Weighted Optimal Linear Attitude and Translation Estimator: Theory and Application

by

Duowen Qian

Department of Mechanical Engineering
McGill University, Montreal
December 2018

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

Supervisors:

Professor James Richard Forbes

Duowen Qian
duowen.qian@mail.mcgill.ca

Acknowledgements

First, I would like to thank Professor James Richard Forbes for his invaluable advice and supervision throughout the course of my Masters program. Prof. Forbes has exceeded all expectations of what a supervisor ought to be, challenging me every step of the way, providing relentless guidance and support in hours of need, and always making himself available for those unplanned emergency meetings to discuss sudden strokes of insight. This work was supported by Mitacs through the Mitacs Accelerate program. Specifically, I would like to thank ARA Robotique and Mitacs for providing me with the resources and opportunity to apply my research in practice. Special thanks to Guillaume Charland-Arcand, Pascal Chiva-Bernard and the rest of the folks at ARA Robotique for being so understanding and supportive throughout the course of this past year. Last but not least, I would like to thank Alex-Ane Mathieu for supporting me every step of the way and Mom and Dad for always believing in me, through all the difficulties and hardships.

Preface

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

- Chapter 3
  - A systematic approach to modelling LIDAR range uncertainty is described and a model of the RP-LIDAR is experimentally obtained.
- Chapter 5
  - The point-to-point WOLATE algorithm is presented.
  - A point-to-plane WOLATE algorithm is derived.
  - A point-to-line WOLATE algorithm is derived.
  - The Total Registration algorithm, which combines point-to-point, point-to-plane and point-to-line registration is presented.
  - A constrained variant of the OLATE/WOLATE algorithm is derived.

All text, plots, illustrations, and numerical and experimental results in this thesis are produced by Duowen Qian. James R. Forbes and Tim D. Barfoot derived the original point-to-point Weighted OLATE registration method which is in turn based on [1]. Duowen Qian modified this Weighted OLATE method for application to point-to-plane and point-to-line registration. This led to a combined registration algorithm that is linear and easily implementable in feature-based scan matching algorithms.

Table of Contents

List of Figures

Figure

List of Tables

Table

List of Abbreviations

SLAM	Simultaneous Localization and Mapping
OLATE	Optimal Linear Attitude and Translation Estimator
WOLATE	Weighted Optimal Linear Attitude and Translation Estimator
ICP	Iterative Closest Point
DCM	Direction Cosine Matrix
IMU	Inertial Measurement Unit
LIDAR	Light Detection and Ranging
RGBD	Red Green Blue Depth
RTK	Real-Time Kinematic
UAV	Unmanned Aerial Vehicle
VO	Visual Odometry
GN	Gauss-Newton
LM	Levenberg-Marquardt

List of Symbols

$\|\cdot\|$	the Euclidian norm of a physical vector
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^n$	the vector space of real n-dimensional vectors
$\mathbb{R}^{m\times n}$	the space of real $m \times n$-dimensional matrices
$\mathbb{Z}$	the set of integers
$SO(n)$	the n-dimensional special orthogonal group
$\mathfrak{so}(n)$	the n-dimensional Lie algebra associated with $SO(n)$, the set of antisymmetric matrices on $\mathbb{R}^{n\times n}$
$SE(n)$	the n-dimensional special Euclidian group
$\mathfrak{se}(n)$	the n-dimensional Lie algebra associated with $SE(n)$.
$\mathrm{tr}(\cdot)$	trace
$(\cdot)^\mathsf{T}$	transpose
$(\cdot)^\times$	cross operator for $\mathfrak{so}(3)$
$(\cdot)^\wedge$	cross operator for $\mathfrak{se}(3)$
$(\cdot)^\vee$	uncross operator
$\mathbf{0}$	zero matrix
$\mathbf{1}$	identity matrix
$\mathrm{diag}\,(\mathbf{M}_1, \ldots, \mathbf{M}_n)$	block diagonal matrix with $\mathbf{M}_1$, $\ldots$, $\mathbf{M}_n$ on diagonals, and zeros elsewhere

$\mathbf{M} > 0$	matrix $\mathbf{M}$ is symmetric positive definite.
$\star$	symmetric portion of a matrix
$\mathcal{F}_i$	the datum frame or inertial frame, depending on context
$\mathcal{F}_v$	the vehicle's body frame
$\mathcal{F}_s$	the sensor frame
$\underrightarrow{r}$	a physical vector
$\underrightarrow{r}^{pq}$	the position of point p relative to point q
$\underrightarrow{r}^{\bullet a}$	the time derivative of $\underrightarrow{r}$ with respect to $\mathcal{F}_a$
$\underrightarrow{r}^{pq\bullet i} = \underrightarrow{v}^{pq/i}$	velocity of point p relative to point q with respect to $\mathcal{F}_i$
$\underrightarrow{\boldsymbol{\mathcal{F}}}_a$	a vectrix, that is a matrix of unit length physical vectors that form a basis for $\mathcal{F}_a$, where $\underrightarrow{\boldsymbol{\mathcal{F}}}_a^{\mathsf{T}} = [\underrightarrow{a}^1 \ \ \underrightarrow{a}^2 \ \ \underrightarrow{a}^3]$
$\mathbf{r}_a$	the physical vector $\underrightarrow{r}$ resolved in $\mathcal{F}_a$
$\mathbf{C}_{vi}$	a DCM parameterizing the attitude of $\mathcal{F}_v$ relative to $\mathcal{F}_i$
$\underrightarrow{\omega}^{ba}$	angular velocity of $\mathcal{F}_b$ relative to $\mathcal{F}_a$

Abstract

This thesis investigates the design of an accurate autonomous navigation strategy for both ground and aerial vehicles in GPS-denied environments. Current methods for Simultaneous Localization and Mapping (SLAM) are often situation specific with performance dictated by the environment in which it operates. The proposed approach uses accurate LIDAR-based scan-matching and attempts to be more adaptive to different environments by matching different types of feature points. A novel weighted optimal linear attitude and translation estimator (WOLATE) is first derived and used as the backbone point cloud registration algorithm. Specifically, point-to-point, point-to-plane, and point-to-line variants of WOLATE are derived and are shown to each be suitable for different types of points, be it feature points, surface-points, or edge-points. This suite of registration algorithms are embedded into the standard Iterative Closest Point (ICP) framework and shown to outperform standard registration methods in terms of accuracy. In particular, when combined with a feature extractor capable of extracting edge and surface features, the point-to-point, point-to-plane, and point-to-line algorithms, when working in tandem, not only ensure that each data point is paired with its most suitable cost function, but also allows the distribution of weights to each point. Furthermore, the linear nature of the solution makes it easy to constrain certain states for which a reliable estimate is already available *a priori*. This approach is shown to be more adaptive to different types of incoming point-cloud data and less prone to drift when used for long trajectory estimation. The resulting local SLAM pipeline is tested on simulated 2D data where the scene can be controlled and where groundtruth is available. The pipeline is then validated on experimental 3D data.

Résumé

Ce mémoire étudie la conception d'un système de navigation autonome précis pour les véhicules terrestres et aériens dans les environnements sans GPS. Les méthodes actuelles de localisation et cartographie simultanées (SLAM) sont souvent appliquées à des situations spécifiques et leurs performances sont dictées par l'environnement qui est exploré. L'approche proposée utilise une correlation précise des balayages LIDAR et tente de mieux s'adapter à différents environnements en reliant différents types de points d'intérêt. Un nouvel estimateur optimal linéaire d'attitude et de translation utilisant des poids (WOLATE) est d'abord dérivé et utilisé comme algorithme de base pour enregistrer des nuages de points. Plus spécifiquement, les variations point-à-point, point-à-plan, et point-à-ligne de WOLATE ont été dérivées et ont été demontrées comme étant applicables à différents types de points d'intérêt, qu'il s'agisse de points individuels, de points de surface ou de points de coin. Cet ensemble d'algorithmes d'enregistrement est intégré au cadre ICP (Iterative Closest Point) et est démontré supérieur, en terme de précision, aux méthodes d'enregistrement de référence. En particulier, lorsqu'ils sont combinés avec un extracteur de point d'intérêt capable d'extraire des points de coin et de surface, les algorithmes point-à-point, point-à-plan et point-à-ligne fonctionnant en tandem ne garantissent pas seulement que chaque point est associé à la fonction de cot la plus appropriée, mais garantissent également la répartition des poids pour chaque point. De plus, la nature linéaire de la solution facilite la contrainte de certains états pour lesquels une estimation fiable est déjà disponible. Cette approche s'avère plus adaptable à différents types de nuages de points et moins sujette à la déviation lorsqu'elle est utilisée pour l'estimation d'une longue trajectoire. Le processus SLAM résultant est testé sur des données 2D simulées dans lesquelles l'environment peut être contrôlé et où la vraie trajectoire est connue. Le processus est ensuite validé sur des données 3D expérimentales.

Chapter 1

Introduction

Robotics is arguably the study of giving life, or autonomy to machines. For a robot to realize autonomy, it must solve the navigation, guidance, and control problems. Navigation answers the questions "where is the robot". Guidance answers the question "where should the robot go or what path must the robot follow, given its current location". Control answers the question "what inputs must be applied to the robot in order to follow the desired path". Information from each stage of this problem is passed onto the next stage. Thus, errors made earlier on can have irreparable impact. Ensuring that the navigation problem is solved accurately and efficiently is therefore of crucial importance to the quality of the solution to the overall autonomy problem.

Simultaneous Localization and Mapping (SLAM) is a general term to describe the suite of methods and algorithms used to solve a more sophisticated navigation problem, that being estimation of where objects in the environment are located relative to each other, and estimation of where the robot is located relative to the objects.

Many roboticists consider SLAM to be a solved problem [3]. Though this may be true on a case-by-case basis, much research still needs to be conducted to obtain a solution that is robust to different type of sensor measurements, arrangements, scenes, and environment conditions. For example, a ground vehicle equipped with wheel encoders and GPS can attain fairly accurate pose estimates while operating outdoors, but if given only a monocular camera and an IMU while operating indoors, the pose estimate of the robot can be very poor. Similarly, there exist many low-drift visual odometry solutions using stereo or RGBD cameras, but these algorithms will break under poor lighting conditions. Another example would be the suite of feature-based visual odometry methods that will fail in featureless environments, such as a long road or corridor. Furthermore, all sensor measurements are corrupted by noise, and in the event where the noise is sufficiently large, any SLAM solution will be affected. As described in Cadena *et al.* [3], we are entering the *robust-perception-*

age characterized by robustness to different environments and scenarios, resource awareness, and task-driven perception. In this thesis, special attention is paid to UAVs operating in both GPS-available and GPS-denied environments where odometry is difficult to obtain. This motivates the development of a reliable scan-matching algorithm. Furthermore, the inherent noise in all measurements is recognized and utilized to improve current state of the art LIDAR-based SLAM methods to be less affected by poor measurement conditions.

1.1 Thesis Objectives

The objective of this thesis is to design a LIDAR-based scan-matching method that is more accurate than existing methods. The contribution of this thesis is the derivation of a novel point cloud registration method, henceforth called Total Registration, that leverages different types of points, such as feature, surface, and edge points, and can weigh individual point uncertainty on a component level to achieve greater accuracy.

A major contribution of this thesis is the derivation of the standalone registration methods that lead up to the formulation of the Total Registration algorithm. These include the point-to-point Weighted Optimal Linear Attitude and Translation Estimator (WOLATE), point-to-plane WOLATE, and point-to-line WOLATE. Individual points in a point cloud originated from a LIDAR or a camera are always accompanied with varying uncertainties. WOLATE provides a way to account for these uncertainties, something current-day methods tend to ignore. Furthermore, a constrained variant is derived to account for prior state estimates.

Another contribution of this thesis is the fusion of a wide variety of methods designed to tackle different parts of the SLAM problem into a cohesive local SLAM pipeline. To perform SLAM from start to finish is not a trivial task. In fact, the majority of SLAM resources are in the form of conference and journal papers that tackle only a small piece of the SLAM pipeline or describe a particular application. This thesis will offer an overview of what a common LIDAR-based SLAM pipeline would look like and present tools that are tailored to weighted point cloud scan matching.

1.2 Thesis Overview

The remainder of Chapter 1 will offer a brief overview of SLAM history and introduce both classical and new methods for the front and back-end of SLAM.

Chapter 2 will cover basic mathematical notation and concepts that will be applied throughout this paper.

In Chapter 3, measurements from a commercial-grade LIDAR is modelled in order to

simulate noise in later simulations.

In Chapter 4, data preprocessing techniques are introduced and a strategy is offered to tackle different data inputs.

In Chapter 5, the local scan matching strategy for both 2D and 3D LIDAR data is presented. In particular, the novel point-to-plane WOLATE registration method and its point-to-plane and point-to-line variants will be derived. The Total Registration algorithm will also be outlined.

In Chapter 6, comparison results between WOLATE and existing registration methods are provided. Tests conducted on simulation data across a wide range of scenarios will show accuracy improvements compared to traditional methods.

Chapter 7 concludes the thesis along with recommendations for future work.

1.3 A Brief History of SLAM

Early formulation of the SLAM problem within a probabilistic framework is generally attributed to Smith, Self, and Cheeseman [4] who showed that as a mobile robot moves through an unknown environment, taking relative landmark measurements, the estimates of these landmarks are correlated with each other because of the common error in estimated vehicle location. Since then, a number of solutions to the SLAM problem have been presented, most of which differ in either the front-end data-association approach or the back-end estimation approach used. The data-association problem deals with the correct identification of constraints based on the acquired sensor measurements. For image data, this is a computer vision problem solved with feature matching whereas for laser range-finder data, laser scan matching or point-cloud alignment are popular methods. The back-end estimation problem deals with solving for the configuration of the robot states that maximizes the likelihood of the measurements taken. The full SLAM solution is a marriage between the front-end and back-end methodologies. Each problem will now be introduced separately, starting with the front-end.

1.3.1 The SLAM Front-End

The first step of SLAM involves processing incoming noisy sensor data and interpreting it in a meaningful and useful way. Most importantly, the relationship between each measurement, be it taken immediately after one another, or taken in between long periods of time, must be established such that their information can be optimized later on. These tasks are collectively referred to as the SLAM front-end.

In the case of UAVs, pose-to-pose constraints that can be obtained via wheel odometry for ground vehicles is no longer an option. Instead, *dead-reckoning*, which involves integrating linear acceleration and angular velocity obtained via an Inertial Measurement Unit (IMU) is a potential odometry method, though the 3D transformation obtained this way can be heavily biased, noisy, and prone to drift. A common way to mitigate this drift is to use the gravity vector obtained through the accelerometer and a heading obtained through the magnetometer of an IMU to correct accumulating errors. Instruments that perform this correction are known as attitude and heading reference systems (AHRS).

Visual odometry (VO) refers to the suite of methods that aim to obtain a good local estimate of successive pose-to-pose measurements via camera images. VO is often used to mitigate the drift problem of dead reckoning. Visual odometry estimate pose changes by analyzing the change in subsequent camera images as the camera is in motion [5]. Visual odometry methods can be broadly categorized into feature-based methods and direct methods. Feature-based methods detect and extract features of interest from an image and track them through subsequent frames, thus estimating camera motion. Early features were merely image patches with high gradiants such as corners and edges. More sophisticated features descriptors such as SIFT [6] features that are scale invariant were soon established as more robust and reliable features to track. Other feature descriptors such as SURF [7], ORB [8] or BRIEF [9] features were also devised to be more efficient or have better scale and orientation invariance. Perhaps the most ground-breaking paper in feature-based VO is that of Parallel Tracking and Mapping (PTAM), which makes use of the FAST corner detector as well as the novel idea of separating feature tracking and mapping into separate threads [10]. Another well-known feature-based VO method is RGB-D SLAM [11], which detected SIFT, SURF, and ORB features, projected them to 3D using the depth measurement of the RGB-D camera, and performed 3D point cloud registration. State of the art feature-based VO methods such as ORB-SLAM [12] and ORB-SLAM2 [13] have since become the benchmark in feature-based VO while more recently, point and edge-based methods such as PL-SLAM [14] and EdgeSLAM [15] have seen a surge in interest. In general, feature-based VO methods operate on a subset of pixels and can thus run efficiently in real-time. Furthermore, they can utilize bag-of-words [16, 17] techniques to characterize features for loop-closure detection. On the other hand, direct methods operate directly on the image pixels and extract image-to-image transforms by minimizing the photometric error between reprojected pixels and their reference pixels. Early direct methods such as DTAM [18] sparked interest in the field of direct VO due to its ability to reconstruct the entire image and function under featureless environments. Later, methods such as SVO [19] popularized the use of a sparse subset of all the available pixels of the image instead of the entire image. Some state-of-the-art methods

in recent years such as LSD-SLAM [20] and DSO [21] can run in real-time and outperform feature-based methods in certain conditions. To conclude, these vision-based methods can be applied to monocular, stereo, or RGBD cameras, but because these sensors obtain depth via intermediate reprojection steps, the raw data must be constantly maintained to ensure that no outliers corrupt the motion estimate. Furthermore, all camera-based visual-odometry methods are affected by lighting conditions and can outright fail in poorly lit indoor areas.

Another alternative to obtaining pose-to-pose constraints is to use laser scan-matching of LIDAR data. Because LIDAR measure the depth of the environment directly, they are considered more accurate than visual reconstruction, and are even lighting-independent. This comes at the cost of losing measurement correspondences. Specifically, points measured at a subsequent step do not correspond to points in the previous step and there is not even any guarantee that the same scene is being measured. Scene overlap is implicitly assumed and scan-matching methods are relied upon to reconstruct the scene and obtain pose estimates. When using 3D LIDAR such as the Velodyne Puck, methods such as LOAM [22] and V-LOAM [23] are among the best performing algorithms reported on the popular KITTI [24] benchmark datasets. Google Cartographer [25], a popular open-source package, uses Ceres [26], an open-source nonlinear optimization solver, coupled with Real-time Correlative Scan Matching [27] as the backbone scan matching algorithms. Although occupancy-grid methods such Olsons method [27] has recently taken a front seat for 2D scan-matching, they tend to treat measurement uncertainty on the occupancy grid level, abstracting away component-based uncertainty to a point uncertainty. Traditional Iterative Closest Point (ICP)-based methods, or variants thereof, are still preferred in many applications, especially in 3D scan-matching, due to their versatility and ease of use. Some recently developed ICP-variants include point-to-plane ICP [28], Iterative Closest Line (ICL) [29], Iterative Dual Correspondence (IDC) [30], Probabilistic Iterative Correspondence (pIC) [31] or Polar Scan Matching (PSM) [32]. The Weighted OLATE method to be derived in this thesis is a novel point registration method that falls in this category of algorithms. Unlike some of its predecessors, it can weigh measurement uncertainty and embed point cloud features into a combined linear formulation.

