}_{B_j}^{q_k b_j} + \bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i\right) - \mathbf{r}_{B_i}^{p_k b_i} - \tag{4.140}$$

$$\delta\boldsymbol{\phi}_i^\times \bar{\mathbf{C}}_i^\mathsf{T} \left(\bar{\mathbf{C}}_j \mathbf{r}_{B_j}^{q_k b_j} + \bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i\right) + \bar{\mathbf{C}}_i^\mathsf{T} \bar{\mathbf{C}}_j \delta\boldsymbol{\phi}_j^\times \mathbf{r}_{B_j}^{q_k b_j} + \bar{\mathbf{C}}_i^\mathsf{T} \bar{\mathbf{C}}_j \delta \mathbf{r}_j - \delta \mathbf{r}_i \tag{4.141}$$

$$= \bar{\mathbf{e}}^p_{\mathcal{L},j,k} + \left(\bar{\mathbf{C}}_i^\mathsf{T} \left(\bar{\mathbf{C}}_j \mathbf{r}_{B_j}^{q_k b_j} + \bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i\right)\right)^\times \delta\boldsymbol{\phi}_i - \bar{\mathbf{C}}_i^\mathsf{T} \bar{\mathbf{C}}_j \mathbf{r}_{B_j}^{q_k b_j \times} \delta\boldsymbol{\phi}_j + \bar{\mathbf{C}}_i^\mathsf{T} \bar{\mathbf{C}}_j \delta \mathbf{r}_j - \delta \mathbf{r}_i \tag{4.142}$$

$$= \bar{\mathbf{e}}^p_{\mathcal{L},j,k} + \frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_i} \delta \boldsymbol{\xi}_i + \frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_j} \delta \boldsymbol{\xi}_j, \tag{4.143}$$

where

$$\frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_i} = \begin{bmatrix} \left(\bar{\mathbf{C}}_i^\mathsf{T} \left(\bar{\mathbf{C}}_j \mathbf{r}_{B_j}^{q_k b_j} + \bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i\right)\right)^\times & \mathbf{0} & -\mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{4.144}$$

$$\frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_j} = \begin{bmatrix} -\bar{\mathbf{C}}_i^\mathsf{T} \bar{\mathbf{C}}_j \mathbf{r}_{B_j}^{q_k b_j \times} & \mathbf{0} & \bar{\mathbf{C}}_i^\mathsf{T} \bar{\mathbf{C}}_j & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{4.145}$$

Combining the linearized point-to-point error with (4.136) yields

$$\mathbf{e}^l_{\mathcal{L},j,k}\left(\mathbf{X}_i, \mathbf{X}_j\right) = \bar{\mathbf{e}}^p_{\mathcal{L},j,k} + \frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_i} \delta \mathbf{X}_i + \frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_j} \delta \mathbf{X}_j - \tag{4.146}$$

$$\mathbf{l}_{B_i}^k \mathbf{l}_{B_i}^{k\,\mathsf{T}} \left(\bar{\mathbf{e}}^p_{\mathcal{L},j,k} + \frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_i} \delta \mathbf{X}_i + \frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_j} \delta \mathbf{X}_j\right) \tag{4.147}$$

$$= \bar{\mathbf{e}}^p_{\mathcal{L},j,k} - \mathbf{l}_{B_i}^k \mathbf{l}_{B_i}^{k\,\mathsf{T}} \bar{\mathbf{e}}^p_{\mathcal{L},j,k} + \tag{4.148}$$

$$\left(\frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_i} - \mathbf{l}_{B_i}^k \mathbf{l}_{B_i}^{k\,\mathsf{T}} \frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_i}\right) \delta \mathbf{X}_i + \left(\frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_j} - \mathbf{l}_{B_i}^k \mathbf{l}_{B_i}^{k\,\mathsf{T}} \frac{\partial \mathbf{e}^p_{\mathcal{L},j,k}}{\partial \mathbf{X}_j}\right) \delta \mathbf{X}_j \tag{4.149}$$

$$= \bar{\mathbf{e}}^l_{\mathcal{L},j,k} + \frac{\partial \mathbf{e}^l_{\mathcal{L},j,k}}{\partial \mathbf{X}_i} \delta \boldsymbol{\xi}_i + \frac{\partial \mathbf{e}^l_{\mathcal{L},j,k}}{\partial \mathbf{X}_j} \delta \boldsymbol{\xi}_j. \tag{4.150}$$

4.2.3.2 Point-to-plane Factor

Moreover, the LIDAR PCR point-to-plane factor residual is given as

$$e^{pl}_{\mathcal{L},j,k}\left(\mathbf{X}_i, \mathbf{X}_j\right) = \mathbf{n}_{B_i}^{k\,\mathsf{T}} \left(\mathbf{C}_i^\mathsf{T} \left(\mathbf{C}_j \mathbf{r}_{B_j}^{q_k b_j} + \mathbf{r}_j - \mathbf{r}_i\right) - \mathbf{r}_{B_i}^{p_k b_i}\right). \tag{4.151}$$

The analytic expressions for Jacobian matrix of the residual is derived via linearization of (4.151) using a first order Taylor-series expansion. Substituting the errors (4.65) and (4.67) into (4.151) yields

$$e^{pl}_{\mathcal{L},j,k}\left(\mathbf{X}_i, \mathbf{X}_j\right) = {\mathbf{n}^k_{B_i}}^{\mathsf{T}} \left(\left(\bar{\mathbf{C}}_i \delta \mathbf{C}_i\right)^{\mathsf{T}} \left(\bar{\mathbf{C}}_j \delta \mathbf{C}_j \mathbf{r}^{q_k b_j}_{B_j} + \bar{\mathbf{r}}_j + \bar{\mathbf{C}}_j \delta \mathbf{r}_j - \bar{\mathbf{r}}_i - \bar{\mathbf{C}}_i \delta \mathbf{r}_i \right) - \mathbf{r}^{p_k b_i}_{B_i} \right). \tag{4.152}$$

To linearize (4.152), let $\delta \mathbf{C}_j \approx \mathbf{1} + \delta \boldsymbol{\phi}_j^{\times}$. Neglecting the second order terms, (4.152) becomes

$$e^{pl}_{\mathcal{L},j,k}\left(\mathbf{X}_i, \mathbf{X}_j\right) \approx {\mathbf{n}^k_{B_i}}^{\mathsf{T}} \left(\left(\mathbf{1} - \delta \boldsymbol{\phi}_i^{\times}\right) \bar{\mathbf{C}}_i^{\mathsf{T}} \left(\bar{\mathbf{C}}_j \left(\mathbf{1} + \delta \boldsymbol{\phi}_j^{\times}\right) \mathbf{r}^{q_k b_j}_{B_j} + \bar{\mathbf{r}}_j + \bar{\mathbf{C}}_j \delta \mathbf{r}_j - \bar{\mathbf{r}}_i - \bar{\mathbf{C}}_i \delta \mathbf{r}_i \right) - \mathbf{r}^{p_k b_i}_{B_i} \right) \tag{4.153}$$

$$= {\mathbf{n}^k_{B_i}}^{\mathsf{T}} \left(\bar{\mathbf{C}}_i^{\mathsf{T}} \left(\bar{\mathbf{C}}_j \mathbf{r}^{q_k b_j}_{B_j} + \bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i \right) - \mathbf{r}^{p_k b_i}_{B_i} \right) + \tag{4.154}$$

$${\mathbf{n}^k_{B_i}}^{\mathsf{T}} \left(-\delta \boldsymbol{\phi}_i^{\times} \bar{\mathbf{C}}_i^{\mathsf{T}} \left(\bar{\mathbf{C}}_j \mathbf{r}^{q_k b_j}_{B_j} + \bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i \right) + \bar{\mathbf{C}}_i^{\mathsf{T}} \bar{\mathbf{C}}_j \delta \boldsymbol{\phi}_j^{\times} \mathbf{r}^{q_k b_j}_{B_j} + \bar{\mathbf{C}}_i^{\mathsf{T}} \bar{\mathbf{C}}_j \delta \mathbf{r}_j - \delta \mathbf{r}_i \right) \tag{4.155}$$

$$= \bar{e}^{pl}_{\mathcal{L},j,k} + {\mathbf{n}^k_{B_i}}^{\mathsf{T}} \left(\bar{\mathbf{C}}_i^{\mathsf{T}} \left(\bar{\mathbf{C}}_j \mathbf{r}^{q_k b_j}_{B_j} + \bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i \right) \right)^{\times} \delta \boldsymbol{\phi}_i - \tag{4.156}$$

$${\mathbf{n}^k_{B_i}}^{\mathsf{T}} \bar{\mathbf{C}}_i^{\mathsf{T}} \bar{\mathbf{C}}_j {\mathbf{r}^{q_k b_j}_{B_j}}^{\times} \delta \boldsymbol{\phi}_j + {\mathbf{n}^k_{B_i}}^{\mathsf{T}} \bar{\mathbf{C}}_i^{\mathsf{T}} \bar{\mathbf{C}}_j \delta \mathbf{r}_j - {\mathbf{n}^k_{B_i}}^{\mathsf{T}} \delta \mathbf{r}_i. \tag{4.157}$$

$$= \bar{e}^{pl}_{\mathcal{L},j,k} + \frac{\partial e_{\mathcal{L},j,k}}{\partial \mathbf{X}_i} \delta \boldsymbol{\xi}_i + \frac{\partial e_{\mathcal{L},j,k}}{\partial \mathbf{X}_j} \delta \boldsymbol{\xi}_j, \tag{4.158}$$

where

$$\frac{\partial e_{\mathcal{L},j,k}}{\partial \mathbf{X}_i} = \left[{\mathbf{n}^k_{B_i}}^{\mathsf{T}} \left(\bar{\mathbf{C}}_i^{\mathsf{T}} \left(\bar{\mathbf{C}}_j \mathbf{r}^{q_k b_j}_{B_j} + \bar{\mathbf{r}}_j - \bar{\mathbf{r}}_i \right) \right)^{\times} \quad \mathbf{0} \quad -{\mathbf{n}^k_{B_i}}^{\mathsf{T}} \quad \mathbf{0} \quad \mathbf{0} \right], \tag{4.159}$$

$$\frac{\partial e_{\mathcal{L},j,k}}{\partial \mathbf{X}_j} = \left[-{\mathbf{n}^k_{B_i}}^{\mathsf{T}} \bar{\mathbf{C}}_i^{\mathsf{T}} \bar{\mathbf{C}}_j {\mathbf{r}^{q_k b_j}_{B_j}}^{\times} \quad \mathbf{0} \quad {\mathbf{n}^k_{B_i}}^{\mathsf{T}} \bar{\mathbf{C}}_i^{\mathsf{T}} \bar{\mathbf{C}}_j \quad \mathbf{0} \quad \mathbf{0} \right]. \tag{4.160}$$

The point-to-plane error metric has been widely used in structured environment. To reduce computational burden, only the point-to-plane factor is used in the LIVO. However, point-to-line can be still used to increase the robustness of the navigation solution.