1.3.2 The SLAM Back-End

Once pose-to-pose constraints have been established, uncertainties and bias can be mitigated if given additional exteroceptive information. This is commonly known as the back-end of SLAM.

The first attempt at solving the SLAM back-end is using a nonlinear version of the canonical Kalman Filter (KF) [33], which employs a prediction-correction strategy to in-

corporate lower frequency exteroceptive measurements into higher frequency odometry predictions. In particular, its nonlinear variants, the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF), are used in 2D or 3D navigation. These filter-based methods performed well for a wide range of linear and nonlinear problems, and later on, with advancements made in the field of Sparse Matrices, methods such as the Sparse Extended Information Filter (SEIF) [34] were also employed to take advantage of the sparsity of the information matrix to solve each iteration effectively. However, filter methods come with a set of inherent problems [35]. Researchers soon realized that filter-based algorithms cannot effectively update past estimates given new information. This is because filter-based algorithms marginalize away old robot poses and, to make matters worse, they destroy the sparse structure of the information matrix in doing so [36]. To get around this problem, researchers found that for a fixed amount of computing time, batch methods that keep some or all parts of the robot's old poses, can perform more accurately than filter-based approaches. Batch methods are more popularly known as graph-based methods. They were coined as such because the nodes and edges in factor graphs [37] were used to represent the interdependence between robot and landmark poses [11]. The goal of all graph-based methods is therefore "to jointly optimize the poses of the nodes so as to minimize the error introduced by the constraints" [38]. Graph-optimization was historically regarded as too computationally expensive for real-time implementation. However, recent advancements in the field of direct linear solvers coupled with insights on the naturally sparse structure of the factor graph paved way for a resurgence in these methods. Many approaches to solving this graph optimization problem have since been proposed. The graph optimization problem is most popularly formulated as a least-squares problem and can be solved using methods such as Gauss-Newton (GN) or Levenberg-Marquardt (LM), a middle-ground between GN and gradient descent. Specifically, methods such as GraphSLAM [39] solve the direct linearized problem whereas smoothing methods such as rootSAM [40] iSAM [36], iSAM2 [41] and SPA [38] factorize the sparse information matrix first via either QR or Cholesky decomposition. The sparse structure of the factors are then maintained while periodic batch steps and variable reordering are used to relinearize the system and avoid error build-up.

Chapter 2

Preliminaries

The contents of this thesis, namely estimation, involves aspects of linear algebra, vector operations, kinematics, optimization, and matrix Lie groups. As such, an overview of the most relevant topics is presented in this chapter.

2.1 Physical Vectors and Reference Frames

A *physical vector* $\underrightarrow{v}$ is an element of *physical space* $\mathbb{P}$ that has a magnitude and a direction. Note that the definition of a physical vector does not involve any reference frame.

Physical vectors can be used to denote the relative position between two points. Given points p and q, $\underrightarrow{r}^{qp}$ denotes the position of point q relative to point p.

A *reference frame* $\mathcal{F}$ is composed of three orthonormal physical vectors that are called *basis vectors*. The basis vectors can be written together in a column matrix as

$$\underrightarrow{\boldsymbol{\mathcal{F}}}_a = \begin{bmatrix} \underrightarrow{a}^1 \\ \underrightarrow{a}^2 \\ \underrightarrow{a}^3 \end{bmatrix}.$$

This is also known as a *vectrix* because it is a column matrix of physical vectors.

A vector $\underrightarrow{r}^{qp}$ resolved in frame $\mathcal{F}_a$ is thus

$$\underrightarrow{r}^{qp} = \underrightarrow{\boldsymbol{\mathcal{F}}}_a^{\mathsf{T}} \mathbf{r}_p^{qp}$$

where $\mathbf{r}_a^{qp}$ is the column matrix whose elements represent the components of the physical vector $\underrightarrow{r}^{qp}$ resolved in $\mathcal{F}_a$.

2.2 Direction Cosine Matrices and Rotations

Given a physical vector $\underrightarrow{r}$ resolved in frames $\mathcal{F}_a$ and $\mathcal{F}_b$,

$$\underrightarrow{r} = \underrightarrow{\boldsymbol{\mathcal{F}}}_a^{\mathsf{T}} \mathbf{r}_a = \underrightarrow{\boldsymbol{\mathcal{F}}}_b^{\mathsf{T}} \mathbf{r}_b,$$

by taking the dot product from the left by $\mathcal{F}_b$ it follows that

$$\mathbf{r}_b = \underrightarrow{\boldsymbol{\mathcal{F}}}_b \cdot \underrightarrow{\boldsymbol{\mathcal{F}}}_a^{\mathsf{T}} \mathbf{r}_a.$$

where

$$\mathbf{C}_{ba} = \underrightarrow{\boldsymbol{\mathcal{F}}}_b \cdot \underrightarrow{\boldsymbol{\mathcal{F}}}_a^{\mathsf{T}}.$$

The matrix $\mathbf{C}_{ba}$ is known as the Direction Cosine Matrix (DCM). When interpreted as a change of perspective, $\mathbf{C}_{ba}$ changes the perspective of any vector as seen from (resolved in) $\mathcal{F}_a$ to a new perspective as seen from (resolved in) $\mathcal{F}_b$. Put another way, $\mathbf{C}_{ba}$ resolves any vector in $\mathcal{F}_a$ to a new frame $\mathcal{F}_b$. When interpreted as a rotation, $\mathbf{C}_{ab}$, $\mathbf{C}_{ba}$, or simply $\mathbf{C}$, is an orthogonal matrix that rotates any physical vector by a new attitude, regardless of what frame it is resolved in. When this rotation is applied to the basis vectors of an arbitrary frame $\mathcal{F}_a$, the resulting rotated basis vectors will form a new frame that can be denoted as $\mathcal{F}_b$. Throughout this thesis, DCMs will be the preferred choice of interpreting attitude changes between a robot-centered frame and some datum reference frame.

Note that

$$\mathbf{C}_{ba} = \mathbf{C}_{ab}^{\mathsf{T}}.$$

2.3 Angular Velocity

Given Poisson's equation,

$$\dot{\mathbf{C}}_{ba} + \boldsymbol{\omega}_b^{ba\times} \mathbf{C}_{ba} = \mathbf{0},$$

where the $\times$ operator is the skew-symmetric operator for $\mathfrak{so}(3)$ given by

$$\mathbf{v}^{\times} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix},$$

and where $\boldsymbol{\omega}_b^{ba}$ denotes the angular velocity of $\mathcal{F}_b$ relative to $\mathcal{F}_a$ resolved in $\mathcal{F}_b$. Poisson's equation can also be written as

$$\dot{\mathbf{C}}_{ab} = \mathbf{C}_{ab}\boldsymbol{\omega}_b^{ba\times}. \tag{2.1}$$

Equation (2.1) is sometimes more useful if given a $T = t_k - t_{k-1}$, where the relative orientation between $\mathcal{F}_{b_{k-1}}$ and $\mathcal{F}_{b_k}$ can then be computed as

$$\mathbf{C}_{ab_k} = \mathbf{C}_{ab_{k-1}} e^{\boldsymbol{\psi}_{k-1}^{\times}}$$

where

$$\boldsymbol{\psi}_{k-1} = \boldsymbol{\omega}_b^{ba} T.$$

2.4 Rigid-Body Transformation Matrices and Poses

A rigid-body transformation is composed of a translation and a rotation, which can be carried out in any order. Denoted $\mathbf{T}$, the transformation matrix "translates" a vector to a new location and then rotates it. To be more accurate, "translate a vector" is the act of adding a vector with another vector, yielding a new vector. Specifically,

$$\mathbf{T}_{ab} = \begin{bmatrix} \mathbf{C}_{ab} & \mathbf{r}_a^{ba} \\ \mathbf{0} & 1 \end{bmatrix}$$

contains both a rotation $\mathbf{C}_{ab}$ that rotates a physical vector to a new orientation, and a translation $\mathbf{r}_a^{ba}$ that transforms a physical vector resolved in $\mathcal{F}_a$ to a new vector via vector addition. Thus, the product of a rigid body transformation matrix and a vector $\underrightarrow{r}^{pb}$ resolved in $\mathcal{F}_b$ is

$$\mathbf{T}_{ab}\mathbf{r}_b^{pb} = \begin{bmatrix} \mathbf{C}_{ab} & \mathbf{r}_a^{ba} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{r}_b^{pb} \\ 1 \end{bmatrix} = \mathbf{C}_{ab}\mathbf{r}_b^{pb} + \mathbf{r}_a^{ba} = \mathbf{r}_a^{pa}.$$

2.4.1 Robotics Convention Regarding Poses [2]

The pose of a robot can be described using a transformation matrix. Specifically, the pose of a robot is defined as

$$\mathbf{T}_{iv} = \begin{bmatrix} \mathbf{C}_{iv} & \mathbf{r}_i^{vi} \\ \mathbf{0} & 1 \end{bmatrix}.$$

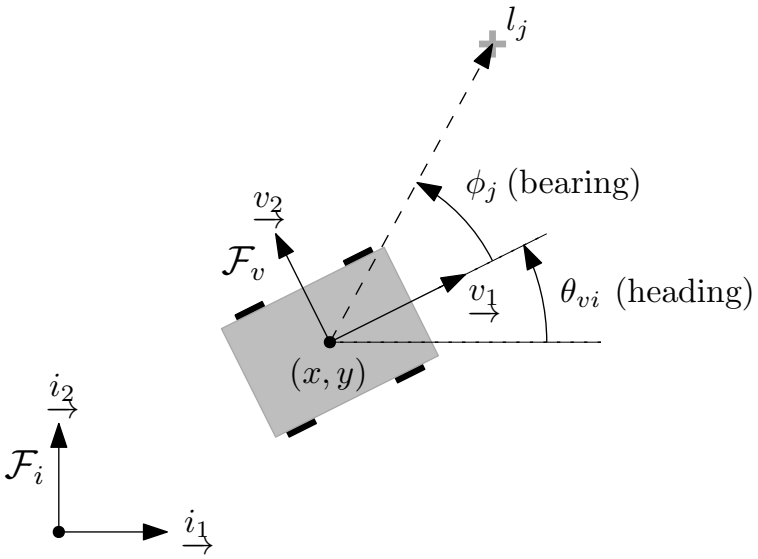


Figure 2.1: Robotics convention for a 2D planar robot [2].

The frame $\mathcal{F}_i$ is the inertial or datum frame, and $\mathcal{F}_v$ is the vehicle frame. Referring to Fig. 2.1, the position of the robot resolved in $\mathcal{F}_i$ is

$$\mathbf{r}_i^{vi} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}.$$

The orientation of the robot, that being the orientation of frame $\mathcal{F}_v$ relative to frame $\mathcal{F}_i$, can be described by the DCM

$$\mathbf{C}_{vi} = \begin{bmatrix} \cos\theta_{vi} & \sin\theta_{vi} & 0 \\ -\sin\theta_{vi} & \cos\theta_{vi} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

It is more natural to use θ_{vi} as it represents the heading of the robot. However, the pose of a robot is defined by $\mathbf{C}_{iv}$, the rotation matrix of θ_{vi}. Thus to keep using θ_{vi} instead of θ_{iv},

$$\mathbf{C}_{iv} = \mathbf{C}_{vi}^{\mathsf{T}} = \begin{bmatrix} \cos\theta_{vi} & -\sin\theta_{vi} & 0 \\ \sin\theta_{vi} & \cos\theta_{vi} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

In some situations, it is preferred to write the state of a robot as

$$\mathbf{T}_{vi} = \mathbf{T}_{vi}^{-1} = \begin{bmatrix} \mathbf{C}_{vi} & -\mathbf{C}_{vi}\mathbf{r}_i^{vi} \\ \mathbf{0} & 1 \end{bmatrix}.$$

2.5 Matrix Lie Groups [2]

The *special orthogonal group*, denoted $SO(3)$, is the set of all matrices $\mathbf{C} \in \mathbb{R}^{3\times 3}$ that satisfy $\mathbf{C}\mathbf{C}^\mathsf{T} = \mathbf{1}$ and $\det \mathbf{C} = 1$.

The *special euclidian group*, denoted $SE(3)$, is the set of all transformation matrices $\mathbf{T} \in \mathbb{R}^{4\times 4}$ where $\mathbf{C} \in SO(3)$ and $\mathbf{r} \in \mathbb{R}^3$.

Associated with every matrix Lie group is a matrix Lie algebra, that is a vector space. The vector space of a Lie algebra is the *tangent space* of its associated Lie group at identity, and perfectly describes the local structure of that group.

The Lie algebra of the $SO(3)$ group, denoted $\mathfrak{so}(3)$, is the vector space $\boldsymbol{\Phi} = \boldsymbol{\phi}^\wedge \in \mathbb{R}^{3\times 3}$ where $\boldsymbol{\phi} \in \mathbb{R}^3$.

Note that we have adopted the $(\cdot)^\wedge$ skew-symmetric operator that maps $\mathbb{R}^3 \to \mathbb{R}^{3\times 3}$. Given a column matrix $\boldsymbol{\phi} = \begin{bmatrix} \phi_1 & \phi_2 & \phi_3 \end{bmatrix}^\mathsf{T}$, $\boldsymbol{\phi}^\wedge$ is given by

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

Similarly, the $(\cdot)^\vee$ operator maps a $\mathbb{R}^{3\times 3}$ skew-symmetric matrix back to $\mathbb{R}^3$ and $(\boldsymbol{\phi}^\wedge)^\vee$ is given by

$$(\boldsymbol{\phi}^\wedge)^\vee = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}^\vee = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \boldsymbol{\phi}.$$

The Lie algebra of the $SE(3)$ group, denoted $\mathfrak{se}(3)$, is the vectorspace $\boldsymbol{\Xi} = \boldsymbol{\xi}^\wedge \in \mathbb{R}^{4\times 4}$ where $\boldsymbol{\xi} \in \mathbb{R}^6$.

For the $SE(3)$ case, the $(\cdot)^\wedge$ skew-symmetric operator maps $\mathbb{R}^6$ to a $\mathbb{R}^{4\times 4}$ of the form

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0} & 0 \end{bmatrix}$$

2.6 Exponential Map

The exponential map relates elements of a Lie group to its associated Lie algebra.

For rotations, the matrix exponential maps elements of $\mathfrak{so}(3)$ to $SO(3)$ via

$$\begin{aligned} \exp(\boldsymbol{\phi}^\wedge) &= \mathbf{C}, \\ \boldsymbol{\phi} &= \ln(\mathbf{C})^\vee. \end{aligned}$$

For poses, the matrix exponential maps elements of $\mathfrak{se}(3)$ to $SE(3)$ via

$$\begin{aligned} \exp(\boldsymbol{\xi}^\wedge) &= \mathbf{T}, \\ \boldsymbol{\xi} &= \ln(\mathbf{T})^\vee. \end{aligned}$$

2.7 Solving Linear Least Squares Problems

The standard form of any optimization problem is

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} J(\mathbf{x})$$

where $J(\mathbf{x})$ is the *objective* or *cost* function to be minimized, and $\mathbf{x}$ are the design variables.

One specific type of optimization problems, called least squares problems, take the form

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{e}(\mathbf{x})\|_2^2$$

because the objective function $J(\mathbf{x})$ is the square of a residual $\mathbf{e}(\mathbf{x})$. In particular, if the residual can be written in the form $\mathbf{e}(\mathbf{x}) = \mathbf{b} - \mathbf{A}\mathbf{x}$, the problem becomes a *linear* least squares problem.

Given a set of measurements $\mathbf{b}$ corrupted by identically distributed noise $\mathbf{v} \sim (\mathbf{0}, \sigma^2\mathbf{1})$ with zero mean and variance σ^2, and a model matrix $\mathbf{A}$ that relates the measurements to the unknown states $\mathbf{x}$, the least squares estimator of $\mathbf{x}$ in the model

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{v}$$

is

$$\hat{\mathbf{x}} = (\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}(\mathbf{A}^\mathsf{T}\mathbf{b}). \tag{2.2}$$

This also turns out to be the best linear unbiased estimator of $\mathbf{x}$ [42].

There are a number of ways to solve for $\hat{\mathbf{x}}$ without the need to compute an expensive matrix inverse. The normal equation method for computing $\hat{\mathbf{x}}$ is as follows.

1. Form $\mathbf{A}^\mathsf{T}\mathbf{A}$ and $\mathbf{A}^\mathsf{T}\mathbf{b}$,
2. Perform the Cholesky factorization $\mathbf{A}^\mathsf{T}\mathbf{A} = \mathbf{L}\mathbf{L}^\mathsf{T}$ where $\mathbf{L}$ is lower triangular,
3. Rewite in the form $\mathbf{L}\underbrace{\mathbf{L}^\mathsf{T}\hat{\mathbf{x}}}_{\mathbf{z}} = \mathbf{A}^\mathsf{T}\mathbf{b}$. From here, solve $\mathbf{L}^\mathsf{T}\hat{\mathbf{x}} = \mathbf{z}$ via back substitution.
4. Solve $\mathbf{L}\mathbf{z} = \mathbf{A}^\mathsf{T}\mathbf{b}$ via forward substitution.

2.8 Taylor-Series Expansion

The Taylor-series expansion of a nonlinear function $f(\mathbf{x})$ about an *operating point* $\mathbf{x}_{op}$ is given by

$$f(\mathbf{x}_{op} + \delta\mathbf{x}) = f(\mathbf{x}_{op}) + \left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_{op}} \delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^\mathsf{T}\left.\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^\mathsf{T}\partial \mathbf{x}}\right|_{\mathbf{x}_{op}} \delta\mathbf{x} + (\text{H. O. T.})$$

where H.O.T. stands for the Higher Order Terms. This is useful in optimization methods that require approximating a nonlinear cost function with a linear one about some operating point.

2.9 Solving Nonlinear Least Squares Problems

Nonlinear least squares problems retain the form

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{e}(\mathbf{x})\|_2^2 \tag{2.3}$$

where the residual $\mathbf{e}(\mathbf{x})$ is nonlinear. One way to solve this problem is via the Gauss-Newton method, that itself can be derived from Newton's method for solving general nonlinear optimization problems.

2.9.1 Newton's Method [2]

Newton's method iteratively approximates a nonlinear cost function, computing the optimal perturbation $\delta\mathbf{x}$ that minimizes the approximated cost function at each step. Suppose an initial guess for the states $\mathbf{x}_{op}$ is available. In each iteration, the cost function at $\mathbf{x} = \mathbf{x}_{op} + \delta\mathbf{x}$

is approximated using the first three terms of the Taylor series expansion. The result is a quadratic cost function in $\delta\mathbf{x}$,

$$J(\mathbf{x}) = J(\mathbf{x}_{op} + \delta\mathbf{x}) \approx J(\mathbf{x}_{op}) + \left(\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)\delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^{\mathsf{T}}\left(\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x}^{\mathsf{T}}\partial \mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)\delta\mathbf{x}$$

Finding the $\delta\mathbf{x}$ that minimizes this approximated cost function is then equivalent to minimizing the original cost function. This can be done by computing the derivative of $J(\mathbf{x}_{op} + \delta\mathbf{x})$ with respect to $\delta\mathbf{x}$ and setting to zero, that is

$$\frac{\partial J(\mathbf{x}_{op} + \delta\mathbf{x})}{\partial \delta\mathbf{x}} = \left(\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right) + \delta\mathbf{x}^{\mathsf{T}}\left(\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x}^{\mathsf{T}}\mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right) = \mathbf{0}.$$

This leads to

$$\left(\frac{\partial^2 J(\mathbf{x})}{\partial \mathbf{x}^{\mathsf{T}}\mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)\delta\mathbf{x} = -\left(\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)^{\mathsf{T}},$$

which is a linear equation. After solving for $\delta\mathbf{x}$ the operating point can be updated via $\mathbf{x}_{op} \leftarrow \mathbf{x}_{op} + \delta\mathbf{x}$. Successive iterations of this algorithm can typically lead to convergence of the solution.

2.9.2 Gauss-Newton Method [2]

The Gauss-Newton method tackles the special case of (2.3) where the cost function is a nonlinear least-squares problem. In such a case, the cost function can be written as

$$J(\mathbf{x}) = \mathbf{e}(\mathbf{x})^{\mathsf{T}}\mathbf{e}(\mathbf{x}) = \mathbf{e}(\mathbf{x}_{op} + \delta\mathbf{x})^{\mathsf{T}}\mathbf{e}(\mathbf{x}_{op} + \delta\mathbf{x}).$$

Applying the Taylor series expansion on $\mathbf{e}(\mathbf{x})$ the cost functon becomes

$$\begin{aligned} J(\mathbf{x}) &= \left(\mathbf{e}(\mathbf{x}_{op}) + \left(\frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)\delta\mathbf{x}\right)^{\mathsf{T}}\left(\mathbf{e}(\mathbf{x}_{op}) + \left(\frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)\delta\mathbf{x}\right) \\ &= \mathbf{e}(\mathbf{x}_{op})^{\mathsf{T}}\mathbf{e}(\mathbf{x}_{op}) + \mathbf{e}(\mathbf{x}_{op})^{\mathsf{T}}\left(\frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)\delta\mathbf{x} + \delta\mathbf{x}^{\mathsf{T}}\left(\frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)^{\mathsf{T}}\mathbf{e}(\mathbf{x}_{op}) \\ &\quad + \delta\mathbf{x}^{\mathsf{T}}\left(\frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)^{\mathsf{T}}\left(\frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)\delta\mathbf{x}. \end{aligned}$$

Minimizing the cost function with respect to $\delta\mathbf{x}$, it follows that

$$\frac{\partial J(\mathbf{x}_{op}+\delta\mathbf{x})}{\partial\delta\mathbf{x}} = 2\mathbf{e}(\mathbf{x}_{op})^{\mathsf{T}}\left(\frac{\partial\mathbf{e}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right) + 2\delta\mathbf{x}^{\mathsf{T}}\left(\frac{\partial\mathbf{e}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)^{\mathsf{T}}\left(\frac{\partial\mathbf{e}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right) = \mathbf{0}.$$

Transposing both sides, the least-squares solution becomes

$$\left(\frac{\partial\mathbf{e}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)^{\mathsf{T}}\left(\frac{\partial\mathbf{e}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)\delta\mathbf{x} = -\left(\frac{\partial\mathbf{e}(\mathbf{x})}{\partial\mathbf{x}}\bigg|_{\mathbf{x}_{op}}\right)^{\mathsf{T}}\mathbf{e}(\mathbf{x}_{op}) \tag{2.4}$$

or simply

$$(\mathbf{H}^{\mathsf{T}}\mathbf{H})\delta\mathbf{x} = -\mathbf{H}^{\mathsf{T}}\mathbf{e}(\mathbf{x}_{op})$$

where $\mathbf{H} = \frac{\partial\mathbf{e}(\mathbf{x})}{\partial\mathbf{x}}\Big|_{\mathbf{x}_{op}}$.

2.9.3 Levenberg-Marquardt Method

The Levenberg-Marquardt method is a more conservative variant of the GN method. In practical applications, certain states are only observed once. This can lead to the system to become ill-conditioned. For this reason, it is often desirable to regularize the linear system with a damping factor λ such that the new system

$$(\mathbf{H}^{\mathsf{T}}\mathbf{H} + \lambda\mathbf{1})\delta\mathbf{x} = -\mathbf{H}^{\mathsf{T}}\mathbf{e}(\mathbf{x}_{op}),$$

is well-conditioned. Put another way, when the damping factor $\lambda >> \|\mathbf{H}\|_{\mathsf{F}}$ where $\|\cdot\|_{\mathsf{F}}$ is the Frobenius norm,

$$\delta\mathbf{x} \approx -\frac{1}{\lambda}\mathbf{H}^{\mathsf{T}}\mathbf{e}(\mathbf{x}_{op}),$$

which corresponds to the gradient-descent method, that is, taking a very small step in the direction of the gradient. When $\lambda << \|\mathbf{H}\|_{\mathsf{F}}$, the standard GN method is recovered. This allows the algorithm to adjust the rate of descent based on how the error term evolves. If at any given time, the new error obtained through incorporating a computed $\delta\mathbf{x}^*$ is larger than the error before incorporating the new $\delta\mathbf{x}^*$, the step taken is likely too large and $\delta\mathbf{x}^*$ is recomputed with a larger λ. On the other hand, if the new error is indeed less than the error before incorporating the new $\delta\mathbf{x}^*$, λ can be decreased to help speed up convergence.

2.9.4 Basic Statistics Concepts

The mean or expected value of a random vector $\mathbf{x}$ with a discrete set of outcomes u_i and associated probabilities p_i is defined as

$$E[\mathbf{x}] = \sum_{i=1}^{\infty} u_i p_i$$

The covariance matrix of a random vector $\mathbf{x}$ is defined as

$$\mathbf{\Sigma}_{\mathbf{x}} = E[(\mathbf{x} - E[\mathbf{x}])(\mathbf{x} - E[\mathbf{x}])^{\mathsf{T}}]$$

When a random state vector $\mathbf{x}_a$ resolved in an arbitrary frame $\mathcal{F}_a$ is rotated such that $\mathbf{x}_b = \mathbf{C}_{ba}\mathbf{x}_a$, the resulting covariance matrix of the same state vector resolved in the new frame $\mathcal{F}_b$, that being $\mathbf{x}_b$, is

$$\begin{aligned}
\mathbf{\Sigma}_{\mathbf{x}_b} &= E[(\bar{\mathbf{x}}_b - \mathbf{x}_b)(\bar{\mathbf{x}}_b - \mathbf{x}_b)^{\mathsf{T}}] \\
&= E[(\mathbf{C}_{ba}\bar{\mathbf{x}}_a - \mathbf{C}_{ba}\mathbf{x}_a)(\mathbf{C}_{ba}\bar{\mathbf{x}}_a - \mathbf{C}\mathbf{x}_a)^{\mathsf{T}}] \\
&= \mathbf{C}_{ba}\mathbf{\Sigma}_{\mathbf{x}_a}\mathbf{C}_{ba}^{\mathsf{T}}
\end{aligned}$$

Chapter 3

Sensor Modelling

To make use of the Weighted OLATE algorithm to be derived in Chapter 5, it is important to characterize the uncertainty of a given sensor. In this chapter, focus will be placed on the modelling of LIDAR sensors, but concepts carry over to other time-of-flight sensors such as certain RGBD cameras.

3.1 Time-of-Flight Sensors and their Sources of Uncertainty

Time-of-flight sensors such as LIDAR estimate distance by emitting a laser and measuring the time of arrival of its reflection. The principal equation is thus

$$\text{distance} = \frac{c \cdot T_{OF}}{2} \tag{3.1}$$

where c is the speed of flight, and T_{OF} is the time of flight [43]. Equation (3.1) indicates that uncertainty in time of flight propagate to the range measurement. The longer the time of flight, the greater the uncertainty in range. In reality, a number of other external factors such as temperature, reflectivity of the surface being measured, surface roughness, and incidence angle can all affect the quality of the measurement. Data taken by Lichti *et al.* [44] suggest that angle of incidence plays the largest role in LIDAR uncertainty, while the other aforementioned factors are unobtainable in practical situations. Because the actual uncertainty of the range component of a LIDAR measurement depend on the electronics and various other internal components and because every sensor is different, it should be in the user's interest to model their own sensors. In the following section, one particular approach is offered.

3.2 2D LIDAR Characterization

The task at hand is to build a model that is representative of measurement variances experienced by a real 2D LIDAR. Specifically, the following experiment will analyze the measurement variance of a RP-LIDAR.

The RP-LIDAR is an inexpensive hobby-grade lidar intended for short-range scan matching and mapping. It consists of a laser mounted on a rotating platform that rotates at roughly 10Hz. Manufacturer data-sheets indicate an operational range between 0.15m and 8m. The angular resolution can be varied between 0.45deg and 1.35deg. Because the manufacturers provide neither range nor bearing uncertainty information, they are modelled as follows. Note that the coordinate frame of the RP-LIDAR is shown in Fig. 3.1.

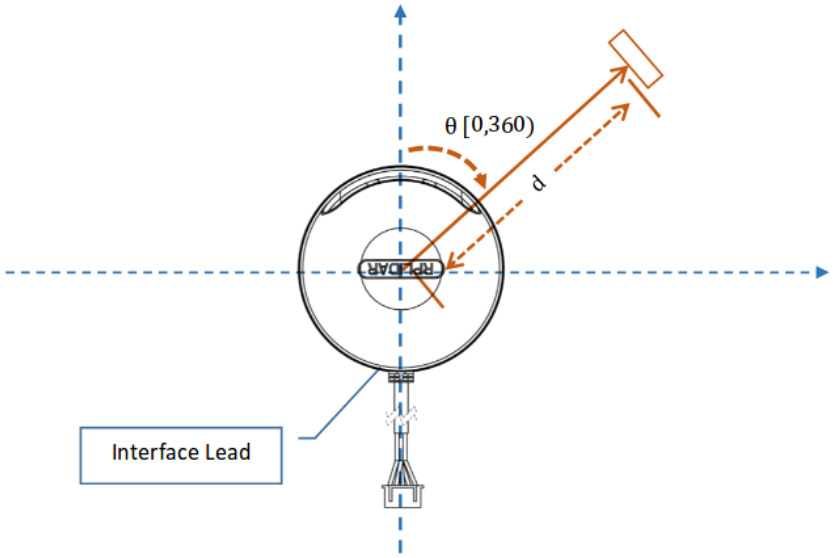
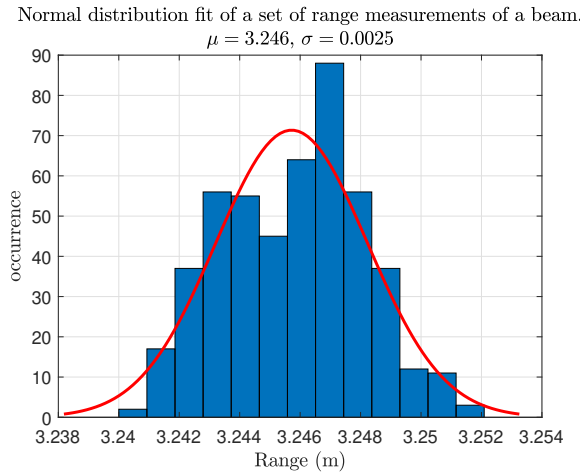


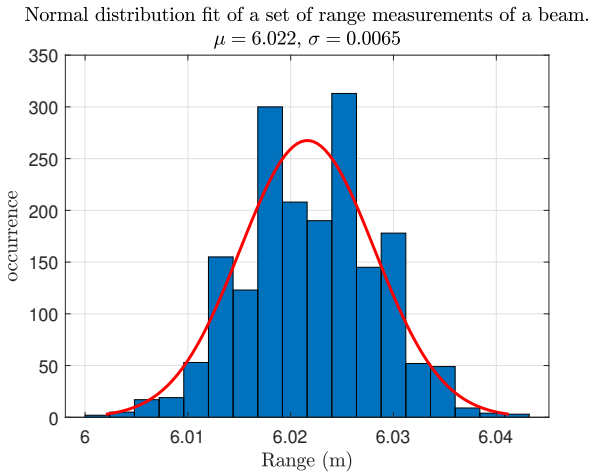
Figure 3.1: RP-LIDAR coordinate frame.

First, range-induced variance in LIDAR range measurements is modelled. This type of variance is affected by the way that time-of-flight measurements are taken. For this experiment, the LIDAR is positioned along a narrow corridor, facing the end of the corridor, and left stationary to collect data for roughly five minutes. This can typically yield 2000 points per LIDAR beam. The data is fit to a Gaussian distribution as shown in Fig. 4.5 and the variance is computed. The LIDAR is then repositioned further away from the end of the corridor and this process is repeated until a sufficient number of ranges between the operational ranges are sampled. This experimental setup is displayed in Fig. 3.3.

Notice from Fig. 3.4a that the error in range measurements increase linearly with distance up until $\approx$ 4m, which is in line with Eq. (3.1). However, error begins to grow following an



(a) ≈ 3m away from wall.



(b) ≈ 6m away from wall.

Figure 3.2: Normally distributed range measurements.

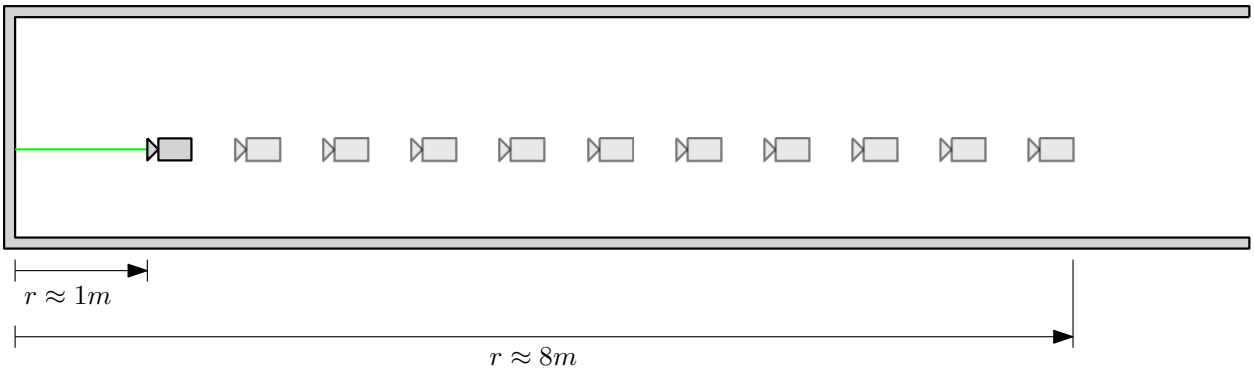
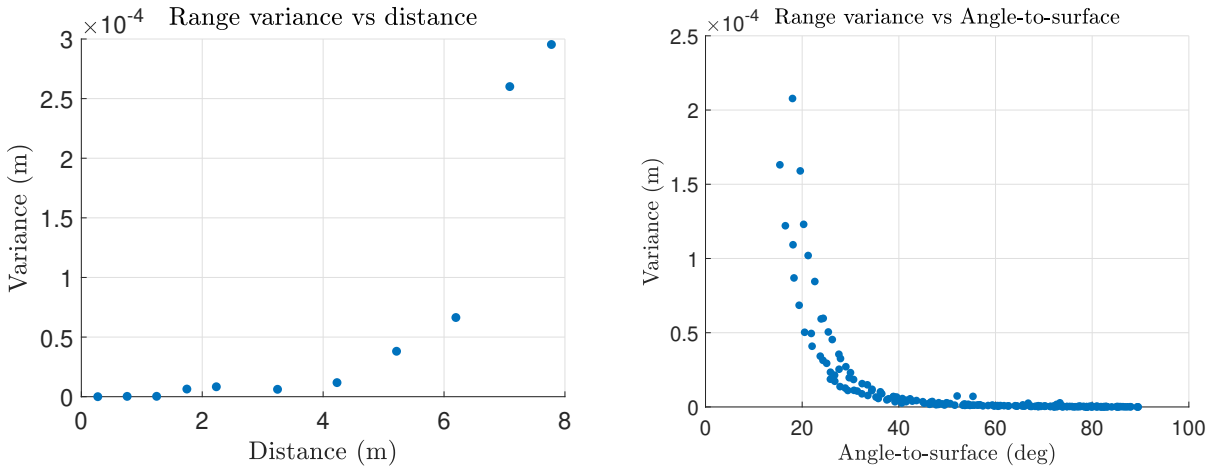


Figure 3.3: Test setup for range variance vs. distance.

exponential function as the distance is further increased, suggesting that other factors are at play.