4.2.4 Optimization

Finally, using the linearized factors and by setting $\bar{\mathbf{e}} = \mathbf{e}\left(\bar{\mathcal{X}}\right)$, the objective function can be linearized as

$$
\begin{aligned}
J\left(\mathcal{X}\right) &= \tfrac{1}{2}\mathbf{e}_p^{\mathsf{T}}\left(\bar{\mathcal{X}}+\delta\mathcal{X}\right)\mathbf{H}_p\mathbf{e}_p\left(\bar{\mathcal{X}}+\delta\mathcal{X}\right) + \tfrac{1}{2}\mathbf{e}_{\mathcal{I}}^{\mathsf{T}}\left(\bar{\mathcal{X}}+\delta\mathcal{X}\right)\mathbf{\Sigma}_{\mathcal{I}}^{-1}\mathbf{e}_{\mathcal{I}}\left(\bar{\mathcal{X}}+\delta\mathcal{X}\right) + && (4.161)\\
&\quad \tfrac{1}{2}\mathbf{e}_{\mathcal{C}}^{\mathsf{T}}\left(\bar{\mathcal{X}}+\delta\mathcal{X}\right)\mathbf{\Sigma}_{\mathcal{C}}^{-1}\mathbf{e}_{\mathcal{C}}\left(\bar{\mathcal{X}}+\delta\mathcal{X}\right) + \tfrac{1}{2}\mathbf{e}_{\mathcal{L}}^{\mathsf{T}}\left(\bar{\mathcal{X}}+\delta\mathcal{X}\right)\mathbf{\Sigma}_{\mathcal{L}}^{-1}\mathbf{e}_{\mathcal{L}}\left(\bar{\mathcal{X}}+\delta\mathcal{X}\right) && (4.162)\\
&= \tfrac{1}{2}\mathbf{e}_p^{\mathsf{T}}\left(\bar{\mathcal{X}}+\delta\mathcal{X}\right)\mathbf{H}_p\mathbf{e}_p\left(\bar{\mathcal{X}}+\delta\mathcal{X}\right) + \tfrac{1}{2}\left(\bar{\mathbf{e}}_{\mathcal{I}} + \frac{\partial \mathbf{e}_{\mathcal{I}}}{\partial \mathcal{X}}\delta\mathbf{X}\right)^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{I}}^{-1}\left(\bar{\mathbf{e}}_{\mathcal{I}} + \frac{\partial \mathbf{e}_{\mathcal{I}}}{\partial \mathcal{X}}\delta\mathbf{X}\right) + && (4.163)\\
&\quad \tfrac{1}{2}\left(\bar{\mathbf{e}}_{\mathcal{C}} + \frac{\partial \mathbf{e}_{\mathcal{C}}}{\partial \mathcal{X}}\delta\mathbf{X}\right)^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{C}}^{-1}\left(\bar{\mathbf{e}}_{\mathcal{C}} + \frac{\partial \mathbf{e}_{\mathcal{C}}}{\partial \mathcal{X}}\delta\mathbf{X}\right) + \tfrac{1}{2}\left(\bar{\mathbf{e}}_{\mathcal{L}} + \frac{\partial \mathbf{e}_{\mathcal{L}}}{\partial \mathcal{X}}\delta\mathbf{X}\right)^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{L}}^{-1}\left(\bar{\mathbf{e}}_{\mathcal{L}} + \frac{\partial \mathbf{e}_{\mathcal{L}}}{\partial \mathcal{X}}\delta\mathbf{X}\right) && (4.164)\\
&= \left(\tfrac{1}{2}\bar{\mathbf{e}}_p^{\mathsf{T}}\mathbf{H}_p\bar{\mathbf{e}}_p + \mathbf{r}_p^{\mathsf{T}}\delta\mathbf{X} + \tfrac{1}{2}\delta\mathbf{X}^{\mathsf{T}}\mathbf{H}_p\delta\mathbf{X}\right) + && (4.165)\\
&\quad \left(\tfrac{1}{2}\bar{\mathbf{e}}_{\mathcal{I}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{I}}^{-1}\bar{\mathbf{e}}_{\mathcal{I}} + \bar{\mathbf{e}}_{\mathcal{I}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{I}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{I}}}{\partial \mathcal{X}}\delta\mathbf{X} + \tfrac{1}{2}\delta\mathbf{X}^{\mathsf{T}}\frac{\partial \mathbf{e}_{\mathcal{I}}}{\partial \mathcal{X}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{I}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{I}}}{\partial \mathcal{X}}\delta\mathbf{X}\right) + && (4.166)\\
&\quad \left(\tfrac{1}{2}\bar{\mathbf{e}}_{\mathcal{C}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{C}}^{-1}\bar{\mathbf{e}}_{\mathcal{I}} + \bar{\mathbf{e}}_{\mathcal{C}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{C}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{C}}}{\partial \mathcal{X}}\delta\mathbf{X} + \tfrac{1}{2}\delta\mathbf{X}^{\mathsf{T}}\frac{\partial \mathbf{e}_{\mathcal{C}}}{\partial \mathcal{X}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{C}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{C}}}{\partial \mathcal{X}}\delta\mathbf{X}\right) + && (4.167)\\
&\quad \left(\tfrac{1}{2}\bar{\mathbf{e}}_{\mathcal{L}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{L}}^{-1}\bar{\mathbf{e}}_{\mathcal{L}} + \bar{\mathbf{e}}_{\mathcal{L}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{L}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{L}}}{\partial \mathcal{X}}\delta\mathbf{X} + \tfrac{1}{2}\delta\mathbf{X}^{\mathsf{T}}\frac{\partial \mathbf{e}_{\mathcal{L}}}{\partial \mathcal{X}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{L}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{L}}}{\partial \mathcal{X}}\delta\mathbf{X}\right) && (4.168)\\
&= J\left(\bar{\mathcal{X}}\right) + \left(\mathbf{r}_p^{\mathsf{T}} + \bar{\mathbf{e}}_{\mathcal{I}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{I}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{I}}}{\partial \mathcal{X}} + \bar{\mathbf{e}}_{\mathcal{C}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{C}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{C}}}{\partial \mathbf{X}} + \bar{\mathbf{e}}_{\mathcal{L}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{L}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{L}}}{\partial \mathbf{X}}\right)\delta\mathbf{X} + && (4.169)\\
&\quad \tfrac{1}{2}\delta\mathbf{X}^{\mathsf{T}}\left(\mathbf{H}_p + \frac{\partial \mathbf{e}_{\mathcal{I}}}{\partial \mathcal{X}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{I}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{I}}}{\partial \mathcal{X}} + \frac{\partial \mathbf{e}_{\mathcal{C}}}{\partial \mathcal{X}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{C}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{C}}}{\partial \mathcal{X}} + \frac{\partial \mathbf{e}_{\mathcal{L}}}{\partial \mathcal{X}}^{\mathsf{T}}\mathbf{\Sigma}_{\mathcal{L}}^{-1}\frac{\partial \mathbf{e}_{\mathcal{L}}}{\partial \mathcal{X}}\right)\delta\mathbf{X} && (4.170)\\
&= J\left(\mathcal{X}\right) + \frac{\partial J\left(\mathcal{X}\right)}{\partial \mathcal{X}}\delta\mathbf{X} + \tfrac{1}{2}\delta\mathbf{X}^{\mathsf{T}}\frac{\partial^2 J\left(\mathcal{X}\right)}{\partial \mathcal{X}^2}\delta\mathbf{X}, && (4.171)
\end{aligned}
$$

where $\delta\mathbf{X} = \begin{bmatrix}\delta\boldsymbol{\xi}_k^{\mathsf{T}} & \delta\boldsymbol{\xi}_{k+1}^{\mathsf{T}} & \ldots & \delta\boldsymbol{\xi}_{k+N}^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}}$. Given that the approximate Hessian matrix, $\frac{\partial^2 J(\mathcal{X})}{\partial \mathcal{X}^2}$, is positive definite, Gauss-Newton method solves iteratively by solving for $\delta\mathbf{X}$ that minimizes the cost

$$
\begin{aligned}
\frac{\partial J\left(\mathcal{X}\right)}{\partial \delta\mathcal{X}} = \frac{\partial^2 J\left(\mathcal{X}\right)}{\partial \mathcal{X}^2}\delta\mathbf{X}^* + \frac{\partial J\left(\mathcal{X}\right)}{\partial \mathcal{X}} &= 0. && (4.172)\\
\rightarrow \delta\mathbf{X}^* &= -\frac{\partial^2 J\left(\mathcal{X}\right)}{\partial \mathcal{X}^2}\frac{\partial J\left(\mathcal{X}\right)}{\partial \mathcal{X}}, && (4.173)
\end{aligned}
$$

and updating the nominal solution using the error definitions defined in Section 4.2

$$
\hat{\mathbf{X}}_t = \bar{\mathbf{X}}_t\delta\mathbf{X}_t, \qquad (4.174)
$$

for $t \in [k, N]$.

4.3 Marginalization

As the robot continuously moves and observes new features, the size of the state $\mathcal{X}$ in the window constantly increases. To bound the computational complexity, older measurement factors get discarded from the sliding window and get integrated into a prior. This process is called marginalization [37]. Marginalization is necessary not only to bound the computational complexity but also to carry over the uncertainty associated with the prior state as a covariance $\mathbf{P}_0$. Consistency of an estimator is crucial as it determines how reliable the estimator is despite the accuracy estimated trajectory. The marginalization is carried out using the Schur complement [38]. As a result, a new prior can be integrated into the existing prior.