The other and perhaps largest source of error is the variation in bearing measurements. This variation can be a direct result of the synchronization of measurements, that is, if a certain measurement frequency is required, the LIDAR must time the firing of its laser based on encoder readings that are in turn uncertain.

It is observed that bearing variance directly translates to range variance depending on the incidence angle of observed surfaces. This phenomenon is known as *bearing-induced range uncertainty*. Referring to Fig. 3.5, it can be seen that if the incidence angle ψ is small, bearing uncertainty does not contribute significantly to range uncertainty. As the incidence angle approaches 90deg error in range increases drastically and must be accounted for. In the same experiment, measurements of a section of the wall at different incidence angles are taken at approximately the same ranges. In application, it is more convenient to work with



(a) Range variance vs. distance.

(b) Range variance vs. Angle-to-surface.

Figure 3.4: Relationship of range variance to each independent variable.

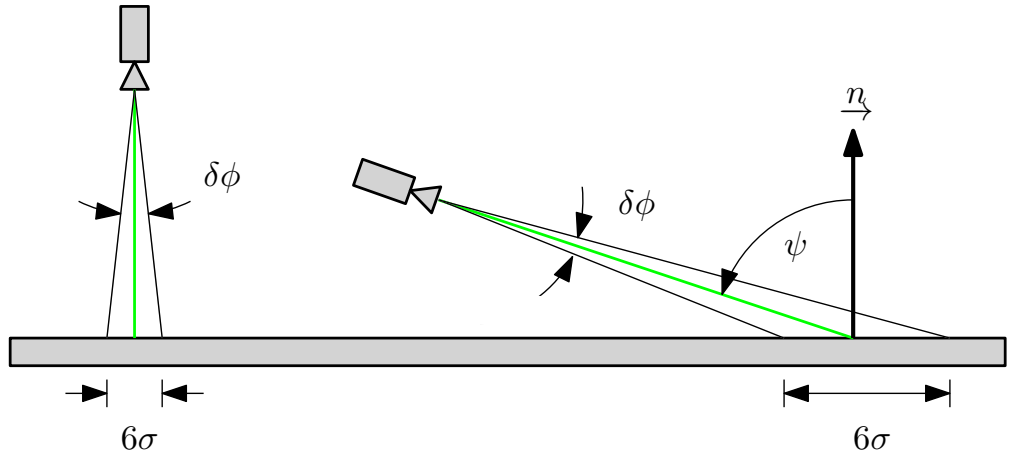


Figure 3.5: Bearing-induced uncertainty. The same bearing variance has a larger effect on range variance at larger incidence angles.

$\theta = \frac{\pi}{2} - \psi$, which is the angle between the laser beam and the surface. The relationship between variance and angle-to-surface θ is an inverse exponential shown in Fig. 3.4b.

When plotting range variance as a function of both range and angle-to-surface as seen in Fig. 3.6, an additional relation between the two variables is noticed. At smaller ranges, variance decreases quickly with decreasing angle-to-surface. At larger ranges, variance decreases at a much slower rate with decreasing angle-to-surface. It can be hypothesized that this inverse relation is of the form

$$\sigma^2 = a\left(\frac{r}{\sin(\theta)}\right)^b$$

where $r \in \mathbb{R}^+$ is range and $\theta \in [0, \frac{\pi}{2}]$ is the angle between the laser beam and the surface. In other words, as angle-to-surface approaches 0deg, bearing-induced range variance is magnified whereas if the angle-to-surface is large, bearing-induced range variance is minimal.

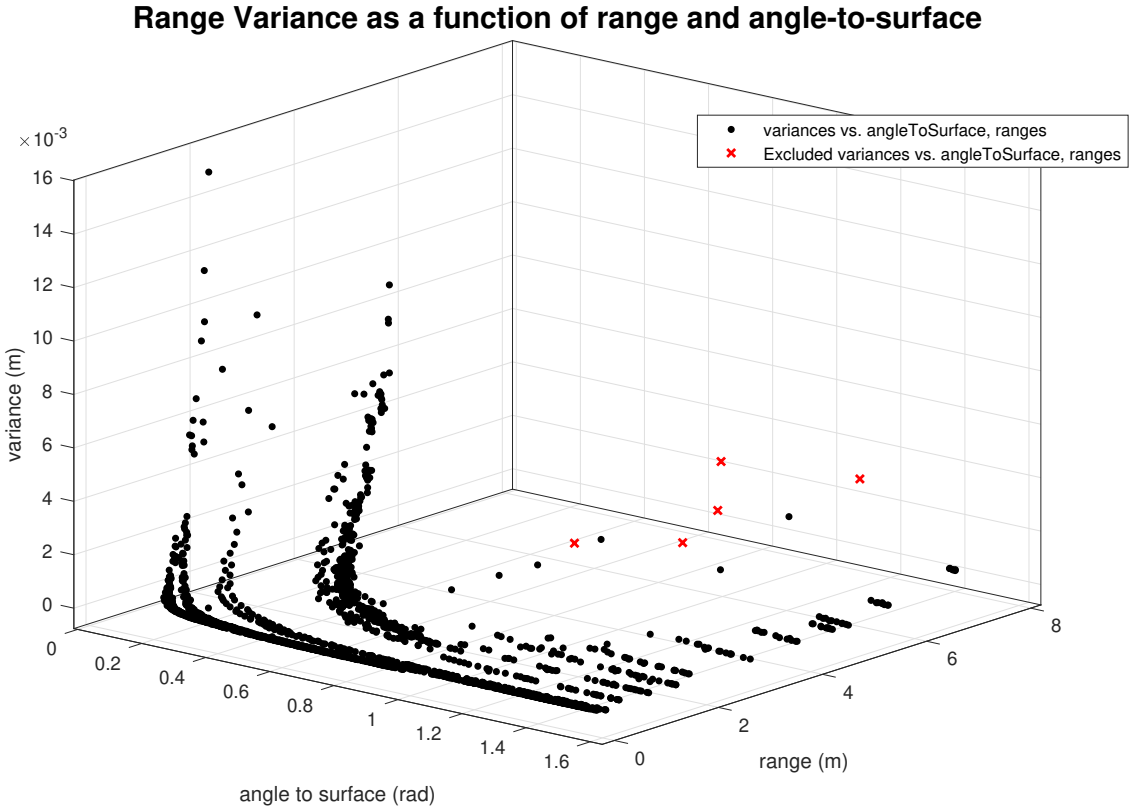


Figure 3.6: Range variance as a function of both range and angle-to-surface.

Fitting this model to the data of Fig. 3.6 using nonlinear least-squares optimization yields a and b values of

$$a = 2.277 \times 10^{-5}, \quad b = 1.841.$$

The model describing range variance as a function of range and angle-to-surface is thus

$$\sigma^2 = 2.277 \times 10^{-5} \left(\frac{r}{\sin(\theta)} \right)^{1.841},$$

and can be visualized in Fig. 3.7. This model will be directly used in Chapter 5 to inject noise of realistic magnitudes into simulated LIDAR data.

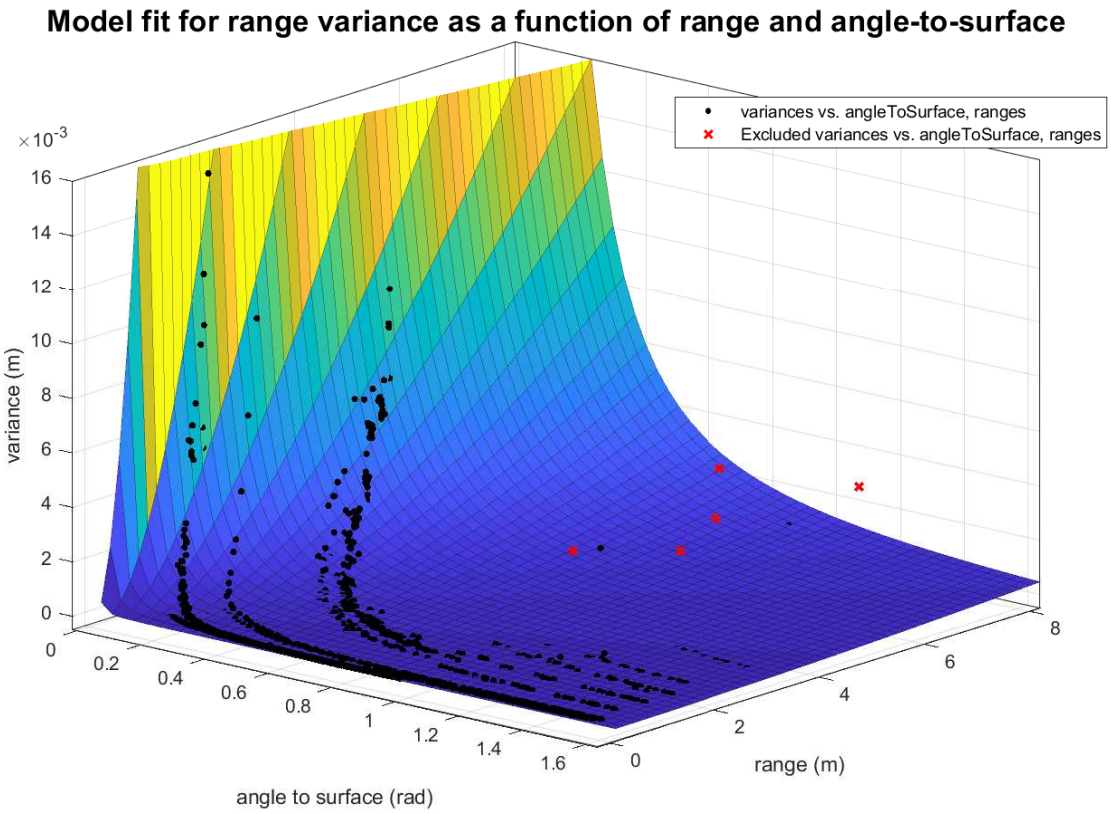


Figure 3.7: Model of range variance as a function of both range and angle-to-surface.

Chapter 4

Local SLAM Strategy - Data Pre-Processing

Local SLAM, as the term suggests, deals with solving the SLAM problem locally in both a spatial and temporal sense. Local SLAM encompasses methods such as *dead-reckoning* - integrating acceleration and angular velocity, *wheel odometry* - tracking wheel revolutions and steering angles, *visual odometry* - matching subsequent images or projections thereof, and *scan-matching* - matching subsequent LIDAR scans, to obtain the transformation between successive poses. The underlying assumption is that Local SLAM is locally consistent with minimal drift. This assumption allows it to be decoupled from the Global SLAM problem and solved separately.

In this thesis, focus will be placed on methods of dead-reckoning and Scan-Matching to obtain pose-to-pose transformations. Before introducing Scan-Matching in detail, it is important to understand that the quality and type of the data given to the Scan Matching step often determines what kind of Scan Matching algorithm to use. As such, in this chapter, an overview of the commonly used pre-processing steps required for effective Scan Matching will be offered.

4.1 Processing and Unwarping of LIDAR Data

As mentioned in Chapter 3, LIDAR data is measured in terms of range measurements taken at known times with a constant angle between successive points. If the range is infinite or greater than the hardware's max range, a default value is returned. These points are often classified as "misses". The remaining points that fall within the hardware's operating range are classified as "hits".

The next detail to note is that the points of a LIDAR sweep are never taken simultaneously. This is because LIDAR points are derived from a rotating beam that, upon completing a 360deg revolution, form a "sweep". This means that if a robot carrying a LIDAR is in motion, the resulting point-cloud of a LIDAR scan will be distorted or warped if resolved in the sensor frame. This is especially true if the LIDAR in question has a slow sampling rate. However, scan matching must occur in the vehicle's body frame in order to infer the relative transformation between two scans. It is therefore imperative that this distortion be corrected and projected back into the vehicle's body frame before carrying on with scan matching. To simplify the situation, first assume that the sensor frame and the vehicle's body frame are the same. Given a distorted set of points $\mathbf{r}_{s_k}^{p_k s_k}$ where $k = 1, \ldots, N$, a common way to extract the real scene points is as follows.

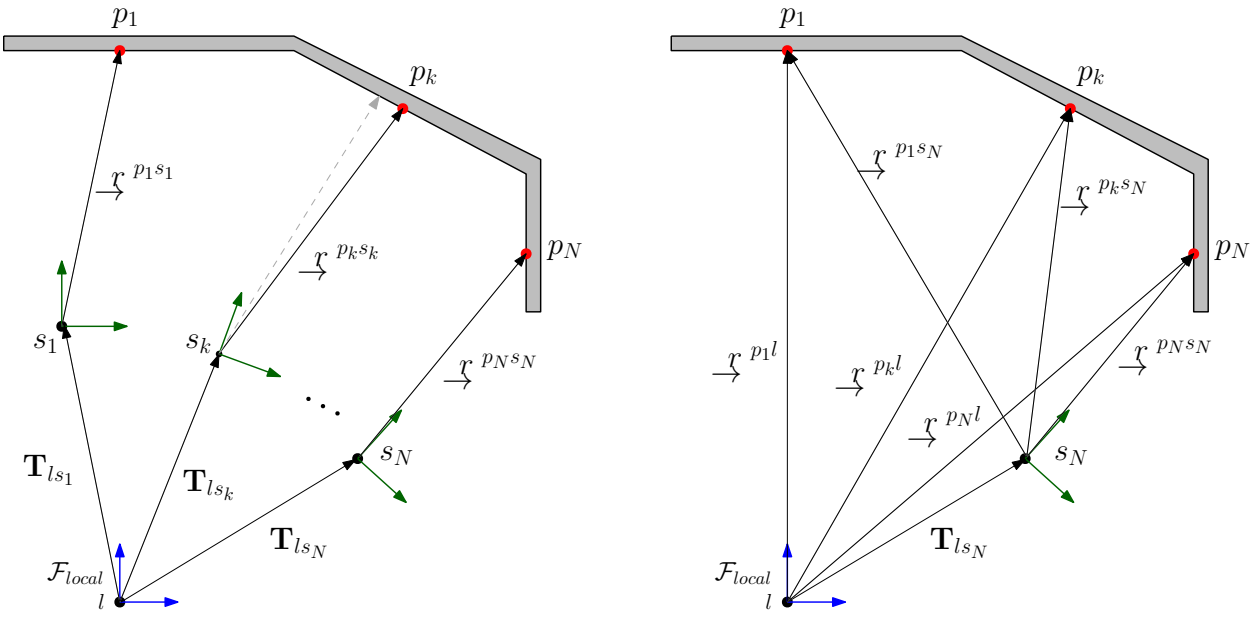


Figure 4.1: Range data accumulation process. Data is taken from the sensor frame at different poses.

Referring to Fig. 4.1, assume that the array of poses $\mathbf{T}_{ls_k}$ is known, where hits in the sensor frame and in the local frame are defined as

$$\text{hit} = \mathbf{r}_{s_k}^{p_k s_k},$$
$$\text{hit in local frame} = \mathbf{r}_{l}^{p_k l} = \mathbf{T}_{ls_k} \mathbf{r}_{s_k}^{p_k s_k}.$$

In this case, the local frame is any arbitrary datum frame that is stationary with respect to the sensor frames. In practice, it can be initialized at the first sensor frame. Note that at this stage, $\mathbf{r}_{l}^{p_k l}$ contains the corrected LIDAR points in the local frame. Since scan matching

must be applied to the vehicle's body frame, or in this case, the sensor frame, project $\mathbf{r}_l^{p_k l}$ back into the last sensor frame of the current sweep, yielding $\mathbf{r}_{s_N}^{p_k s_N}$.

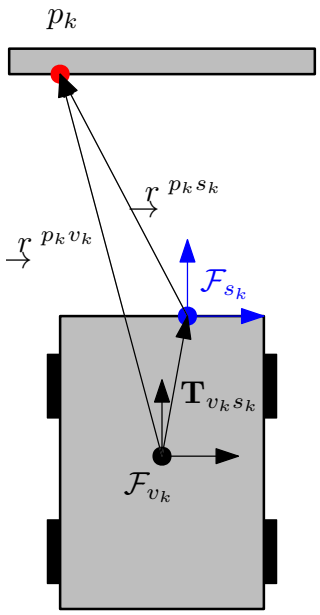


Figure 4.2: Sensor frame being offset from vehicle's body frame.

4.2 Sensor Level Arm

On a robotic system such as that of Fig. 4.2, the LIDAR sensor is usually offset from the vehicle's base frame, which is typically the IMU frame. This transformation, namely $\mathbf{T}_{v_k s_k}$, also known as the sensor level arm, must be taken into account when computing the positions of measured points since it is ultimately $\mathbf{r}_{v_k}^{p_k v_k}$ that is sought.

Assume now that the level arm $\mathbf{T}_{v_k s_k}$ and the array of vehicle poses $\mathbf{T}_{l v_k}$ is known. The position of the LIDAR points resolved in the local frame is simply

$$\begin{aligned} \mathbf{r}_{v_k}^{p_k v_k} &= \mathbf{T}_{v_k s_k} \mathbf{r}_{s_k}^{p_k s_k} \\ \mathbf{r}_{l}^{p_k l} &= \mathbf{T}_{l v_k} \mathbf{r}_{v_k}^{p_k v_k} \end{aligned}$$

and the same points resolved in the last sensor frame of the current sweep is

$$\mathbf{r}_{v_N}^{p_k v_N} = \mathbf{T}_{l v_N}^{-1} \mathbf{r}_{l}^{p_k l}.$$

4.3 Additional Considerations

The array of poses $\mathbf{T}_{lv_k}$ is not easy to obtain but given an IMU with a fast enough sampling rate, it is possible to estimate it via dead-reckoning. Another way to estimate $\mathbf{T}_{lv}$ is to assume constant motion, that is, $\mathbf{v}_i^{v_k v_{k-1}/i} = \mathbf{v}_i^{v_{k-1} v_{k-2}/i}$ where

$$\mathbf{v}_i^{v_{k-1} v_{k-2}/i} = \frac{\mathbf{r}_i^{v_{k-2} i} - \mathbf{r}_i^{v_{k-1} i}}{T}.$$

Doing so allows the relative pose-to-pose translation $\delta \mathbf{r}_{v_k}^{v_k v_{k-1}}$ to be simply computed with the velocity estimate given $T = t_k - t_{k-1}$. Throughout the rest of this thesis, whenever LIDAR odometry results are presented, it is implied that any given LIDAR sweep is already unwarped via a similar procedure where $\mathbf{T}_{lv_k}$ are assumed to be obtained from either an accurate INS, RTK, or high-rate IMU.

Additionally, the relative orientation change between the two frames, $\mathbf{C}_{v_{k-1} v_k}$, is sought. To this end,

$$\begin{aligned} \boldsymbol{\psi}_{k-1} &= \boldsymbol{\omega}_{v_{k-1}}^{v_{k-1} a} T \\ \delta \mathbf{C}_{v_{k-1} v_k} &= e^{\boldsymbol{\psi}_{k-1}^{\times}} \end{aligned}$$

where

$$e^{\boldsymbol{\psi}_{k-1}^{\times}} = \cos \boldsymbol{\psi}_{k-1} \mathbf{1} + (1 - \cos \boldsymbol{\psi}_{k-1}) \left(\frac{\boldsymbol{\psi}_{k-1}}{\psi_{k-1}} \right) \left(\frac{\boldsymbol{\psi}_{k-1}}{\psi_{k-1}} \right)^{\mathsf{T}} + \sin \boldsymbol{\psi}_{k-1} \left(\frac{\boldsymbol{\psi}_{k-1}}{\psi_{k-1}} \right)^{\times}$$

represents the small rotation between the previous orientation and the new orientation as expected from the most recent angular velocity measurement.

4.4 Projecting LIDAR Data to a Common Frame

For 2D scan matching on flat terrain such as the floor of a building, all scans are taken at the same pitch and roll, with only yaw varying in time. However, when the terrain is uneven, each scan is taken at a different orientation. To get around this problem without actually estimating pitch and roll, it is common practice to project every scan to a common reference frame before scan matching. The gravity-aligned frame is often chosen to be the common reference frame for scan projection because an onboard IMU can measure the gravity vector directly and because it does not suffer from drift when the robot is stationary. The gravity orientation $\mathbf{C}_{gv_k}$ can be computed as follows.

Given a prior gravity vector $\mathbf{v}_{v_{k-1}}^{gv_{k-1}}$ the new gravity vector can be computed via

$$\mathbf{v}_{v_k}^{gv_k} = \delta\mathbf{C}_{v_k v_{k-1}}\mathbf{v}_{v_{k-1}}^{gv_{k-1}},$$

and the transform between the gravity-aligned and the vehicle frame can be computer via

$$\mathbf{T}_{gv_k} = \begin{bmatrix} \mathbf{C}_{gv_k} & \mathbf{r}_g^{v_k g} \\ \mathbf{0} & 1 \end{bmatrix}.$$

The LIDAR points resolved in the local frame can now be transformed to a new gravity aligned frame via

$$\mathbf{T}_{gl} = \mathbf{T}_{gv_k}\mathbf{T}_{lv_k}{}^{-1} = \mathbf{T}_{gv_k}\mathbf{T}_{v_k l}$$
$$\mathbf{r}_g^{p_k g} = \mathbf{T}_{gl}\mathbf{r}_l^{p_k l}$$

4.5 Predicting Pose-to-Pose Transformations for Scan Matching

Scan matching is how pose-to-pose constraints are formed. Scan matching requires solving an optimization problem. Before performing scan matching, any scan matching algorithm requires a good prediction of the relative transformation between the two scans to initialize the optimization problem. A good prediction of this transformation can be obtained if one has access to an IMU, INS, or RTK.

4.6 Feature Extraction from LIDAR Data

The term feature extraction is often attributed to computer vision methods where high gradient pixels such as corners or edges are detected and tracked through subsequent frames for motion estimation. This technique can also be extended to LIDAR scans. LIDAR point clouds differ from point clouds obtained via stereo cameras because they sample the world with a rotating laser beam. This means that they have a 360 degree Field Of View (FOV) on a single plane. In a 3D LIDAR, many lasers are arranged along the axial direction on an arc/fan such that when rotated, many scan planes are generated, giving a 3D view of the scene. Because the resolution along the direction of rotation is much higher than the resolution between scan planes, the resulting point cloud is sparse in the axial direction but dense radially, producing a series of scan rings as seen in Fig. 4.3. The high resolution of each scan ring allows one to extract features from coplanar points. One method to accomplish this is by following the approach described in [22] where the local curvature of each point is

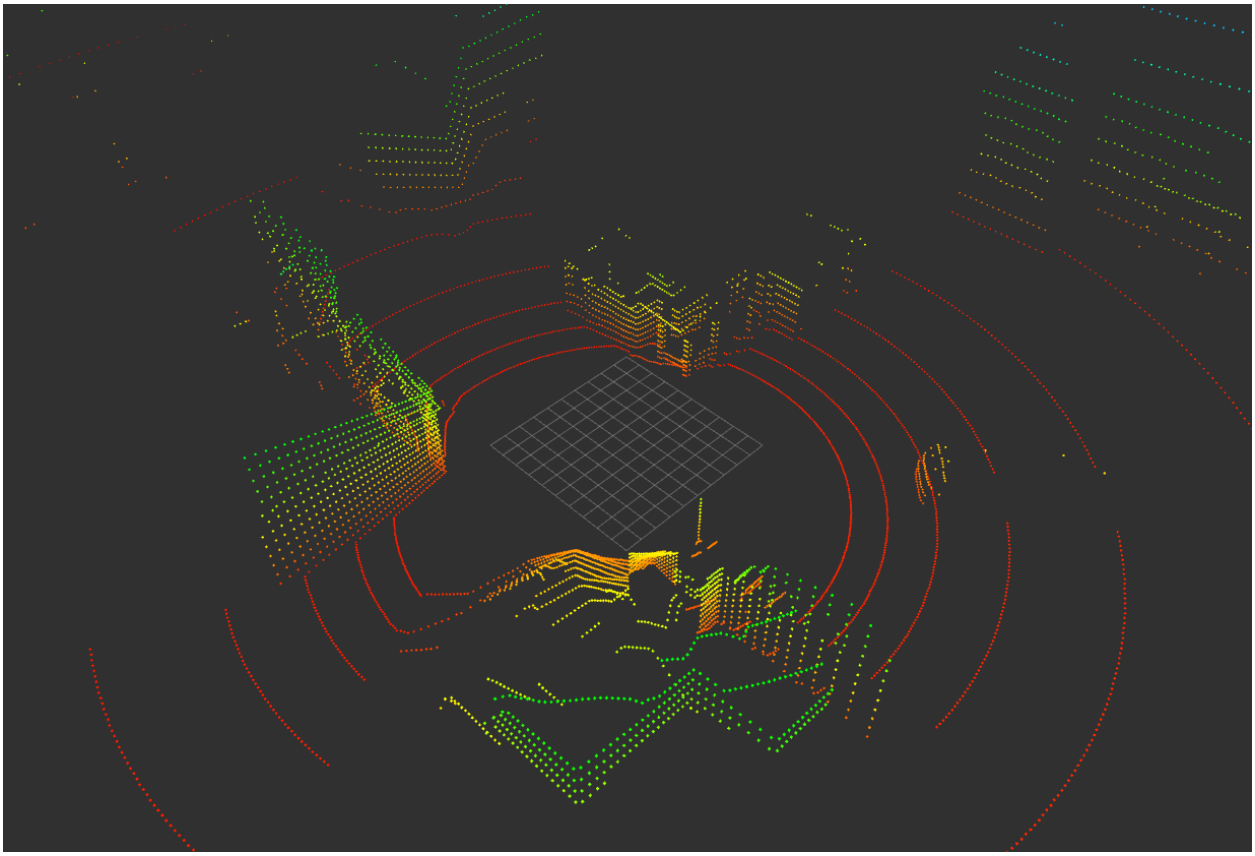


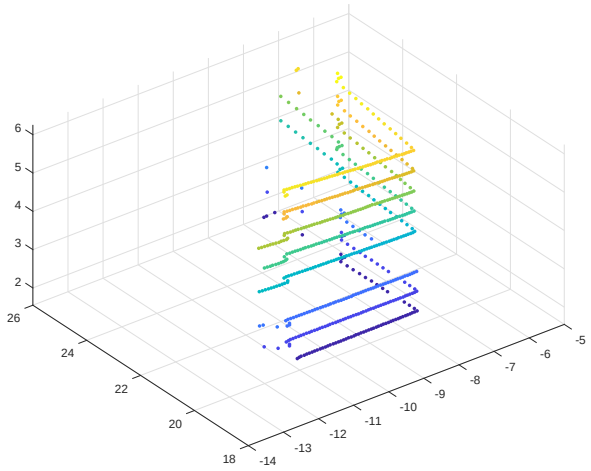
Figure 4.3: A typical point cloud obtained from a single sweep of a Velodyne LIDAR simulated in Gazebo.

computed based on nearby points via

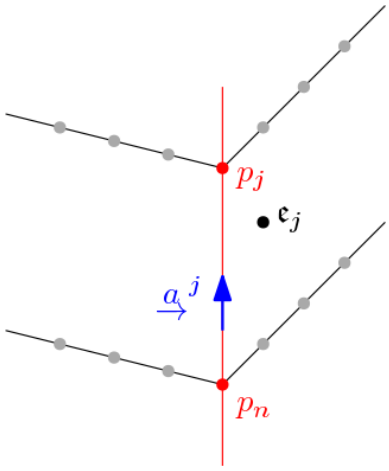
$$s = \frac{1}{N_p \cdot \|\mathbf{r}_{s_k}^{p_i s_k}\|} \left\| \sum_{j=i-N_p, j\neq i}^{i+N_p} \mathbf{r}_{s_k}^{p_i s_k} - \mathbf{r}_{s_k}^{p_j s_k} \right\|,$$

where N_p is the number of nearby points used to evaluate point curvature. However, this method does not account for cases where the sum of points with varying spread along a line result in a large vector in the direction of that line. An alternative is therefore proposed where Principle Component Analysis (PCA) is used to compute local curvature. Refer to Appendix A for a derivation of PCA. Specifically, when computing the covariance matrix of a cluster of points p_j with respect to their centroid, normalize each error residual such that

$$\mathbf{\Sigma} = \frac{1}{N-1} \sum_{j=1}^{N} \left(\frac{\mathbf{r}_a^{p_j a} - \bar{\mathbf{r}}_a}{\left\| \mathbf{r}_a^{p_j a} - \bar{\mathbf{r}}_a \right\|} \right) \left(\frac{\mathbf{r}_a^{p_j a} - \bar{\mathbf{r}}_a}{\left\| \mathbf{r}_a^{p_j a} - \bar{\mathbf{r}}_a \right\|} \right)^{\mathsf{T}}$$



(a) An edge as observed from a Velodyne LIDAR.



(b) Corresponding a corner point $\mathfrak{e}_j$ to its closest edge.

Figure 4.4: Corner points and edge geometry.

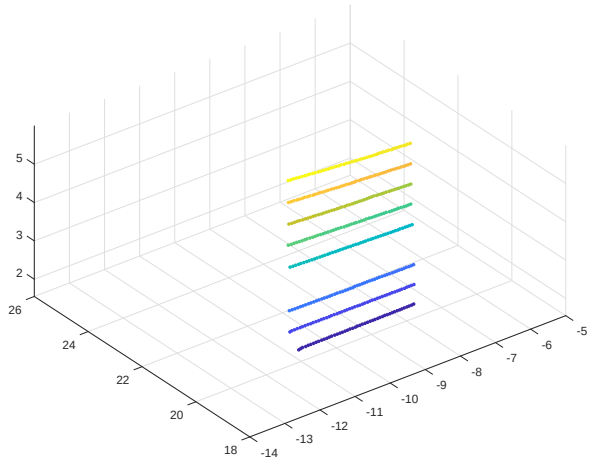
where $\bar{\mathbf{r}}_a$ is the centroid, that is, $\bar{\mathbf{r}}_a = \frac{1}{N}\sum_{j=1}^{N} \mathbf{r}_a^{p_j a}$ and where a is an arbitrary point with associated frame $\mathcal{F}_a$. Once the principle components are extracted, the eigenvector corresponding to the smallest eigenvalue is the normal direction of the line and the curvature can be represented by

$$s = \frac{\lambda_{min}}{\lambda_{min} + \lambda_{max}}.$$

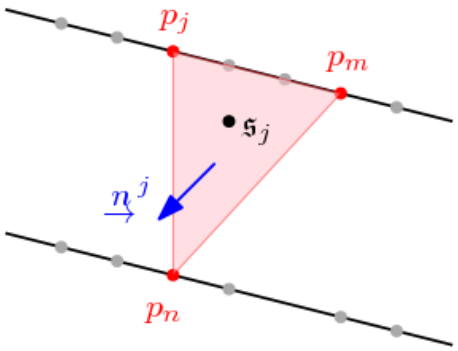
Points with a high s value corresponds to regions of high curvature such as corners whereas points with a low s value correspond to regions of low curvature such as flat surfaces. Sorting these curvature values allows the extraction of feature points from a single scan ring.

When looking at multiple scan rings in 3D, each ring having their own set of corner and surface points, it then becomes possible to extract edges or planes. For example, sometimes it is of interest to correspond an extracted corner point with the nearest edge. Referring to Fig. 4.4b, suppose that point $\mathfrak{e}_k^j$ is a corner point measured in $\mathcal{F}_{v_k}$. Let p_j be the nearest corner point from the closest scan ring and let p_n be the nearest corner point from the second closest scan ring in the point cloud taken at $\mathcal{F}_{v_{k-1}}$. The unit vector $\underrightarrow{a}^j$ of the line or edge formed by these two points that are closest to edge point $\mathfrak{e}_j$ is simply

$$\underrightarrow{a}^j = \frac{\underrightarrow{r}^{p_j s_k} - \underrightarrow{r}^{p_n s_k}}{\left\| \underrightarrow{r}^{p_j s_k} - \underrightarrow{r}^{p_n s_k} \right\|}.$$



(a) A flat surface as observed from a Velodyne LIDAR.



(b) Corresponding a surface point $\mathfrak{s}^j$ to its closest surface.

Figure 4.5: Surface points and plane geometry.

Other times, it is of interest to correspond an extracted surface point with the nearest surface. Referring to Fig. 4.5b, suppose that point $\mathfrak{s}_j$ is a surface point measured in $\mathcal{F}_{v_k}$. Again, let p_j be the nearest surface point from the closest scan ring and let p_m be the second nearest surface point from the same scan ring in the point cloud taken at $\mathcal{F}_{v_{k-1}}$. Also let p_n be the closest surface point from the second closest scan ring. The plane with unit normal $\underrightarrow{n}^j$ formed by these three points that are closest to surface point $\mathfrak{s}_j$ is simply

$$\underrightarrow{n}^j = \frac{(\underrightarrow{r}^{p_m s_k} - \underrightarrow{r}^{p_j s_k}) \times (\underrightarrow{r}^{p_n s_k} - \underrightarrow{r}^{p_j s_k})}{\left\| (\underrightarrow{r}^{p_m s_k} - \underrightarrow{r}^{p_j s_k}) \times (\underrightarrow{r}^{p_n s_k} - \underrightarrow{r}^{p_j s_k}) \right\|}.$$

In Chapter 5, these unit vectors will be used to define point-to-line and point-to-plane distances.

An additional consideration to be made before feature extraction is which areas of a scan should be avoided altogether. As mentioned in Chapter 3, surfaces that form a small angle with the LIDAR ray tend to produce unreliable measurements. Additionally, measurements that are occluded by objects in the foreground are also unreliable as regular points can appear as corners as seen in Fig. 4.6. To filter out unreliable measurements that fall in these two categories, compute the angle $\delta\theta$ formed between $\mathbf{r}_s^{p_j s}$ and $\mathbf{r}_s^{p_j p_{j+1}}$ as seen in Fig. 4.7. Points that surpasses a certain threshold are considered unreliable. Finally, it is also desirable to sample the environment uniformly and a simple solution is offered in [22] where points near

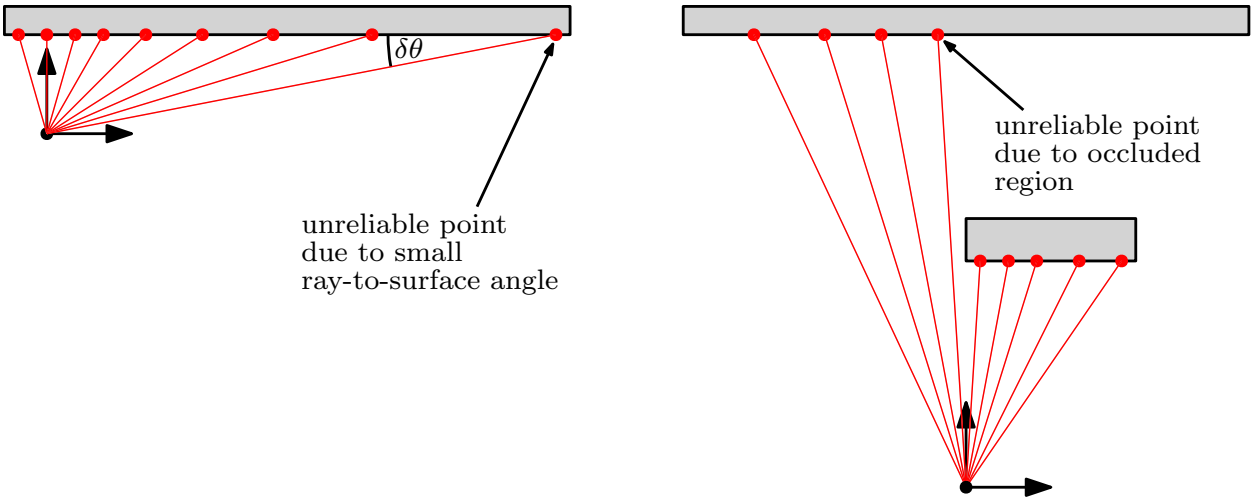


Figure 4.6: Unreliable points to be filtered out prior to feature extraction.

an already selected feature point are disregarded.