Consider a set of states in a window

$$\mathcal{X} = \{\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_k, \lambda_0, \lambda_1, \dots, \lambda_n\}. \tag{4.175}$$

Then the set of states can be divided into m_s, and m_l. The index for marginalizing states and marginalizing landmarks are denoted as m_s and m_l respectively, and the index for remaining states and remaining landmarks are denoted as $k - m_s$ and $n - m_l$. Let

$$\mathcal{X}_m = \{\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_{m_s}, \lambda_0, \lambda_1, \dots, \lambda_{m_l}\}, \tag{4.176}$$

$$\mathcal{X}_r = \{\mathbf{X}_{m_s+1}, \mathbf{X}_{m_s+2}, \dots, \mathbf{X}_k, \lambda_{m_l+1}, \lambda_{m_l+2}, \dots, \lambda_n\}. \tag{4.177}$$

The robot collects more measurements between time $t = k$ and $t = k'$. Thus, new robot and landmarks states $\mathcal{X}_n$ are augmented. Without marginalization, the augmented state $\mathcal{X}_{k'}$ can be solved by minimizing the cost

$$\underset{\mathcal{X}_{k'}}{\arg\min}\; J\left(\mathcal{X}_{k'}\right). \tag{4.178}$$

Given that $\mathcal{X}_{k'} = \{\mathcal{X}_m, \mathcal{X}_r, \mathcal{X}_n\}$, the cost function can be rewritten as

$$\underset{\mathcal{X}_m,\mathcal{X}_r,\mathcal{X}_n}{\arg\min}\; J\left(\mathcal{X}_m, \mathcal{X}_r, \mathcal{X}_n\right) = \underset{\mathcal{X}_r,\mathcal{X}_n}{\arg\min}\left(\underset{\mathcal{X}_m}{\arg\min}\; J\left(\mathcal{X}_m, \mathcal{X}_r, \mathcal{X}_n\right)\right) \tag{4.179}$$

$$= \underset{\mathcal{X}_r,\mathcal{X}_n}{\arg\min}\left(J_n\left(\mathcal{X}_r, \mathcal{X}_n\right) + \underset{\mathcal{X}_m}{\arg\min}\; J_m\left(\mathcal{X}_m, \mathcal{X}_r\right)\right). \tag{4.180}$$

Now, solving for $\arg\min\ J_n(\mathcal{X}_r, \mathcal{X}_n)$ is exactly as shown in (4.70) without marginalization factor. To obtain the marginalization factor the marginal cost $J_m(\mathcal{X}_m, \mathcal{X}_r)$ is minimized. Since the measurement and process models are nonlinear, Taylor-series approximation is employed on the marginal cost function

$$J_m(\mathcal{X}_m, \mathcal{X}_r) \approx J_m\left(\bar{\mathcal{X}}_m, \bar{\mathcal{X}}_r\right) + \delta\mathbf{X}^\mathsf{T} \frac{\partial J_m(\mathcal{X}_m, \mathcal{X}_r)}{\partial \mathcal{X}} + \tfrac{1}{2}\delta\mathbf{X}^\mathsf{T} \frac{\partial^2 J_m(\mathcal{X}_m, \mathcal{X}_r)}{\partial \mathcal{X}^2}\delta\mathbf{X} \tag{4.181}$$

$$= J_m\left(\bar{\mathcal{X}}_m, \bar{\mathcal{X}}_r\right) + \begin{bmatrix}\delta\mathbf{X}_m^\mathsf{T} & \delta\mathbf{X}_r^\mathsf{T}\end{bmatrix} \begin{bmatrix}\frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_m} \\ \frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_r}\end{bmatrix} + \tag{4.182}$$

$$\tfrac{1}{2}\begin{bmatrix}\delta\mathbf{X}_m^\mathsf{T} & \delta\mathbf{X}_r^\mathsf{T}\end{bmatrix} \begin{bmatrix}\frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_m}^\mathsf{T} \\ \frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_r}^\mathsf{T}\end{bmatrix} \begin{bmatrix}\frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_m} & \frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_r}\end{bmatrix} \begin{bmatrix}\delta\mathbf{X}_m \\ \delta\mathbf{X}_r\end{bmatrix}, \tag{4.183}$$

where

$$\delta\mathbf{X} = \begin{bmatrix}\delta\mathbf{X}_m \\ \delta\mathbf{X}_r\end{bmatrix} = \left[\begin{array}{c} \operatorname{Log}\left(\mathbf{X}_0^{-1}\mathbf{X}_0\right) \\ \operatorname{Log}\left(\mathbf{X}_1^{-1}\mathbf{X}_1\right) \\ \vdots \\ \operatorname{Log}\left(\mathbf{X}_{m_s}^{-1}\mathbf{X}_{m_s}\right) \\ \delta\lambda_0 \\ \delta\lambda_1 \\ \vdots \\ \delta\lambda_{m_l} \\ \hline \operatorname{Log}\left(\mathbf{X}_{m_s+1}^{-1}\mathbf{X}_{m_s+1}\right) \\ \operatorname{Log}\left(\mathbf{X}_{m_s+2}^{-1}\mathbf{X}_{m_s+2}\right) \\ \vdots \\ \operatorname{Log}\left(\mathbf{X}_{m_k}^{-1}\mathbf{X}_{m_k}\right) \\ \delta\lambda_{m_l+1} \\ \delta\lambda_{m_l+2} \\ \vdots \\ \delta\lambda_n \end{array}\right]. \tag{4.184}$$

The Jacobian and Hessian matrices of J_m are obtained by the linearization of the cost function $J\left(\mathcal{X}_k\right)$. For brevity of notation, let

$$\mathbf{b}_m = \begin{bmatrix} \frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_m} \\ \frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{mm} \\ \mathbf{b}_{mr} \end{bmatrix}, \tag{4.185}$$

$$\mathbf{A}_m = \begin{bmatrix} \frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_m}^{\mathsf{T}} \\ \frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_r}^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} \frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_m} & \frac{\partial J_m(\mathcal{X}_m,\mathcal{X}_r)}{\partial \mathcal{X}_r} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{mm} & \mathbf{A}_{mr} \\ \mathbf{A}_{rm} & \mathbf{A}_{rr} \end{bmatrix}. \tag{4.186}$$

The solution for $\mathcal{X}_m$ is obtained by finding $\begin{bmatrix} \delta\mathbf{X}_m^{\mathsf{T}} & \delta\mathbf{X}_r^{\mathsf{T}} \end{bmatrix}$ that minimizes $J_m\left(\mathcal{X}_m, \mathcal{X}_r\right)$

$$\begin{bmatrix} \mathbf{A}_{mm} & \mathbf{A}_{mr} \\ \mathbf{A}_{rm} & \mathbf{A}_{rr} \end{bmatrix} \begin{bmatrix} \delta\mathbf{X}_m \\ \delta\mathbf{X}_r \end{bmatrix} = - \begin{bmatrix} \mathbf{b}_{mm} \\ \mathbf{b}_{mr} \end{bmatrix} \tag{4.187}$$

$$\begin{bmatrix} \mathbf{A}_{mm}\delta\mathbf{X}_m + \mathbf{A}_{mr}\delta\mathbf{X}_r \\ \mathbf{A}_{rm}\delta\mathbf{X}_m + \mathbf{A}_{rr}\delta\mathbf{X}_m \end{bmatrix} = - \begin{bmatrix} \mathbf{b}_{mm} \\ \mathbf{b}_{mr} \end{bmatrix}. \tag{4.188}$$

Rearranging the above equation yields

$$\delta\mathbf{X}_m = -\mathbf{A}_{mm}^{-1}\left(\mathbf{b}_{mm} + \mathbf{A}_{mr}\delta\mathbf{X}_r\right). \tag{4.189}$$

Using (4.189), the cost function can be parametrized such that $J_m\left(\mathcal{X}_m, \mathcal{X}_r\right) = \tilde{J}_m\left(\mathcal{X}_r\right)$, and

$$\tilde{J}_m\left(\mathcal{X}_r\right) \approx \tilde{J}_m\left(\bar{\mathcal{X}}_r\right) + \mathbf{b}_p^{\mathsf{T}}\delta\mathbf{X}_r + \tfrac{1}{2}\delta\mathbf{X}_r^{\mathsf{T}}\mathbf{A}_p\delta\mathbf{X}_r, \tag{4.190}$$

where $\mathbf{b}_p$ and $\mathbf{A}_p$ can be obtained via schur complement

$$\mathbf{b}_p = \mathbf{b}_{mr} - \mathbf{A}_{rm}\mathbf{A}_{mm}^{-1}\mathbf{b}_{mm}, \tag{4.191}$$

$$\mathbf{A}_p = \mathbf{A}_{rr} - \mathbf{A}_{rm}\mathbf{A}_{mm}^{-1}\mathbf{A}_{mr}. \tag{4.192}$$

Detailed derivation is given in Appendix A. Substituting (4.190) into (4.178) yields

$$\underset{\mathcal{X}_r,\mathcal{X}_n}{\arg\min}\; J = \underset{\mathcal{X}_r,\mathcal{X}_n}{\arg\min}\; J_n\left(\mathcal{X}_r, \mathcal{X}_n\right) + \tilde{J}_m\left(\bar{\mathcal{X}}_r\right) + \mathbf{b}_p^{\mathsf{T}}\delta\mathbf{X}_r + \tfrac{1}{2}\delta\mathbf{X}_r^{\mathsf{T}}\mathbf{A}_p\delta\mathbf{X}_r. \tag{4.193}$$

Solving for (4.193) using Gauss-Newton method results in

$$\frac{\partial J}{\partial \delta\mathcal{X}} = \frac{\partial J_n\left(\mathcal{X}_r, \mathcal{X}_n\right)}{\partial \delta\mathcal{X}} + \mathbf{b}_p + \mathbf{A}_p\delta\mathbf{X}_r, \tag{4.194}$$

where $\frac{\partial J_n(\mathcal{X}_r,\mathcal{X}_n)}{\partial \delta\mathcal{X}}$ is the partial derivative associated with the active measurement factors in the window. Finally, the marginalization information $\mathbf{b}_p$ and $\mathbf{A}_p$ are augmented to a matrix by

$$\mathbf{r}_p = \mathbf{\Pi}_r \mathbf{b}_p, \tag{4.195}$$

$$\mathbf{H}_p = \mathbf{\Pi}_r \mathbf{A}_p, \tag{4.196}$$

where $\mathbf{\Pi}_r = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \dots \end{bmatrix}$.

4.3.1 Visual Feature Management

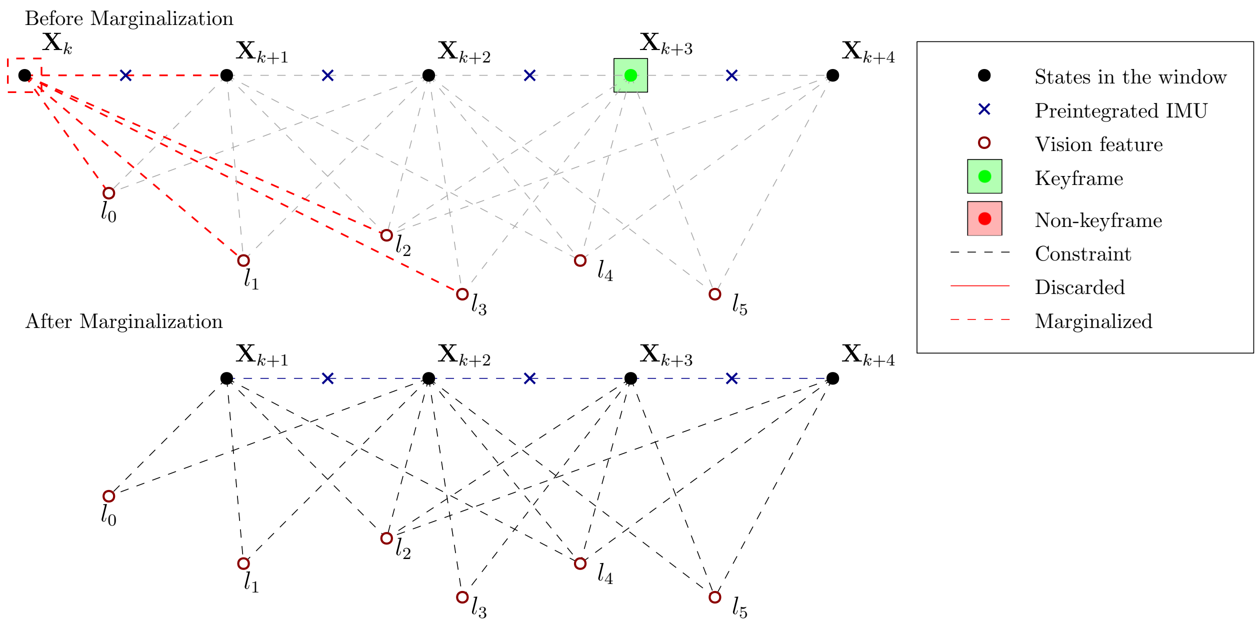
Visual features are regularly removed from the window alongside the marginalization. The feature management scheme herein is based on [1]. The feature manager is responsible for organizing a feature list and an observation list. A feature list contains the feature points tracked in the current window, and each feature contains a list of observations that marks the frames where it has been tracked. When a new image measurement arrives, it is tagged as either a *keyframe* or a *non-keyframe*. New features detected in the new image are added to the feature list, and existing features that have been tracked in the new image are added to the observation list.

KLT tracker needs features to be updated as the features detected in the initial image will no longer be detected in the new image in different angle of view. When the features detected in the initial image is tracked below threshold, it means that there is not enough features to track. As a result, features are detected at every image frame and a keyframe is selected to update the features for tracking. A keyframe is selected using two criteria that include average parallax apart from the previous keyframe and tracking quality. A frame is considered a keyframe if the average parallax of all features between the current frame and the latest keyframe exceeds some threshold. Also, a frame is automatically considered a keyframe if it tracks less than 20 previously-observed features.

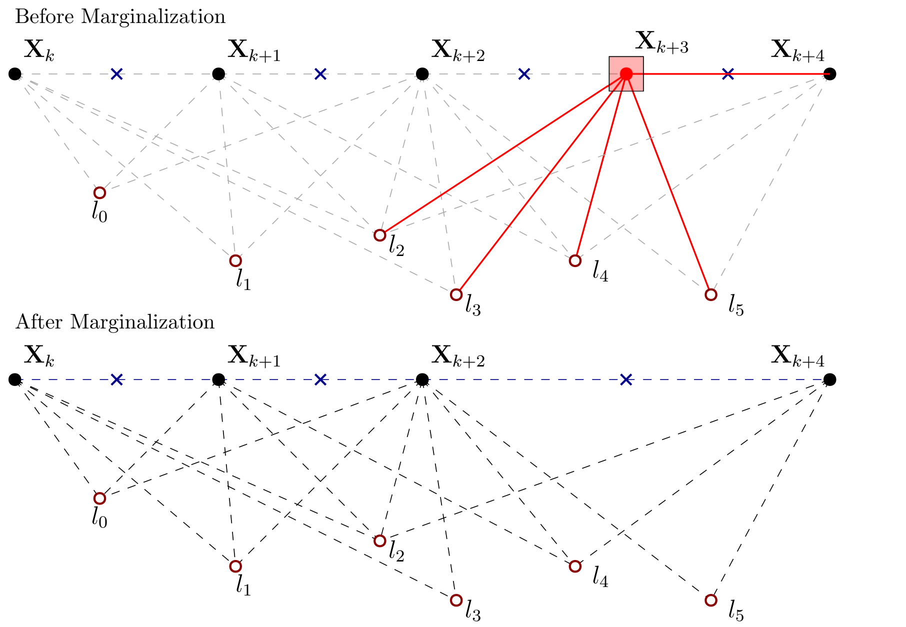
When a new image is acquired, if the second latest frame is a keyframe, the current frame is kept in the window, and the oldest frame is marginalized. Marginalized measurements are stored in a prior factor. Otherwise, if the second latest frame is a non-keyframe, the frame as well as all the observations are discarded from the window without marginalization.

4.3.2 LIDAR Local Map Management

Similar to visual features that are regularly discarded from the window to bound the computational complexity, LIDAR sweeps are discarded to reduce the size of the map. The local map manager is responsible of managing the size of sweeps stored in cache and building the local map out of sweeps in memory. When a new sweep comes in alongside with the camera

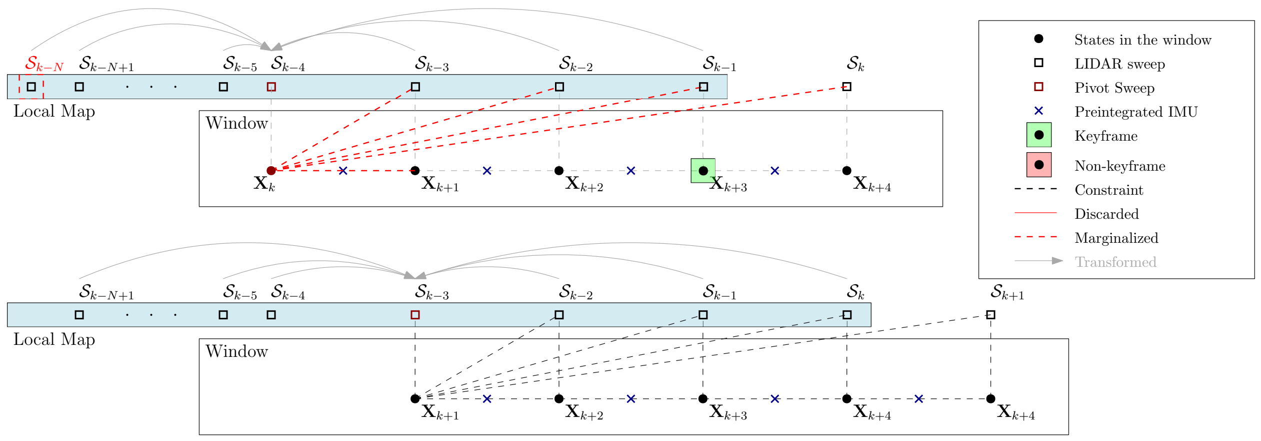


(a) Second latest frame is a keyframe.



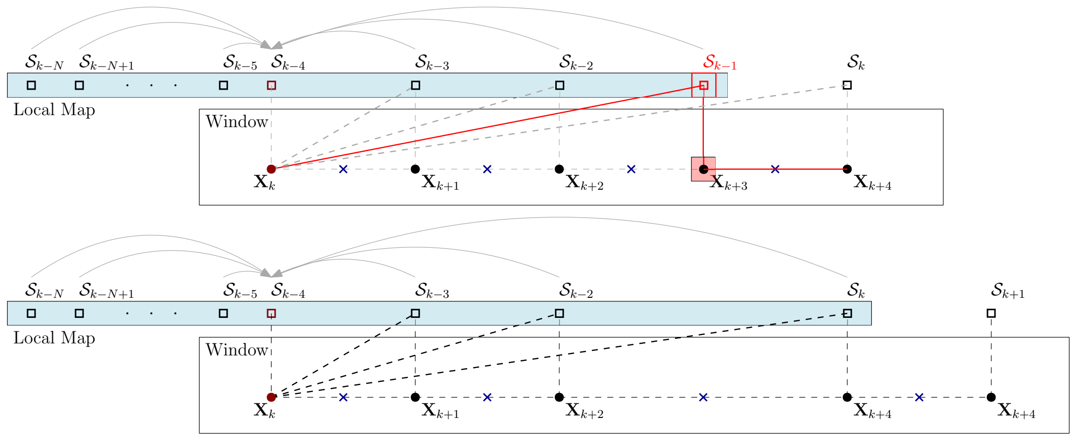
(b) Second latest frame is a non-keyframe.

Figure 4.2: Illustration of visual feature management.



$\mathcal{S}_{k-N}$ $\mathcal{S}_{k-N+1}$ $\mathcal{S}_{k-5}$ $\mathcal{S}_{k-4}$ $\mathcal{S}_{k-3}$ $\mathcal{S}_{k-2}$ $\mathcal{S}_{k-1}$ $\mathcal{S}_{k}$ $\mathcal{S}_{k+1}$
Local Map
Window
$\mathbf{X}_k$ $\mathbf{X}_{k+1}$ $\mathbf{X}_{k+2}$ $\mathbf{X}_{k+3}$ $\mathbf{X}_{k+4}$
States in the window
LIDAR sweep
Pivot Sweep
Preintegrated IMU
Keyframe
Non-keyframe
Constraint
Discarded
Marginalized
Transformed

(a) Second latest frame is a keyframe.



$\mathcal{S}_{k-N}$ $\mathcal{S}_{k-N+1}$ $\mathcal{S}_{k-5}$ $\mathcal{S}_{k-4}$ $\mathcal{S}_{k-3}$ $\mathcal{S}_{k-2}$ $\mathcal{S}_{k-1}$ $\mathcal{S}_{k}$ $\mathcal{S}_{k+1}$
Local Map
Window
$\mathbf{X}_k$ $\mathbf{X}_{k+1}$ $\mathbf{X}_{k+2}$ $\mathbf{X}_{k+3}$ $\mathbf{X}_{k+4}$

(b) Second latest frame is a non-keyframe.

Figure 4.3: Illustration of LIDAR sweep management.

measurement, if the second latest frame is a keyframe, the oldest sweep gets discarded. Otherwise, if the second latest frame is a non-keyframe, the corresponding sweep gets thrown away.

N number of sweeps are stored in cache and used to build a map. A larger N means denser map to be matched and yet higher computation power is required. Smaller N may not have enough points to find accurate point correspondence. The size of N can be chosen by a user.

A local map is built by transforming sweeps in the cache and resolving them in a *pivot* frame using $\hat{\mathbf{T}} \in SE(3)$. A pivot frame is usually chosen such that it is a frame where reliable robot state is resolved in. Consider a list of sweeps

$$\mathcal{M} = \{\mathcal{S}_{k-N}, \ldots, \mathcal{S}_{k-W}, \ldots, \mathcal{S}_{k-1}\}, \tag{4.197}$$

and the window size $W < N$. The most recent sweep acquired $\mathcal{S}_k$ is not used to form local map since the corresponding robot state is yet to be optimized. Thus, the local map is built using the Algorithm 1.