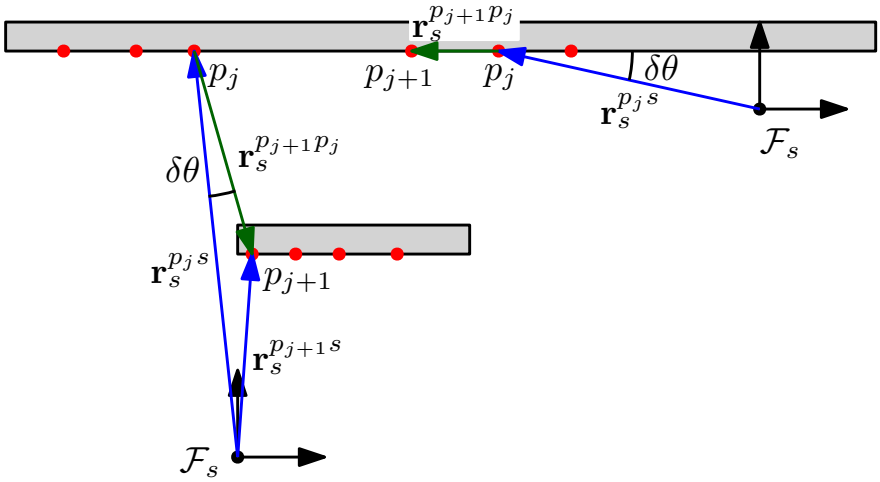


Figure 4.7: Process of filtering unreliable points. Compute $\delta\theta$ and check if it surpasses a threshold.

Figures 4.8 and 4.9 show results of this feature extraction tactic applied to simulated 2D and 3D data respectively. Note that this method works best in urban environments that have an abundance of edges and flat surfaces. Figure 4.10 shows that feature extraction, when performed on such an urban environment, can find many edges and surfaces.

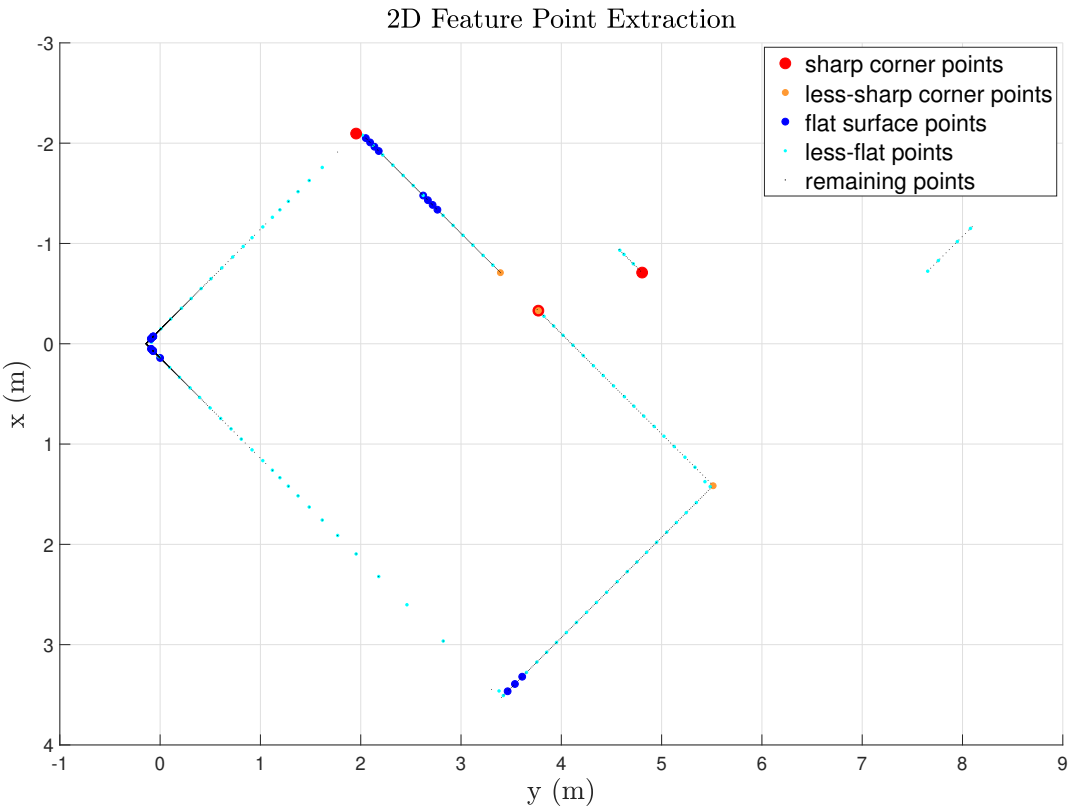


Figure 4.8: Feature points extracted from a 2D LIDAR sweep of a simulated room.

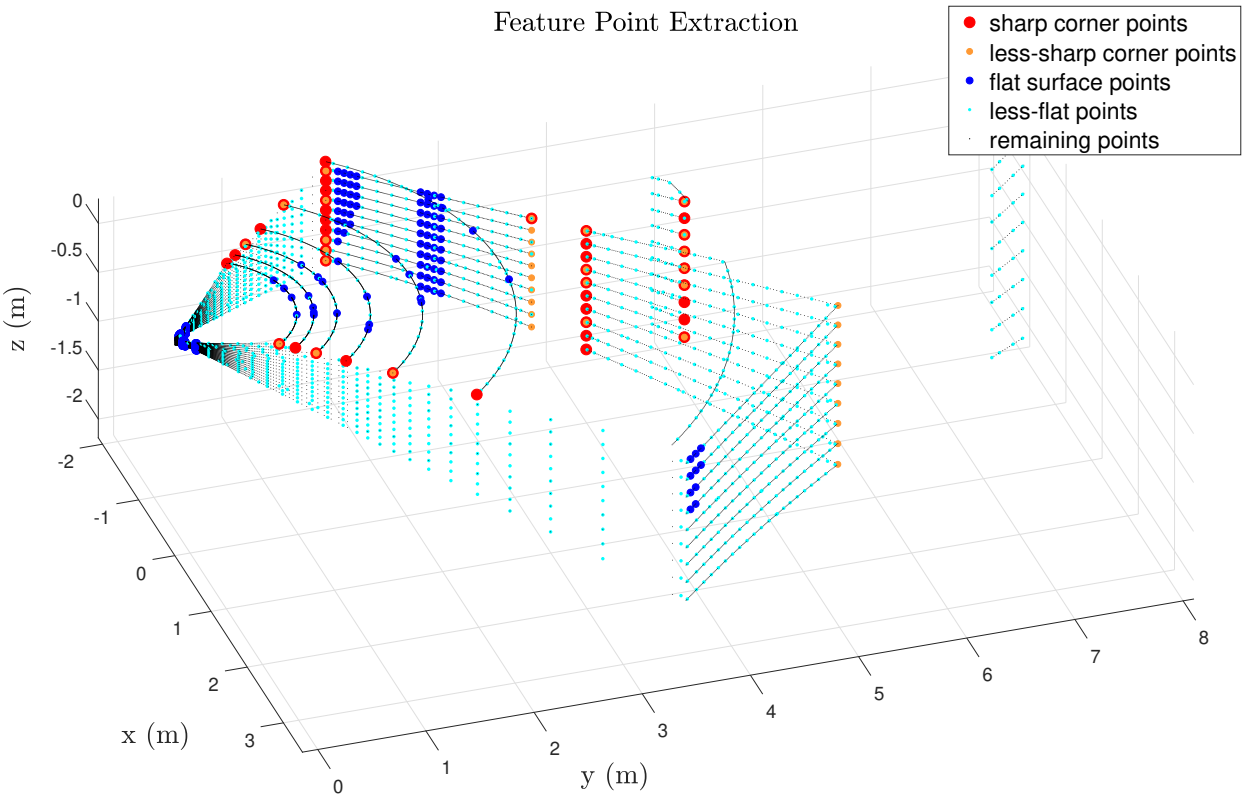


Figure 4.9: Feature points extracted from a 3D LIDAR sweep of a simulated room.

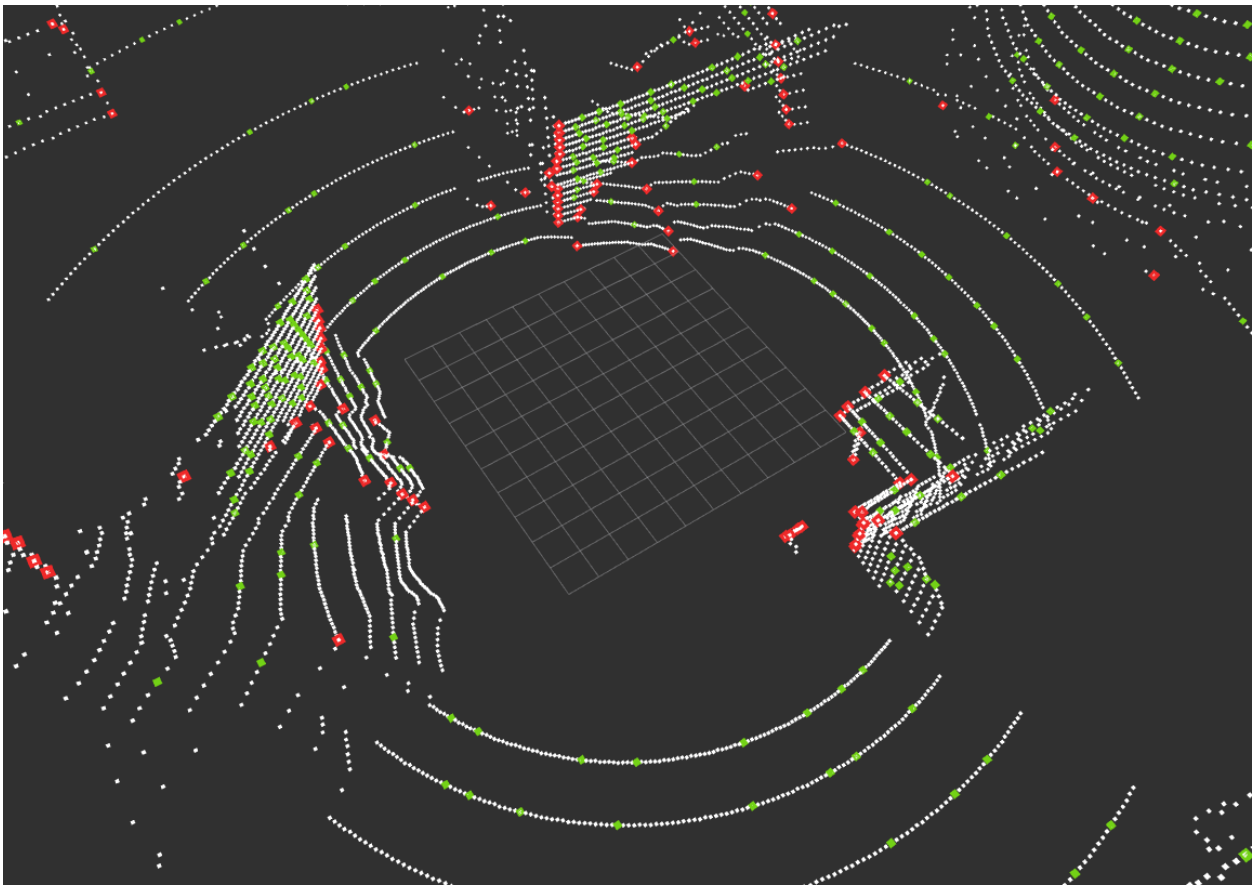


Figure 4.10: Feature points extracted from a 3D LIDAR sweep of a simulated urban area. Red points are sharp corners. Green points are flat surfaces.

4.7 Downsampling Point Clouds

Downsampling incoming point cloud data is a widely applied strategy, especially considering that some 3D LIDAR can measure up to half a million points per second. A simple yet effective method is Voxel Filtering, which involves dividing the subspace into 3D volumes known as voxels of a set dimension and averaging every point found in each voxel. Because some maps require a resolution of at least $1cm$, it becomes evident that a brute force sorting strategy is impractical. Instead, tree data structure, such as Quadtree for 2D or Octree for 3D is popularly used to sort and downsample point clouds. Surface normal information can be averaged as well, though averaging covariance matrices is not as straight forward.

Consider that there are two point measurements,

$$\tilde{\mathbf{r}}_a^{pa} = \mathbf{r}_a^{pa} + \mathbf{v}_a, \quad \text{and} \quad \tilde{\mathbf{r}}_b^{pb} = \mathbf{r}_b^{pb} + \mathbf{v}_b$$

where $\mathbf{v}$ is noise and where a and b are some arbitrary points with associated reference frames $\mathcal{F}_a$ and $\mathcal{F}_b$ respectively. The residual of each measurement is simply equal to the noise. In other words,

$$\mathbf{e}_a = \tilde{\mathbf{r}}_a^{pa} - \mathbf{r}_a^{pa} = \mathbf{v}_a, \quad \text{and} \quad \mathbf{e}_b = \tilde{\mathbf{r}}_b^{pb} - \mathbf{r}_b^{pb} = \mathbf{v}_b.$$

The covariance matrix of each measurement is thus

$$\mathbf{\Sigma}_a = E[\mathbf{e}_a \mathbf{e}_a^\mathsf{T}], \quad \text{and} \quad \mathbf{\Sigma}_b = E[\mathbf{e}_b \mathbf{e}_b^\mathsf{T}].$$

The average of the two measurements can then be computed as

$$\underbrace{\frac{\tilde{\mathbf{r}}_a^{pa} + \tilde{\mathbf{r}}_b^{pb}}{2}}_{\tilde{\mathbf{r}}_{avg}} = \underbrace{\frac{\mathbf{r}_a^{pa} + \mathbf{r}_b^{pb}}{2}}_{\mathbf{r}_{avg}} + \underbrace{\frac{\mathbf{e}_a + \mathbf{e}_b}{2}}_{\mathbf{e}_{avg}}.$$

It follows that the covariance matrix of the averaged measurement is

$$\begin{aligned}
\mathbf{\Sigma}_{avg} &= E[\mathbf{e}_{avg} \mathbf{e}_{avg}^\mathsf{T}] \\
&= E[\frac{1}{4}(\mathbf{e}_a + \mathbf{e}_b)(\mathbf{e}_a + \mathbf{e}_b)^\mathsf{T}] \\
&= \frac{1}{4}(E[\mathbf{e}_a \mathbf{e}_a^\mathsf{T}] + \cancel{E[\mathbf{e}_a \mathbf{e}_b^\mathsf{T}]} + \cancel{E[\mathbf{e}_b \mathbf{e}_a^\mathsf{T}]} + E[\mathbf{e}_b \mathbf{e}_b^\mathsf{T}]) \\
&= \frac{1}{4}\mathbf{\Sigma}_a + \frac{1}{4}\mathbf{\Sigma}_b.
\end{aligned}$$

It has been assumed that the two measurements are uncorrelated, that is, $E[\mathbf{e}_a\mathbf{e}_b^\mathsf{T}] = E[\mathbf{e}_b\mathbf{e}_a^\mathsf{T}] = \mathbf{0}$. From here, generalizing for N points,

$$\mathbf{\Sigma}_{avg} = \frac{1}{N^2}\sum_{i=1}^{N}\mathbf{R}_i. \tag{4.1}$$

4.8 Propagation of Uncertainties

Before continuing, it is worth noting that in practice, covariance matrices do not come in the desired x, y, and z components but rather, in spherical coordinates corresponding to a LIDAR's range, azimuth, and elevation, measurements, denoted r, α, and ψ respectively. To convert covariance matrices to the desired Cartesian coordinates, first note that

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = f(r,\alpha,\psi) = \begin{bmatrix} r\cos\psi\sin\alpha \\ r\cos\psi\cos\alpha \\ r\sin\psi \end{bmatrix}.$$

Let $\mathbf{x}_1 = \begin{bmatrix} r & \alpha & \psi \end{bmatrix}^\mathsf{T}$ and let $f(\mathbf{x}_1) = \mathbf{x}_2 = \begin{bmatrix} x & y & z \end{bmatrix}^\mathsf{T}$. The mapping of $\mathbf{x}_1$ to $\mathbf{x}_2$ can be approximated using a Taylor series expansion about an equilibrium point $\bar{\mathbf{x}}_1$ such that

$$\mathbf{x}_2 \approx f(\bar{\mathbf{x}}_1) + \left.\frac{\partial f(\mathbf{x}_1)}{\partial \mathbf{x}_1}\right|_{\bar{\mathbf{x}}_1}(\mathbf{x}-\mathbf{x}_1).$$

Given a point $\mathbf{x} = \bar{\mathbf{x}}_1 + \mathbf{v}_1$, the resulting propagated mean and noise is

$$\begin{aligned} \bar{\mathbf{x}}_2 &= f(\bar{\mathbf{x}}_1) \\ \mathbf{v}_2 &= \underbrace{\left.\frac{\partial f(\mathbf{x}_1)}{\partial \mathbf{x}_1}\right|_{\bar{\mathbf{x}}_1}}_{\mathbf{F}}\mathbf{v}_1. \end{aligned}$$

Noting that $\mathbf{e}_1 = \mathbf{x} - \bar{\mathbf{x}}_1 = \mathbf{v}_1$, the propagated covariance matrix is thus

$$\begin{aligned} \mathbf{\Sigma}_2 &= E[\mathbf{e}_1\mathbf{e}_1^\mathsf{T}] \\ &= E[(\mathbf{F}\mathbf{v}_1)(\mathbf{F}\mathbf{v}_1)^\mathsf{T}] \\ &= \mathbf{F}E[\mathbf{v}_1\mathbf{v}_1^\mathsf{T}]\mathbf{F}^\mathsf{T} \\ &= \mathbf{F}\mathbf{\Sigma}_1\mathbf{F}^\mathsf{T} \end{aligned} \tag{4.2}$$

where

$$\mathbf{F} = \begin{bmatrix} \cos\psi\sin\alpha & r\cos\psi\cos\alpha & -r\sin\psi\cos\alpha \\ \cos\psi\cos\alpha & -r\cos\psi\sin\alpha & -r\sin\psi\cos\alpha \\ \sin\psi & 0 & r\cos\psi \end{bmatrix}$$

in this particular case.

4.8.1 A Practical Consideration

Storing a 3×3 covariance matrix for every point in a point cloud is not always necessary. In fact, the covariance matrix can be extracted from the range, azimuth, and elevation variances when needed. Even if points were to be downsampled, an approximation exists that makes it possible to compute the averaged covariance from the averaged variances of range, azimuth, and elevation. Specifically, note that (4.1), when applied to two points for example, can be rewritten as

$$\begin{aligned} \boldsymbol{\Sigma}_{avg} &= \frac{1}{4}\boldsymbol{\Sigma}_1 + \frac{1}{4}\boldsymbol{\Sigma}_2 \\ &= \frac{1}{4}\mathbf{F}_1 \begin{bmatrix} r_1 & 0 & 0 \\ 0 & \alpha_1 & 0 \\ 0 & 0 & \psi_1 \end{bmatrix} \mathbf{F}_1^\mathsf{T} + \frac{1}{4}\mathbf{F}_2 \begin{bmatrix} r_2 & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \psi_2 \end{bmatrix} \mathbf{F}_2^\mathsf{T} \\ &\approx \frac{1}{4}\mathbf{F}_{avg} \left(\begin{bmatrix} r_1 & 0 & 0 \\ 0 & \alpha_1 & 0 \\ 0 & 0 & \psi_1 \end{bmatrix} + \begin{bmatrix} r_2 & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \psi_2 \end{bmatrix} \right) \mathbf{F}_{avg}^\mathsf{T} \end{aligned}$$

where

$$\mathbf{F}_{avg} = \frac{\mathbf{F}_1 + \mathbf{F}_2}{2}.$$

Chapter 5

Local SLAM Strategy - Incremental Motion Estimation via Local Scan Matching

Finding the relative rigid-body transformation between a reference point cloud and another point cloud is a common problem encountered in robotics. Consider that a mobile robot measures a point cloud that is representative of the environment at a particular instance in time, and at a particular location, using an on-board sensor such as a camera or a LIDAR. At a later time, when the robot has moved to a new location, it will take a similar measurement of the environment. Provided that the two measurements have some overlap, matching or stitching these two point-cloud measurements in the frame of the reference point cloud is then possible. The resulting rigid-body transformation is equivalent to the motion of the robot as it travels from the first location to the second. Obtaining this relative pose-to-pose transformation is desirable because it provides the robot an exteroceptive visual odometry measurement more robust to error than odometry or dead-reckoning. These visual odometry estimates can then be fused with other sensor measurements in a Simultaneous Localization and Mapping (SLAM) framework to obtain even more accurate pose and mapping estimates for the robot.

Scan matching refers to the solving of this problem when the point cloud measurement, either 2D or 3D, is exclusively obtained via LIDAR. The challenge of scan matching is to not only minimize runtime but also be sufficiently accurate and robust to poor initialization and measurement noise. Most existing methods do not account for the inherent noise present in all LIDAR measurements, or the fact that some points may be more important than others. For instance, measurements of points far from the sensor have greater uncertainty than points closer to the sensor. Furthermore, points that appear as sharp edges of buildings

or surface patches of walls provide more information to the relative transformation than points from tree foliage or unstructured objects. This chapter presents a novel weighted point-cloud registration method that takes measurement uncertainties and feature points into account during the registration step of Iterative Closest Point (ICP)-based methods. In the following section, an overview of the most popular scan matching methods will be presented. A derivation of the weighted OLATE-based point-to-point, point-to-plane, and point-to-line registration methods will be presented followed by an algorithm that we term Total Registration, which solves all three problems at once. Finally, implementation details within an ICP-based framework are also presented.

5.1 Iterative Closest Point and Variants Thereof

Iterative Closest Point (ICP) is perhaps the most popular method for point-cloud registration. First introduced in Besl and McKay's seminal paper, the ICP algorithm has become popular amongst the robotics and computer vision community and is widely used in many applications. Although more accurate and robust algorithms have recently been developed for matching 2D scans [27], ICP still remains the defacto method for matching 3D data [45].

The ICP algorithm can be broadly divided into six steps, namely, 1) point selection from the point cloud to be matched, 2) point correspondence of these points with the reference point cloud, 3) weighting of the corresponding pairs, 4) rejection of outliers, 5) assigning of an error metric, and 6) minimization of the error metric via iteration. Because the ICP algorithm is more accurately a framework rather than a single algorithm, many variants of ICP have been developed that target some of the aforementioned steps. Some of these variants are detailed in [46]. The ICP method is far from perfect. One disadvantage of ICP is that it is known to converge slowly. This can be attributed to the point correspondence step, which generally takes the longest time. Specifically, Lu and Milios identified that "when the model is curved, the correspondences found by the closest-point rule may contain little information about the rotation", thus requiring additional iterations for convergence [30]. In light of this, they developed a new method named Iterative Dual Correspondence (IDC) that computes the translation estimate based on closest point correspondence and the rotation component based on a newly defined matching-range-point rule that responds more strongly to orientation changes.

ICP is also ill-suited to work with point clouds in the shape of non-uniform scan rings such as those obtained from Velodyne LIDARs [45]. This is true unless additional steps are taken to pre-process the data as mentioned in Chapter 4.

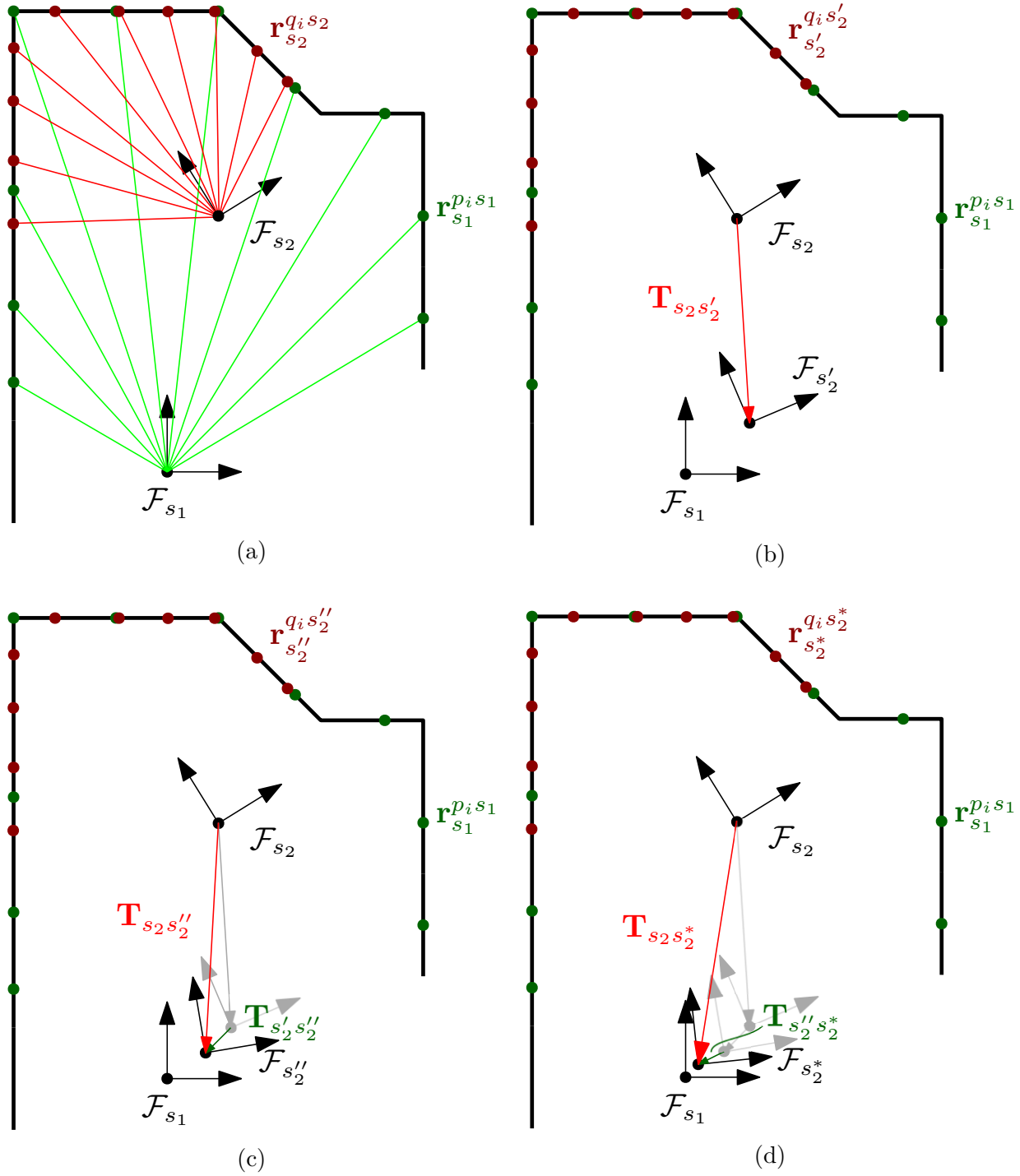


Figure 5.1: ICP procedure. a) Two point clouds $\mathbf{r}_{s_1}^{p_i s_1}$ and $\mathbf{r}_{s_2}^{q_i s_2}$ are taken in the sensor frame at different times. b) Apply an initial transformation of $\mathbf{T}_{s_2' s_2}$ to $\mathbf{r}_{s_2}^{q_i s_2}$, find points in $\mathbf{r}_{s_1}^{p_i s_1}$ that are closest to $\mathbf{r}_{s_2'}^{q_i s_2'}$ and establish point-point correspondence, noting that multiple points from $\mathbf{r}_{s_1}^{p_i s_1}$ can correspond to the same point in $\mathbf{r}_{s_2'}^{q_i s_2'}$. Compute $\mathbf{T}_{s_2'' s_2'}$ using a registration algorithm. Transform point cloud $\mathbf{r}_{s_2'}^{q_i s_2'}$ to obtain $\mathbf{r}_{s_2''}^{q_i s_2''} = \mathbf{T}_{s_2'' s_2'} \mathbf{r}_{s_2'}^{q_i s_2'}$. c) Repeat the same procedure as step b) to obtain $\mathbf{T}_{s_2^* s_2''}$ and $\mathbf{r}_{s_2^*}^{q_i s_2^*}$. d) Compute the total transformation via $\mathbf{T}_{s_2^* s_2} = \mathbf{T}_{s_2^* s_2''} \mathbf{T}_{s_2'' s_2'} \mathbf{T}_{s_2' s_2}$.

5.2 Establishing Point Correspondence

Given a reference point cloud and a target point cloud to be matched, if point-to-point association is not provided a priori, it must be established. This is known as the point-correspondence step.

The most common point correspondence strategy is the closest point correspondence. This involves corresponding each point in the target point cloud with the closest point in the reference point cloud in terms of Euclidean distance. In the event where an initial pose prediction is given, the correspondence is between the transformed target point cloud and their closest neighbouring point in the reference point cloud. In practice, this is done by ordering the point cloud into a k-dimensional (kd) tree structure to speed up the search.

Another way to establish point correspondence is taking the point in terms of an angular distance [30]. This is motivated by the fact that the further an observed point, the further it will be to its corresponding point given some rotation.

5.3 Point Cloud Registration

The registration method is at the heart of the ICP algorithm and is responsible for obtaining the relative transformation between two point clouds given their point-to-point or point-to-feature correspondence. The first attempt at solving the registration problem was to formulate it as a Wahba's problem [47]. Both Horn [48] and Arun *et al.* [49] posed the problem this way in 1987 and to date, this remains the *de facto* registration method. For the formulation of Wahba's problem and a derivation of point cloud registration via SVD, refer to Appendix A.

5.3.1 The Optimal Linear Attitude and Translation Estimator

The original Optimal Linear Attitude and Translation Estimator (OLATE) from [1, 50, 51] is rederived here with the difference that corresponding points are not assumed to be the same point.

Consider the diagram in Fig. 5.2 where point p_j is seen at timestep $k-1$ and point q_j, a point near point p_j, is seen at timestep k. Let s_{k-1} and s_k be the position of an exteroceptive sensor such as a LIDAR at timestep $k-1$ and k and let $\mathcal{F}_{s_{k-1}}$ and $\mathcal{F}_{s_k}$ be the vehicle's sensor frame at those times.

First, note that $\underrightarrow{e}^j$ can be written as

$$\underrightarrow{e}^j = \underrightarrow{r}^{q_j s_k} - \underrightarrow{r}^{p_j s_{k-1}} + \underrightarrow{r}^{s_k s_{k-1}}.$$

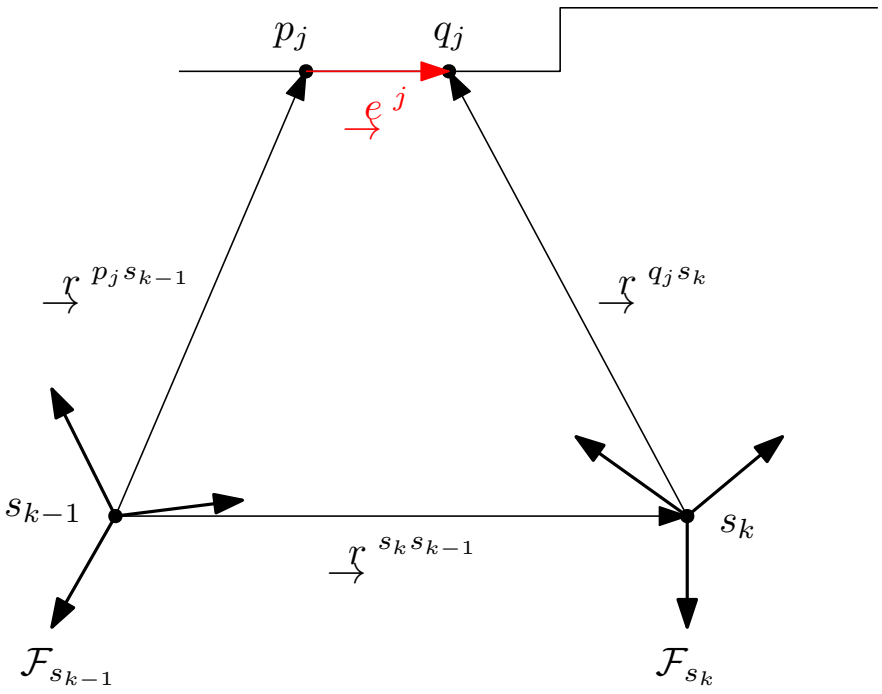


Figure 5.2: A wall as seen from two poses.

Resolving each physical vector in their respective frames yields

$$\mathbf{e}_{s_k}^{q_j p_j} = \mathbf{r}_{s_k}^{q_j s_k} - \mathbf{C}_{s_k s_{k-1}} \mathbf{r}_{s_{k-1}}^{p_j s_{k-1}} + \mathbf{r}_{s_k}^{s_k s_{k-1}}. \tag{5.1}$$

To simplify presentation, the s from the subscripts s_k and s_{k-1} will be henceforth dropped and written as simply k and $k-1$ respectively. Superscripts will be kept as is to distinguish between the measured points q or p. Furthermore, $\mathbf{e}_{s_k}^{q_j p_j}$ will be written as $\mathbf{e}_k^j$ for simplicity.

The position $\mathbf{r}_k^{s_k s_{k-1}}$ and attitude $\mathbf{C}_{kk-1}$ of the vehicle is sought such that $\vec{e_j} \cdot \vec{e_j}$, the distance between points p_j and q_j, is minimized. An optimization problem can be formulated where the cost function is defined as

$$J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) = \sum_{j=1}^{M} \frac{1}{2} \mathbf{e}_k^{j\,\mathsf{T}} \mathbf{e}_k^j. \tag{5.2}$$

Note that this is a least squares problem.
Using the Caley Transformation [52], the DCM $\mathbf{C}_{kk-1}$ can written as

$$\mathbf{C}_{kk-1} = (\mathbf{1} + \mathbf{p}_k^\times)^{-1}(\mathbf{1} - \mathbf{p}_k^\times), \tag{5.3}$$

where $\mathbf{p}_k = \mathbf{a}_k \tan\left(\frac{\phi_k}{2}\right)$ are the classic Rodriguez parameters, also called Gibbs parame-

ters [53]. This allows (5.1) to be rewritten as

$$\mathbf{e}_k^j = \mathbf{r}_k^{q_j s_k} - (\mathbf{1} + \mathbf{p}_k^\times)^{-1}(\mathbf{1} - \mathbf{p}_k^\times)\mathbf{r}_{k-1}^{p_j s_{k-1}} + \mathbf{r}_k^{s_k s_{k-1}}. \tag{5.4}$$

It is worth mentioning that a perceived drawback of choosing this particular parametrization of the DCM is that Rodriguez parameters suffer from a singularity at 180°. However, using a Rodriguez parameter is inconsequential as most modern day robots tend to either limit such a high angular velocity via an on-board controller or avoid such manoeuvres in the guidance algorithm. Continuing, define a new residual $\bar{\mathbf{e}}_k^j = (\mathbf{1} + \mathbf{p}_k^\times)\mathbf{e}_k^j$, that becomes

$$\begin{aligned}
\bar{\mathbf{e}}_k^j &= (\mathbf{1} + \mathbf{p}_k^\times)\mathbf{r}_k^{q_j s_k} - (\mathbf{1} - \mathbf{p}_k^\times)\mathbf{r}_{k-1}^{p_j s_{k-1}} + (\mathbf{1} + \mathbf{p}_k^\times)\mathbf{r}_k^{s_k s_{k-1}} \\
&= (\mathbf{r}_k^{q_j s_k} - \mathbf{r}_{k-1}^{p_j s_{k-1}}) - \Big(- \mathbf{p}^\times(\mathbf{r}_k^{q_j s_k} + \mathbf{r}_{k-1}^{p_j s_{k-1}}) \underbrace{-(\mathbf{1} + \mathbf{p}_k^\times)\mathbf{r}_k^{s_k s_{k-1}}}_{+\mathbf{t}_k} \Big) \\
&= (\mathbf{r}_k^{q_j s_k} - \mathbf{r}_{k-1}^{p_j s_{k-1}}) - \big((\mathbf{r}_k^{q_j s_k} + \mathbf{r}_{k-1}^{p_j s_{k-1}})^\times \mathbf{p}_k + \mathbf{t}_k\big) \\
&= \underbrace{(\mathbf{r}_k^{q_j s_k} - \mathbf{r}_{k-1}^{p_j s_{k-1}})}_{\mathbf{b}_k} - \underbrace{\begin{bmatrix} \mathbf{1} & (\mathbf{r}_k^{q_j s_k} + \mathbf{r}_{k-1}^{p_j s_{k-1}})^\times \end{bmatrix}}_{\mathbf{A}_k^j} \underbrace{\begin{bmatrix} \mathbf{t}_k \\ \mathbf{p}_k \end{bmatrix}}_{\mathbf{x}_k} \\
&= \mathbf{b}_k^j - \mathbf{A}_k^j \mathbf{x}_k
\end{aligned} \tag{5.5}$$

Proposition 5.1. *If a matrix* $\mathbf{A}$ *is skew symmetric, then* $(\mathbf{1} + \mathbf{A})$ *is invertible.*

Proof. See Appendix A.