Algorithm 1

```
1: Set pivot frame $\mathbf{T}_p \leftarrow \hat{\mathbf{T}}_{WB_p}$
2: for $\mathcal{S}_j \in \mathcal{M}$ do
3:     $\mathbf{L} \leftarrow \mathbf{T}_p^{-1}\hat{\mathbf{T}}_{WB_j}\mathcal{S}_j$
4: end for
5: VoxelFilter($\mathbf{L}$)
```

4.4 LIVOM Results and Discussions

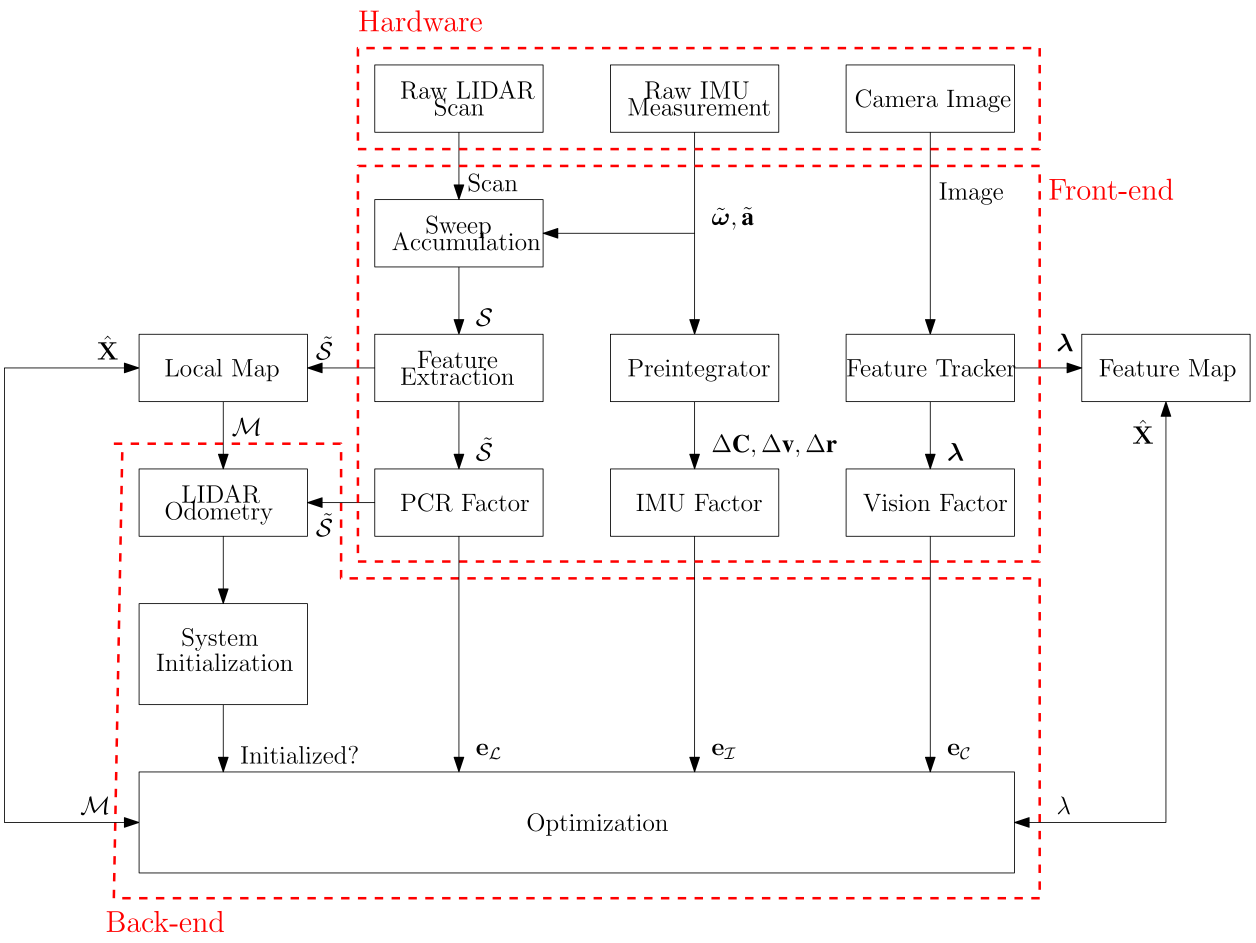


Figure 4.4: Block diagram illustrating the full pipeline of the proposed LIVO.

The full pipline of the proposed LIVO in Figure 4.4 is tested in an experimental setting. Experimental testing is necessary to test the algorithm using real sensor data difficult to model in simulation setting.

4.4.1 Initialization

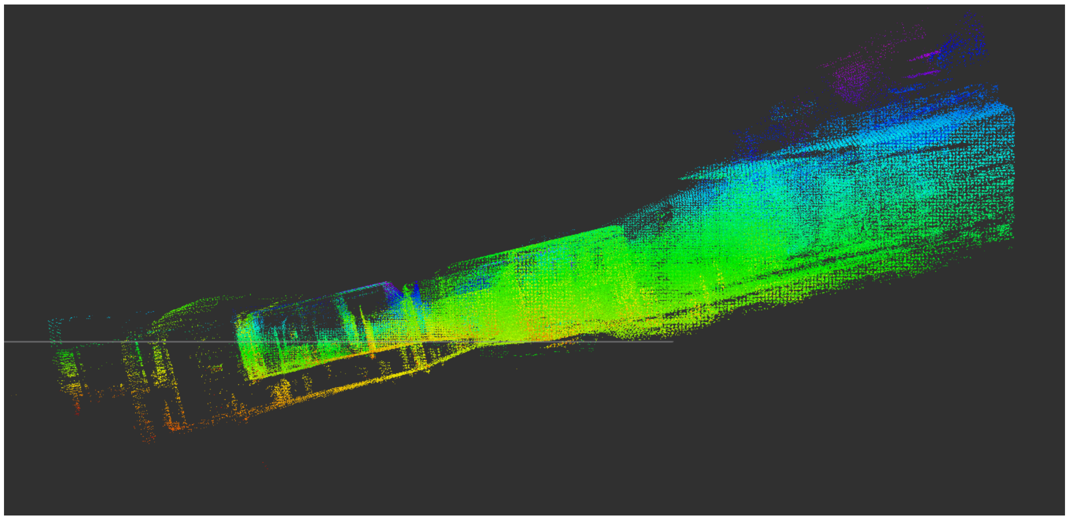
The initialization scheme is tested on the 3D Cartographer - Deutsches Museum dataset [39]. This data was collected using a 3D LIDAR backpact at the Deutsches Museum with an IMU and two Velodyne VLP-16 LIDARs. The result of LIO, shown in Figure 4.5, demonstrates the 3D reconstruction with and without system initialization. The frame associated with the sensor suite is initially not aligned with the gravity vector as shown in Figure 4.5a, and after the initialization the local frame that the map is resolved in is aligned with gravity.

4.4.2 Experimental Results

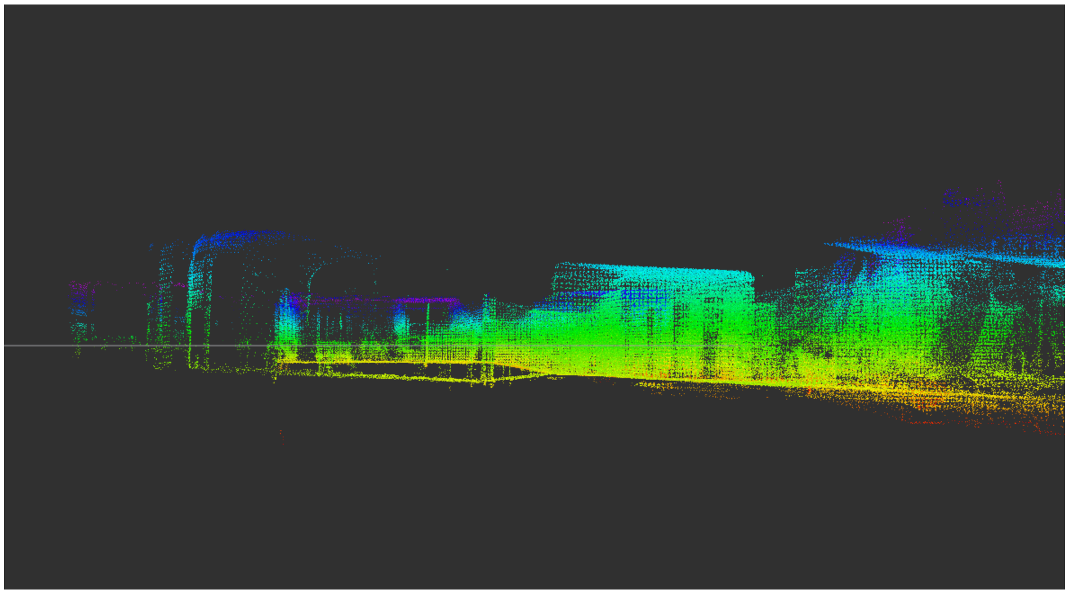
Experimental results comprises of running the LIDAR-Inertial, Visual-Inerital, and the proposed solution to test the robustness of the algorithm. A Velodyne Puck VLP-16, FLIR Camera and MEMS IMU are mounted on a sensor head as shown in Figure 4.6a. For indoor navigation, the sensor head is carried around inside a room while collecting data. To collect the ground truth data to compare with the proposed navigation solution, an Optitrack motion capture system is used. Several Optitrack cameras are installed in the room to accurately track spherical objects mounted on the sensor head, providing the poses of sensor head. Optitrack provides the robot poses resolved in some reference frame. On the other hand, the odometry solution provides the state estimates in its own local frame. Thus, to overcome this issue, Evo trajectory package [40] is used for post processing and evaluation of the odometry.

Solution	RMSE	Std	Max	Min
VIO	0.710595	0.130835	1.04826	0.40776
LIO	0.479786	0.0494921	0.718802	**0.361876**
LIVO	**0.465473**	**0.0304656**	**0.536912**	0.375523

Table 4.1: Summary of odometry results on McGill Lab 1 dataset. RMSE is computed in $SE(3)$ as only the robot poses are available as ground truth.



(a) A 3D Reconstruction of Deutsches Museum without initialization.



(b) A 3D Reconstruction of Deutsches Museum with initialization.

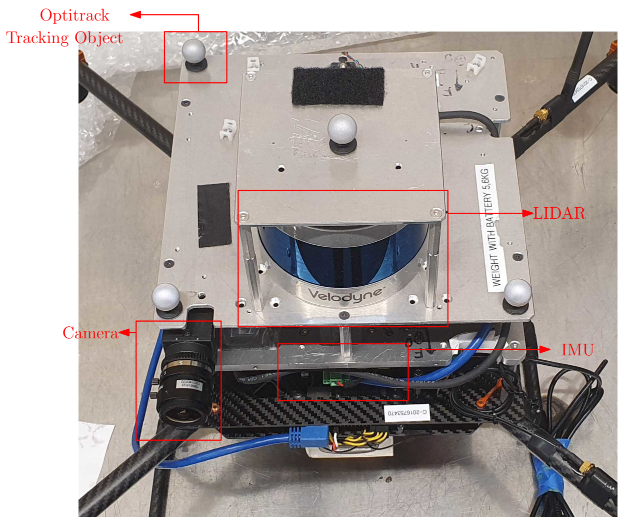
Figure 4.5: Initialization result shown in a 3D reconstruction of Deutsches Museum data.

Solution	RMSE	Std	Max	Min
LIO	2.8321	0.00476637	2.85698	**2.81935**
LIVO	**2.82975**	**0.00142881**	**2.83694**	2.82774

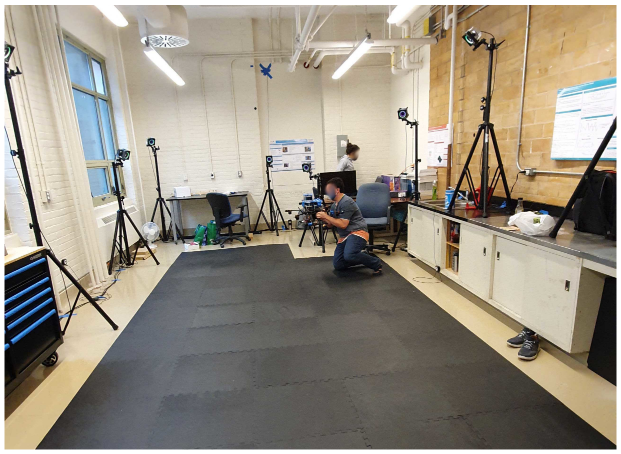
Table 4.2: Summary of odometry results on McGill Lab 2 dataset. VIO fails to navigate.

Two sets of dataset were taken. The first data, McGill Lab 1, consists of slower motion with static initialization and the second data, McGill Lab 2, consists of faster motion with aggressive initialization. Figure 4.7 demonstrates the 3D reconstruction of the lab space and the estimated trajectory result with LIVO. All three odometry solutions are able to perform on the easy dataset with some disparity in accuracy. Notice that the LIO and LIVO outperforms the vision-based solution although the absolute pose error between LIO and LIVO are not significant. One can assume that visual odometry suffers from tracking features due to lack of features in the environment or high illumination inside the room. For McGill Lab 2 dataset, visual-inertial odometry system fails to track features due to fast motion and fails in navigation. On the other hand, laser-based odometry is able to perform just as well as the former dataset. Overall, the proposed solution, LIVO, outperforms both odometry solutions for both datasets by its accuracy as shown in Table 4.1. Further, Figure 4.8 shows that the standard deviation associated with the LIVO results is much smaller than that of LIO. Such comparison is more distinguishable in Figure 4.9. This proves in higher confidence in LIVO results than LIO. LIVO achieves better accuracy and consistency by leveraging the strengths of laser-based sensors in the case when vision fails.

Further, for the outdoor navigation, the sensor head is mounted on a car while the car drives around Saint-Henri neighborhood of Montreal. An accurate point cloud map of the city block is demonstrated in Figure 4.10 with the estimated trajectory shown in green line. The resulting map is compared with the Google Earth image that was reconstructed using bundle adjustment of aerial images.



(a) Representation of sensor configuration.



(b) Optitrack setting for ground truth data collection.

Figure 4.6: Data collection with Optitrack motion capture system.

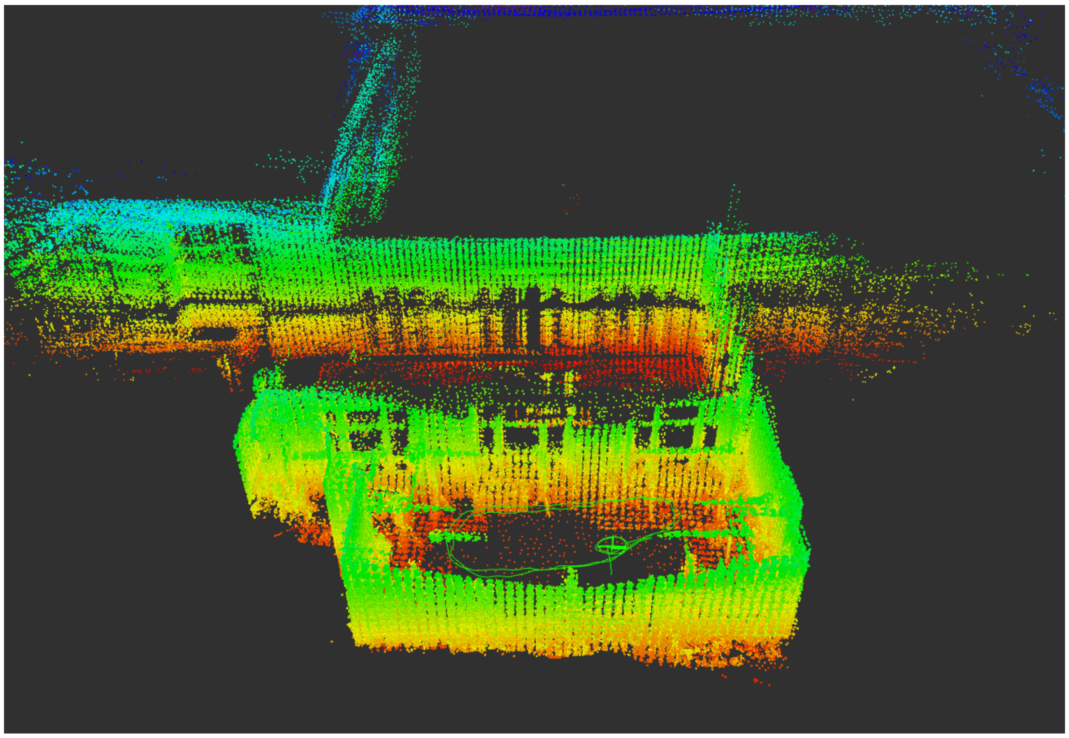
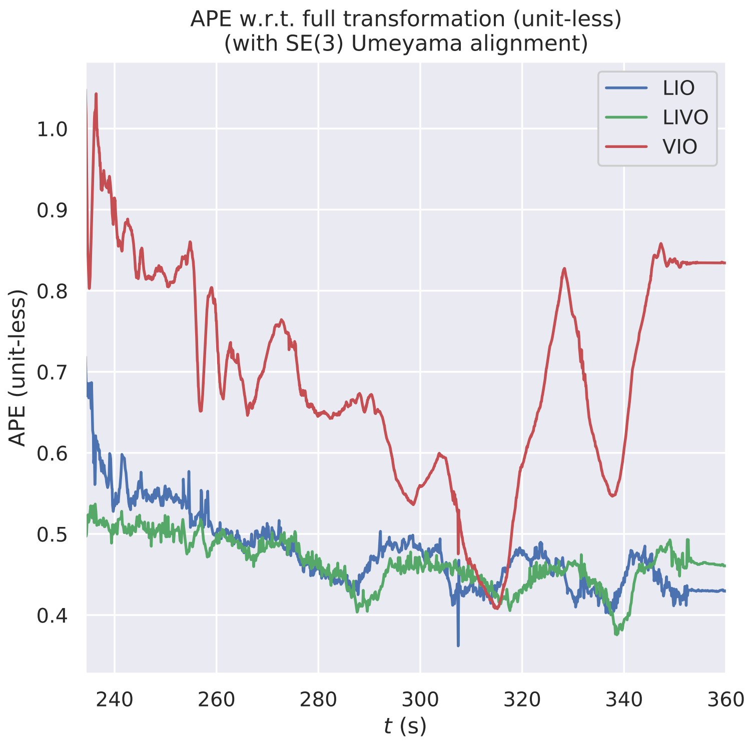
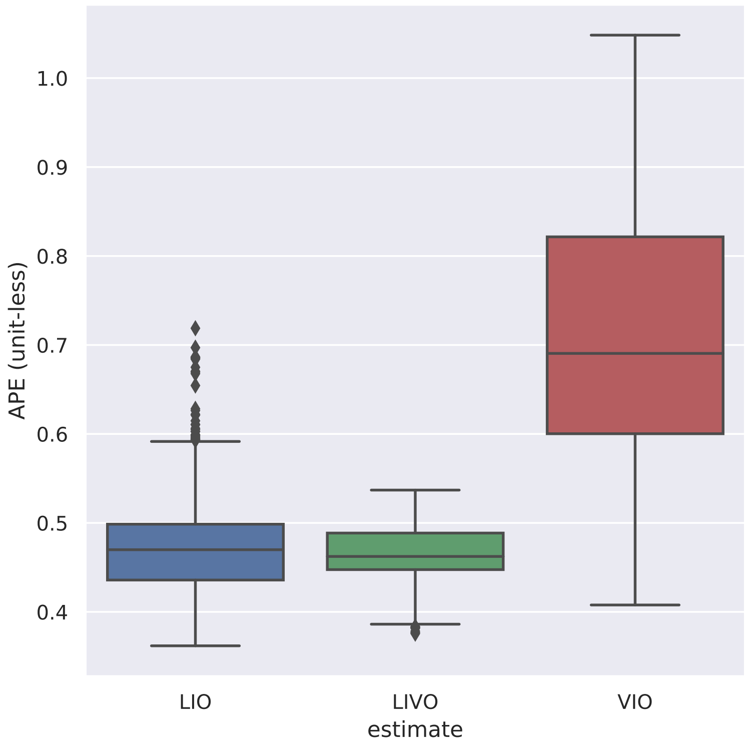


Figure 4.7: A 3D reconstruction of the lab space with the estimated trajectory in green.

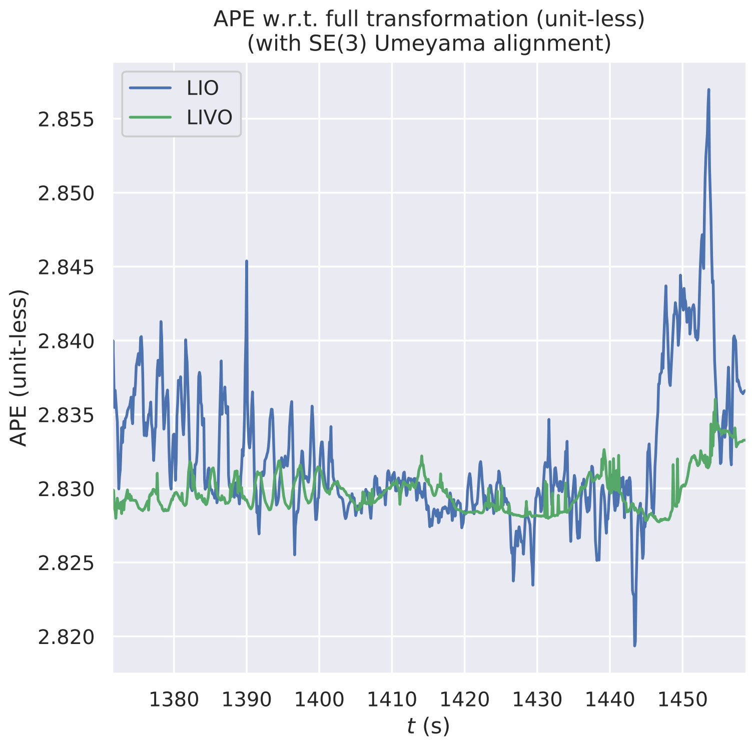


(a) Absolute pose error.

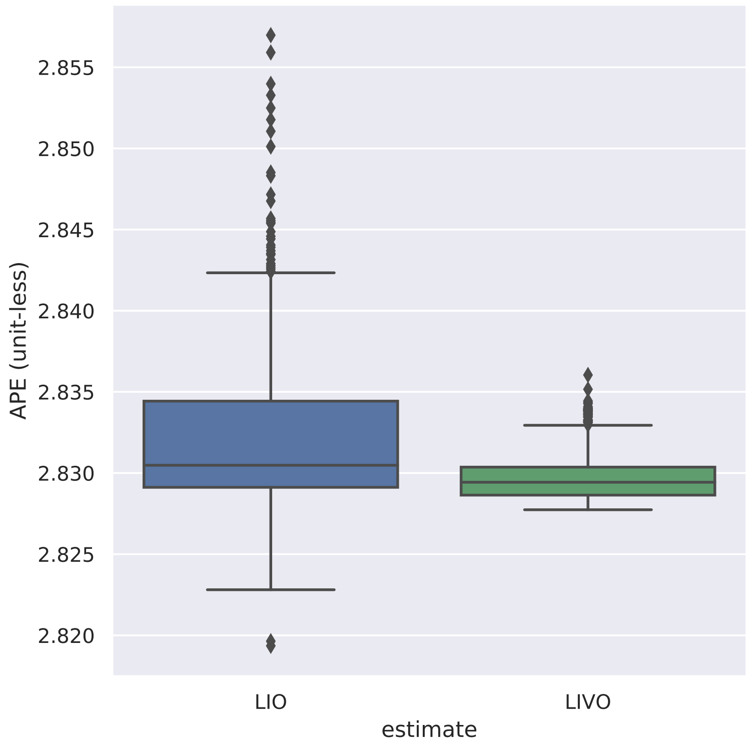


(b) Absolute pose error in box plot.

Figure 4.8: Results of odometry solutions on McGill Lab 1 dataset.

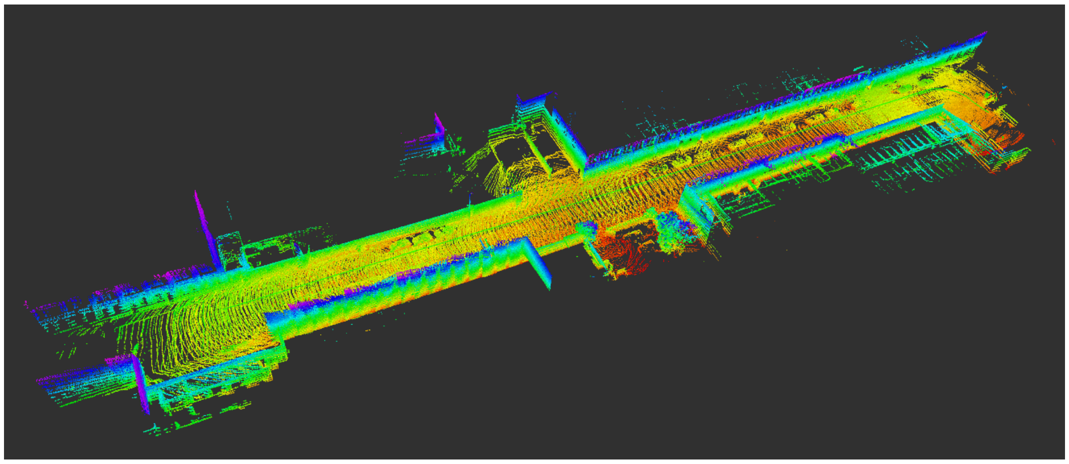


(a) Absolute pose error.



(b) Absolute pose error in box plot.

Figure 4.9: Results of odometry solutions on McGill Lab 2 dataset.


(a) Isometric view.

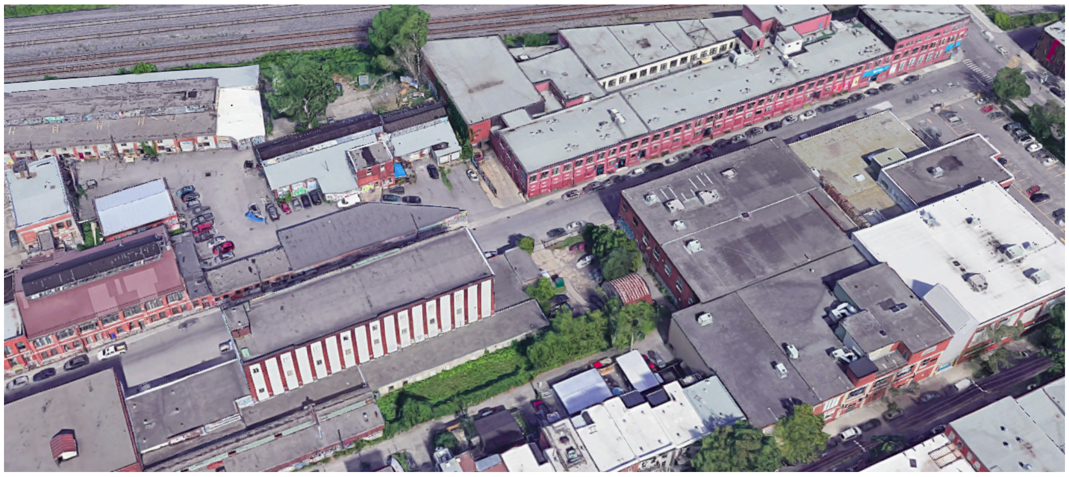

(b) Google Earth view.

Figure 4.10: Outdoor navigation using LIVO.

Chapter 5

Closing Remarks and Future Work

5.1 Conclusions

This thesis has contributed in developing a novel tightly-coupled LIDAR-Inertial-Visual Odometry and Mapping solution in sliding window framework for robot navigation. As an extension to [1] and [2], the depth-based vision factor and point cloud registration factor are derived using matrix Lie group theory. This proposed solution is shown to attain greater accuracy and robustness in experimental trials.

As mentioned in the introduction, most SLAM solutions struggle with a change in operation environment. Also, the SLAM problem does not provide any prior information about the surrounding or the operating conditions. This motivates engineers to research and develop a SLAM solution that can perform in unpredictable real-world environments. LIVO is proposed as a robust odometry solution that can work in both LIDAR-failing and camera-failing GPS-denied environment. To test the accuracy and robustness of the proposed algorithm, indoor and outdoor data are collected using a sensor suite equipped LIDAR, monocular camera, and an IMU. Due to the difficulty of obtaining the outdoor ground truth data, indoor ground truth data is obtained via Optitrack motion capture system. It is shown in Chapter 4 that LIVO can outperform the state-of-the-art solutions by achieving higher accuracy in most cases. In the VIO failing scenario, LIVO not only overcomes the tracking failure due to motion blur caused by fast motion, but also surpasses the accuracy of LIO by adding more constraints between a doublet of robot poses. LIVO is one of the many SLAM solutions that attempts to tackle the robustness and adaptiveness to various environment and provide an accurate and consistent navigation solution.

5.2 Future Work

The proposed work still needs to go through rigorous testing both in simulation and in experimental settings. For instance, LIVO has not been tested in the case when laser-based odometry fails yet vision sustains the navigation system. Such test is essential to prove the adaptiveness of the solution to various environment.

LIVO still suffers from drift after extended periods of time. In practice, SLAM back-end incorporates the loop-closure constraints. Loop closing is the task of deciding whether or not a robot has returned to a previously visited area, and the loop-closure constraints link between the two poses that have visited the same place. Furthermore, the tightly-coupled system requires high computational complexity, and cannot be applied in real time thus far. This problem can be solved by developing a loosely-coupled system with a global pose optimization using loop-closure constraints. Recently Zhao *et al.* [41] proposed coupling the two odometry outputs, LIO and VIO, in a loosely-coupled method and performing a pose graph optimization in the back-end. This method enables the real time application of the solution while preserving the robustness of the solution.

Lastly, the weights associated with the factors used in LIVO are not studied. Consistency of an estimator is equally important as its accuracy because the uncertainty associated with the state estimate determines the confidence in the estimate. As mentioned earlier, the result of the navigation system directly affects guidance and control systems, and a consistent estimator is one where the uncertainty associated with the state estimate is captured properly. In order to achieve consistency, proper weights must be assigned to the factors, and the weights must be derived from the uncertainty characteristics of the sensors. For example, the weight associated with the preintegrated IMU sensor is derived based on the IMU noise parameters. On the other hand, the weights associated with the LIDAR point cloud registration factor nor the depth-based vision factor are derived in this thesis.

Appendices

Appendix A

Marginalization

This section derives the derivation of (4.193). It is based on the assumption that $\mathbf{A}$ is positive semi-definite.

$$J_m\left(\mathcal{X}_m, \mathcal{X}_r\right) \approx J_m\left(\bar{\mathcal{X}}_{\Updownarrow}, \bar{\mathcal{X}}_r\right) + \begin{bmatrix} -\left(\mathbf{b}_{mm} + \mathbf{A}_{mr}\delta\mathbf{X}_r\right)^{\mathsf{T}} \mathbf{A}_{mm}^{-\mathsf{T}} & \delta\mathbf{X}_r^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{b}_{mm} \\ \mathbf{b}_{mr} \end{bmatrix} + \tag{A.1}$$

$$\frac{1}{2} \begin{bmatrix} -\left(\mathbf{b}_{mm} + \mathbf{A}_{mr}\delta\mathbf{X}_r\right)^{\mathsf{T}} \mathbf{A}_{mm}^{-\mathsf{T}} & \delta\mathbf{X}_r^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{mm} & \mathbf{A}_{mr} \\ \mathbf{A}_{rm} & \mathbf{A}_{rr} \end{bmatrix} \begin{bmatrix} -\mathbf{A}_{mm}^{-1}\left(\mathbf{b}_{mm} + \mathbf{A}_{mr}\delta\mathbf{X}_r\right) \\ \delta\mathbf{X}_r \end{bmatrix} \tag{A.2}$$

$$= \bar{J}_m + \left(-\left(\mathbf{b}_{mm} + \mathbf{A}_{mr}\delta\mathbf{X}_r\right)^{\mathsf{T}} \mathbf{A}_{mm}^{-\mathsf{T}} \mathbf{b}_{mm} + \delta\mathbf{X}_r^{\mathsf{T}} \mathbf{b}_{mr} \right) + \tag{A.3}$$

$$\frac{1}{2} \begin{bmatrix} \left(-\left(\mathbf{b}_{mm} + \mathbf{A}_{mr}\delta\mathbf{X}_r\right)^{\mathsf{T}} \mathbf{A}_{mm}^{-\mathsf{T}} \mathbf{A}_{mm} + \delta\mathbf{X}_r^{\mathsf{T}} \mathbf{A}_{rm} \right)^{\mathsf{T}} \\ \left(-\left(\mathbf{b}_{mm} + \mathbf{A}_{mr}\delta\mathbf{X}_r\right)^{\mathsf{T}} \mathbf{A}_{mm}^{-\mathsf{T}} \mathbf{A}_{mr} + \delta\mathbf{X}_r^{\mathsf{T}} \mathbf{A}_{rr} \right)^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} -\mathbf{A}_{mm}^{-1}\left(\mathbf{b}_{mm} + \mathbf{A}_{mr}\delta\mathbf{X}_r\right) \\ \delta\mathbf{X}_r \end{bmatrix}. \tag{A.4}$$

Now expanding the matrices yields

$$J_m \approx \bar{J}_m - \mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} - \delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} + \delta\mathbf{X}_r^{\top}\mathbf{b}_{mr} + \tag{A.5}$$

$$\frac{1}{2}\left(-\left(\mathbf{b}_{mm} + \mathbf{A}_{mr}\delta\mathbf{X}_r\right)^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mm} + \delta\mathbf{X}_r^{\top}\mathbf{A}_{rm}\right)\left(-\mathbf{A}_{mm}^{-1}\left(\mathbf{b}_{mm} + \mathbf{A}_{mr}\delta\mathbf{X}_r\right)\right) + \tag{A.6}$$

$$\frac{1}{2}\left(-\left(\mathbf{b}_{mm} + \mathbf{A}_{mr}\delta\mathbf{X}_r\right)^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mr} + \delta\mathbf{X}_r^{\top}\mathbf{A}_{rr}\right)\left(\delta\mathbf{X}_r\right) \tag{A.7}$$

$$= \bar{J}_m - \mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} - \delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} + \delta\mathbf{X}_r^{\top}\mathbf{b}_{mr} + \tag{A.8}$$

$$\frac{1}{2}\left(-\mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mm} - \delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mm} + \delta\mathbf{X}_r^{\top}\mathbf{A}_{rm}\right)\left(-\mathbf{A}_{mm}^{-1}\mathbf{b}_{mm} - \mathbf{A}_{mm}^{-1}\mathbf{A}_{mr}\delta\mathbf{X}_r\right) + \tag{A.9}$$

$$\frac{1}{2}\left(-\mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mr}\delta\mathbf{X}_r - \delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mr}\delta\mathbf{X}_r + \delta\mathbf{X}_r^{\top}\mathbf{A}_{rr}\delta\mathbf{X}_r\right) \tag{A.10}$$

$$= \bar{J}_m - \mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} - \delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} + \delta\mathbf{X}_r^{\top}\mathbf{b}_{mr} + \tag{A.11}$$

$$\frac{1}{2}\left(\mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} + \delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} - \delta\mathbf{X}_r^{\top}\mathbf{A}_{rm}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm}\right) + \tag{A.12}$$

$$\frac{1}{2}\left(\mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mm}\mathbf{A}_{mm}^{-1}\mathbf{A}_{mr}\delta\mathbf{X}_r + \delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mr}\delta\mathbf{X}_r - \delta\mathbf{X}_r^{\top}\mathbf{A}_{rm}\mathbf{A}_{mm}^{-1}\mathbf{A}_{mr}\delta\mathbf{X}_r\right) + \tag{A.13}$$

$$\frac{1}{2}\left(-\mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mr}\delta\mathbf{X}_r - \delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mr}\delta\mathbf{X}_r + \delta\mathbf{X}_r^{\top}\mathbf{A}_{rr}\delta\mathbf{X}_r\right) \tag{A.14}$$

$$= \bar{J}_m - \mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} - \delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} + \delta\mathbf{X}_r^{\top}\mathbf{b}_{mr} + \tag{A.15}$$

$$\frac{1}{2}\left(\mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} + \cancel{\delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm}} - \cancel{\delta\mathbf{X}_r^{\top}\mathbf{A}_{rm}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm}}\right) + \tag{A.16}$$

$$\frac{1}{2}\left(\cancel{\mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mr}\delta\mathbf{X}_r} + \cancel{\delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mr}\delta\mathbf{X}_r} - \delta\mathbf{X}_r^{\top}\mathbf{A}_{rm}\mathbf{A}_{mm}^{-1}\mathbf{A}_{mr}\delta\mathbf{X}_r\right) + \tag{A.17}$$

$$\frac{1}{2}\left(-\cancel{\mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mr}\delta\mathbf{X}_r} - \cancel{\delta\mathbf{X}_r^{\top}\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{A}_{mr}\delta\mathbf{X}_r} + \delta\mathbf{X}_r^{\top}\mathbf{A}_{rr}\delta\mathbf{X}_r\right) \tag{A.18}$$

$$= \bar{J}_m - \frac{1}{2}\mathbf{b}_{mm}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} + \tag{A.19}$$

$$\delta\mathbf{X}_r^{\top}\left(-\mathbf{A}_{mr}^{\top}\mathbf{A}_{mm}^{-\top}\mathbf{b}_{mm} + \mathbf{b}_{mr}\right) + \frac{1}{2}\delta\mathbf{X}_r^{\top}\left(\mathbf{A}_{rr} - \mathbf{A}_{rm}\mathbf{A}_{mm}^{-1}\mathbf{A}_{mr}\right)\delta\mathbf{X}_r \tag{A.20}$$

Bibliography

[1] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, pp. 1004–1020, Aug 2018.

[2] H. Ye, Y. Chen, and M. Liu, "Tightly Coupled 3D Lidar Inertial Odometry and Mapping," *CoRR*, vol. abs/1904.06993, 2019.

[3] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[4] G. Bourmaud, R. Mégret, M. Arnaudon, and A. Giremus, "Continuous-Discrete Extended Kalman Filter on Matrix Lie Groups Using Concentrated Gaussian Distributions," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 1, pp. 209–228, 2015.

[5] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient Sparse Pose Adjustment for 2D mapping," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 22–29, 2010.

[6] A. Barrau and S. Bonnabel, "Invariant Kalman Filtering," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, 05 2018.

[7] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," *CoRR*, vol. abs/1812.01537, 2018.

[8] T. D. Barfoot and P. T. Furgale, "Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014.

[9] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended Kalman filtering for robot state estimation," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 402–430, 2020.

[10] A. Barrau and S. Bonnabel, "The Invariant Extended Kalman Filter as a Stable Observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2017.

[11] J. Arsenault, *Practical Considerations and Extensions of the Invariant Extended Kalman Filtering Framework*. McGill theses, McGill University Libraries, 2020.

[12] N. van der Laan, M. Cohen, J. Arsenault, and J. R. Forbes, "The Invariant Rauch-Tung-Striebel Smoother," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5067–5074, 2020.

[13] T. Lupton and S. Sukkarieh, "Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 61–76, 2012.

[14] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry," *IEEE Transactions on Robotics*, vol. 33, pp. 1–21, Feb 2017.

[15] J. A. Farrell, *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill professional engineering: Electronic engineering, McGraw-Hill Education, 2008.

[16] S.-C. J. Hsiung, "Toward Invariant Visual-Inertial State Estimation using Information Sparsification," Master's thesis, Carnegie Mellon University, Pittsburgh, PA, August 2018.

[17] S. Heo and C. G. Park, "Consistent EKF-Based Visual-Inertial Odometry on Matrix Lie Group," *IEEE Sensors Journal*, vol. 18, no. 9, pp. 3780–3788, 2018.

[18] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI)," vol. 81, 04 1981.

[19] C. G. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Alvey Vision Conference*, 1988.

[20] J. Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

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

[22] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, 01 2016.

[23] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 ed., 2004.

[24] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), IEEE, May 9-13 2011.

[25] D. Qian, "Weighted Optimal Linear Attitude and Translation Estimator: Theory and Application," Master's thesis, McGill University, Dec. 2018.

[26] J. Zhang and S. Singh, "Low-drift and Real-time Lidar Odometry and Mapping," *Autonomous Robots*, vol. 41, pp. 401 – 416, February 2017.

[27] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[28] N. S. Altman, "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[29] M. Brossard, S. Bonnabel, and A. Barrau, “A New Approach to 3D ICP Covariance Estimation for Mobile Robotics,” *CoRR*, vol. abs/1909.05722, 2019.

[30] K.-L. Low, “Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration,” tech. rep., University of North Carolina, Feb. 2004.

[31] D. Qian, G. Charland-Arcand, and J. R. Forbes, “TWOLATE: Total Registration of Point-Clouds Using a Weighted Optimal Linear Attitude and Translation Estimator,” in *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 43–48, 2020.

[32] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *Journal of Basic Engineering*, vol. 82, pp. 35–45, 03 1960.

[33] S. Agarwal, K. Mierle, and Others, “Ceres Solver.” `http://ceres-solver.org`.

[34] A. P. Badali, Y. Zhang, P. Carr, P. J. Thomas, and R. I. Hornsey, “Scale factor in digital cameras,” in *Photonic Applications in Biosensing and Imaging* (B. C. Wilson, R. A. Weersink, R. I. Hornsey, W. C. W. Chan, K. Yu, and U. J. Krull, eds.), vol. 5969, pp. 556 – 565, International Society for Optics and Photonics, SPIE, 2005.

[35] J. Zhang and S. Singh, “LOAM: Lidar Odometry and Mapping in real-time,” *Robotics: Science and Systems Conference (RSS)*, pp. 109–111, 01 2014.

[36] D. Pojar, P. Jeong, and S. Nedevschi, “Robust visual odometry using stereo reconstruction error model,” in *2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing*, pp. 149–154, 2012.

[37] G. Sibley, L. Matthies, and G. Sukhatme, “Constant time sliding window filter slam as a basis for metric visual perception,” 01 2007.

[38] T. Dong-Si and A. I. Mourikis, “Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis,” in *2011 IEEE International Conference on Robotics and Automation*, pp. 5655–5662, 2011.

[39] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-Time Loop Closure in 2D LIDAR SLAM,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278, 2016.

[40] M. Grupp, “evo: Python package for the evaluation of odometry and SLAM..” `https://github.com/MichaelGrupp/evo`, 2017.

[41] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. A. Scherer, “Super Odometry: IMU-centric LiDAR-Visual-Inertial Estimator for Challenging Environments,” *CoRR*, vol. abs/2104.14938, 2021.