Applying proposition 5.1, $(\mathbf{1} + \mathbf{p}_k^\times)$ is shown to be invertible, and using the relationship

$$\mathbf{e}_k^j = (\mathbf{1} + \mathbf{p}_k^\times)^{-1}\bar{\mathbf{e}}_k^j$$

the cost function defined in (5.2) can be rewritten as

$$\begin{aligned}
J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) &= \sum_{j=1}^{M} \frac{1}{2} \bar{\mathbf{e}}_k^{j\mathsf{T}} \underbrace{(\mathbf{1} + \mathbf{p}_k^\times)^{-\mathsf{T}}(\mathbf{1} + \mathbf{p}_k^\times)^{-1}}_{\mathbf{W}_k^j} \bar{\mathbf{e}}_k^j \\
&= \frac{1}{2}(\mathbf{b}_k - \mathbf{A}_k\mathbf{x}_k)^\mathsf{T}\mathbf{W}_k(\mathbf{b}_k - \mathbf{A}_k\mathbf{x}_k)
\end{aligned} \tag{5.6}$$

where

$$\mathbf{b}_k = \begin{bmatrix} \mathbf{b}_k^1 \\ \vdots \\ \mathbf{b}_k^j \\ \vdots \\ \mathbf{b}_k^M \end{bmatrix}, \qquad \mathbf{A}_k^j = \begin{bmatrix} \mathbf{A}_k^{j\,1} \\ \vdots \\ \mathbf{A}_k^j \\ \vdots \\ \mathbf{A}_k^{j\,M} \end{bmatrix}, \qquad \mathbf{W}_k = \operatorname{diag}\left\{\mathbf{W}_k^1, \cdots, \mathbf{W}_k^j \cdots, \mathbf{W}_k^M\right\}.$$

The weight $\mathbf{W}_k^j$ depends on the rotation. Noting that

$$\begin{aligned}(\mathbf{1} + \mathbf{p}_k^\times)(\mathbf{1} + \mathbf{p}_k^\times)^\mathsf{T} &\geq \mathbf{1}. \\ \left((\mathbf{1} + \mathbf{p}_k^\times)(\mathbf{1} + \mathbf{p}_k^\times)^\mathsf{T}\right)^{-1} &\leq \mathbf{1},\end{aligned}$$

setting $\bar{\mathbf{W}}_k^j = \mathbf{1}$ leads to a new cost function

$$\bar{J}_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) = \frac{1}{2}\bar{\mathbf{e}}_k^\mathsf{T}\bar{\mathbf{W}}_k\bar{\mathbf{e}}_k \geq \frac{1}{2}\bar{\mathbf{e}}_k^\mathsf{T}\mathbf{W}_k\bar{\mathbf{e}}_k = J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}).$$

Thus, $\bar{J}_k$ overbounds the cost function of (5.6).

The solution to this optimization is then as follows. First, set $\bar{\mathbf{W}}_k^j = \mathbf{1}$ and proceed to solve the modified cost function $\bar{J}_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1})$ which is an ordinary least squares problem with the normal solution outlined in (2.2). The solution is thus

$$\begin{bmatrix} \mathbf{t}_k \\ \mathbf{p}_k \end{bmatrix} = (\mathbf{A}_k^{j\,\mathsf{T}}\mathbf{A}_k^j)^{-1}(\mathbf{A}_k^{j\,\mathsf{T}}\mathbf{b}_k). \tag{5.7}$$

After solving for $\mathbf{t}_k$ and $\mathbf{p}_k$, the relative transformation of interest can be extracted via

$$\begin{aligned}\mathbf{r}_k^{s_k s_{k-1}} &= -(\mathbf{1} + \mathbf{p}_k^\times)^{-1}\mathbf{t}_k, \\ \mathbf{C}_{kk-1} &= (\mathbf{1} + \mathbf{p}_k^\times)^{-1}(\mathbf{1} - \mathbf{p}_k^\times).\end{aligned}$$

Once an estimate for $\mathbf{p}_k$ has been obtained, substitute into (5.6) and iterate to obtain a better estimate for the desired states.

In practical applications, when ICP scan matching is used for consecutive pose-to-pose scan matching, the rotation between each pose is small and using a small angle approximation, $\mathbf{W}_k^j \approx \mathbf{1}$, meaning that the unweighed solution is generally valid, especially if a good initial prediction is given. However in the case of an aggressive turn, and with noise corrupting heading estimates, a reliable heading estimate can no longer be provided and the

expected heading error can be much larger. When ICP is used for loop closure, one can expect arbitrarily large rotations and therefore, the small angle approximation is no longer valid. In such a case, there is ample motivation to compute the weight matrix $\mathbf{W}_k$.

5.3.2 Weighted OLATE

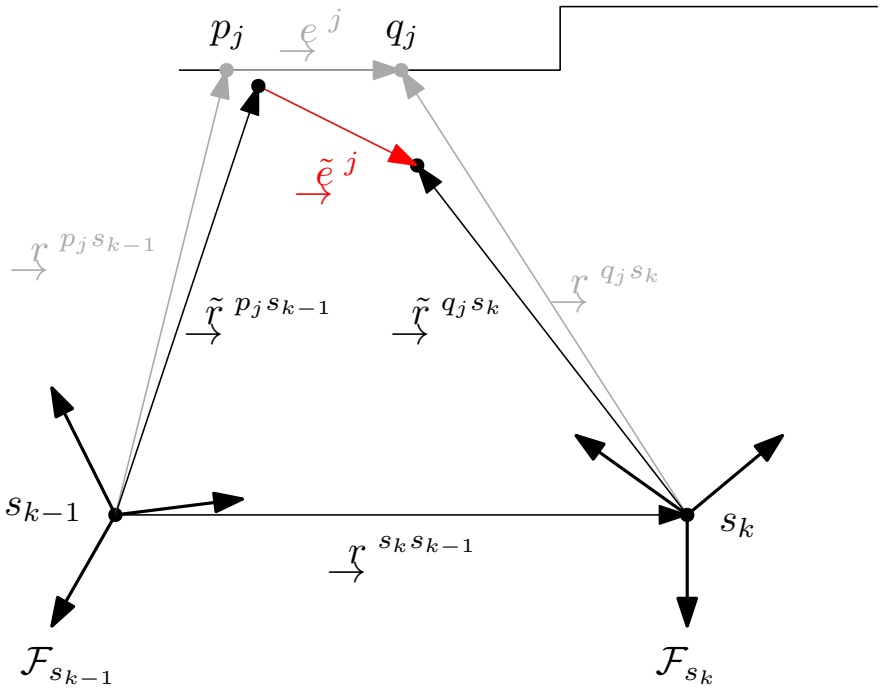


Figure 5.3: A wall as seen from two poses.

Taking the previous idea of weighing the error residuals of the cost function further, it is possible to introduce additional weights into $\mathbf{W}_k$ that correspond to the uncertainty of the measurements. Suppose that the measurements $\mathbf{r}_k^{q_j s_k}$ and $\mathbf{r}_{k-1}^{p_j s_{k-1}}$ are no longer deterministic, but rather are corrupted by noise of the form

$$\tilde{\mathbf{r}}_k^{q_j s_k} = \mathbf{r}_k^{q_j s_k} + \mathbf{w}_k^{q_j}, \tag{5.8}$$

$$\tilde{\mathbf{r}}_k^{p_j s_{k-1}} = \mathbf{r}_{k-1}^{p_j s_{k-1}} + \mathbf{w}_{k-1}^{p_j}. \tag{5.9}$$

where $\mathbf{w}_{k-1}^{p_j} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{k-1}^j)$ and $\mathbf{w}_k^{q_j} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k^j)$. Define a new error term

$$\begin{aligned}
\tilde{\mathbf{e}}_k^j &= \tilde{\mathbf{r}}_k^{q_j s_k} - (\mathbf{C}_{kk-1}\tilde{\mathbf{r}}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{s_k s_{k-1}}) \\
&= \mathbf{r}_k^{q_j s_k} - (\mathbf{C}_{kk-1}\mathbf{r}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{s_k s_{k-1}}) + \mathbf{w}_k^{q_j} - \mathbf{C}_{kk-1}\mathbf{w}_{k-1}^{p_j}, \\
&= \mathbf{e}_k^j + \mathbf{w}_k^{q_j} - \mathbf{C}_{kk-1}\mathbf{w}_{k-1}^{p_j}.
\end{aligned}$$

It is common to set the weight as the inverse of the covariance matrix $\mathbf{\Sigma}_k^j$ associated with the error, where

$$\begin{aligned}
\mathbf{\Sigma}_k^j &= E[\tilde{\mathbf{e}}_k^j \tilde{\mathbf{e}}_k^{j\mathsf{T}}] \\
&= E[(\mathbf{e}_k^j + \mathbf{w}_k^{q_j} - \mathbf{C}_{kk-1}\mathbf{w}_{k-1}^{p_j})(\mathbf{e}_k^j + \mathbf{w}_k^{q_j} - \mathbf{C}_{kk-1}\mathbf{w}_{k-1}^{p_j})^{\mathsf{T}}] \\
&= E[\mathbf{e}_k^j \mathbf{e}_k^{j\mathsf{T}}] + E[\mathbf{e}_k^j \mathbf{w}_k^{j\mathsf{T}}] + E[\mathbf{w}_k^j \mathbf{e}_k^{j\mathsf{T}}] - E[\mathbf{e}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}}]\mathbf{C}_{kk-1}^{\mathsf{T}} - \mathbf{C}_{kk-1}E[\mathbf{w}_{k-1}^j \mathbf{e}_k^{j\mathsf{T}}] \\
&\quad + E[\mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}}] - E[\mathbf{w}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}}]\mathbf{C}_{kk-1}^{\mathsf{T}} - \mathbf{C}_{kk-1}E[\mathbf{w}_{k-1}^j \mathbf{w}_k^{j\mathsf{T}}] \\
&\quad + \mathbf{C}_{kk-1}E[\mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\mathsf{T}}]\mathbf{C}_{kk-1}^{\mathsf{T}}.
\end{aligned}$$

At this stage, it is clear that $\mathbf{e}_k^j$ is correlated with the noise vectors $\mathbf{w}_{k-1}^j$ and $\mathbf{w}_k^j$. Because there is no information about the actual value of $\mathbf{e}_k^j$, this becomes an unsolvable problem. If however we make the assumption that point p_j and q_j are the same point, it follows that $\mathbf{e}_k^j = \mathbf{0}$ and we can continue to solve for the error covariance, yielding

$$\begin{aligned}
\mathbf{\Sigma}_k^j &= \cancel{E[\mathbf{e}_k^j \mathbf{e}_k^{j\mathsf{T}}]} + \cancel{E[\mathbf{e}_k^j \mathbf{w}_k^{j\mathsf{T}}]} + \cancel{E[\mathbf{w}_k^j \mathbf{e}_k^{j\mathsf{T}}]} - \cancel{E[\mathbf{e}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}}]\mathbf{C}_{kk-1}^{\mathsf{T}}} - \cancel{\mathbf{C}_{kk-1}E[\mathbf{w}_{k-1}^j \mathbf{e}_k^{j\mathsf{T}}]} \\
&\quad + E[\mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}}] - E[\mathbf{w}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}}]\mathbf{C}_{kk-1}^{\mathsf{T}} - \mathbf{C}_{kk-1}E[\mathbf{w}_{k-1}^j \mathbf{w}_k^{j\mathsf{T}}] \\
&\quad + \mathbf{C}_{kk-1}E[\mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\mathsf{T}}]\mathbf{C}_{kk-1}^{\mathsf{T}} \\
&= \mathbf{R}_k^j + \mathbf{C}_{kk-1}\mathbf{R}_{k-1}^j \mathbf{C}_{kk-1}^{\mathsf{T}}.
\end{aligned}$$

The point cost function is thus

$$J_k^j(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) = \frac{1}{2}\tilde{\mathbf{e}}_k^{j\mathsf{T}} \mathbf{\Sigma}_k^{j\,-1} \tilde{\mathbf{e}}_k^j.$$

5.3.2.1 Using the Cayley Transformation to Rewrite the Cost Function

Applying the previously shown Cayley transform to $\mathbf{C}_{kk-1}$ and using the relationship

$$\tilde{\mathbf{e}}_k^j = (\mathbf{1} + \mathbf{p}_k^{\times})^{-1} \bar{\mathbf{e}}_k^j,$$

as in Section 5.3.1, the point cost function can be written as

$$\begin{aligned}
J_k^j(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) &= \tfrac{1}{2}\tilde{\mathbf{e}}_k^{j\mathsf{T}} \mathbf{\Sigma}_k^{j\,-1} \tilde{\mathbf{e}}_k^j \\
&= \tfrac{1}{2}\bar{\mathbf{e}}_k^{j\mathsf{T}} (\mathbf{1} + \mathbf{p}_k^{\times})^{-\mathsf{T}} \mathbf{\Sigma}_k^{j\,-1} (\mathbf{1} + \mathbf{p}_k^{\times})^{-1} \bar{\mathbf{e}}_k^j \\
&= \tfrac{1}{2}\bar{\mathbf{e}}_k^{j\mathsf{T}} \left((\mathbf{1} + \mathbf{p}_k^{\times}) \mathbf{\Sigma}_k^j (\mathbf{1} + \mathbf{p}_k^{\times})^{\mathsf{T}} \right)^{-1} \bar{\mathbf{e}}_k^j \\
&= \tfrac{1}{2}\bar{\mathbf{e}}_k^{j\mathsf{T}} \mathbf{W}_k^j \bar{\mathbf{e}}_k^j
\end{aligned}$$

where

$$\mathbf{W}_k^j = \left((\mathbf{1} + \mathbf{p}_k^\times)\boldsymbol{\Sigma}_k^j(\mathbf{1} + \mathbf{p}_k^\times)^\mathsf{T}\right)^{-1}. \tag{5.10}$$

5.3.2.2 Weight Simplification and Overbounding the Cost Function

It turns out that the weight matrix $\mathbf{W}_k^j$ can be overbounded. First, simplify the weight, letting

$$\begin{aligned}
\bar{\boldsymbol{W}}_k^j &= (\mathbf{1} + \mathbf{p}_k^\times)\boldsymbol{\Sigma}_k^j(\mathbf{1} + \mathbf{p}_k^\times)^\mathsf{T} \\
&= (\mathbf{1} + \mathbf{p}_k^\times)\left(\mathbf{R}_k^j + \mathbf{C}_{k\,k-1}\mathbf{R}_{k-1}^j\mathbf{C}_{k\,k-1}^\mathsf{T}\right)(\mathbf{1} + \mathbf{p}_k^\times)^\mathsf{T} \\
&= (\mathbf{1} + \mathbf{p}_k^\times)\mathbf{R}_k^j(\mathbf{1} + \mathbf{p}_k^\times)^\mathsf{T} + (\mathbf{1} + \mathbf{p}_k^\times)\mathbf{C}_{k\,k-1}\mathbf{R}_{k-1}^j\mathbf{C}_{k\,k-1}^\mathsf{T}(\mathbf{1} + \mathbf{p}_k^\times)^\mathsf{T}.
\end{aligned}$$

Using Equation (5.3),

$$\begin{aligned}
(\mathbf{1} + \mathbf{p}_k^\times)\mathbf{C}_{k\,k-1} &= (\mathbf{1} + \mathbf{p}_k^\times)(\mathbf{1} + \mathbf{p}_k^\times)^{-1}(\mathbf{1} - \mathbf{p}_k^\times) \\
&= (\mathbf{1} - \mathbf{p}_k^\times), \\
\mathbf{C}_{k\,k-1}^\mathsf{T}(\mathbf{1} + \mathbf{p}_k^\times)^\mathsf{T} &= \left((\mathbf{1} + \mathbf{p}_k^\times)\mathbf{C}_{k\,k-1}\right)^\mathsf{T} \\
&= (\mathbf{1} - \mathbf{p}_k^\times)^\mathsf{T} \\
&= (\mathbf{1} - \mathbf{p}_k^{\times\mathsf{T}}).
\end{aligned}$$

Therefore, the weighting matrix $\bar{\boldsymbol{W}}_k^j$ can be expanded and written as

$$\begin{aligned}
\bar{\boldsymbol{W}}_k^j &= (\mathbf{1} + \mathbf{p}_k^\times)\mathbf{R}_k^j(\mathbf{1} + \mathbf{p}_k^\times)^\mathsf{T} + (\mathbf{1} - \mathbf{p}_k^\times)\mathbf{R}_{k-1}^j(\mathbf{1} - \mathbf{p}_k^{\times\mathsf{T}}) \\
&= \mathbf{R}_k^j + \mathbf{R}_k^j\mathbf{p}_k^{\times\mathsf{T}} + \mathbf{p}_k^\times\mathbf{R}_k^j + \mathbf{p}_k^\times\mathbf{R}_k^j\mathbf{p}_k^{\times\mathsf{T}} + \mathbf{R}_{k-1}^j - \mathbf{R}_{k-1}^j\mathbf{p}_k^{\times\mathsf{T}} - \mathbf{p}_k^\times\mathbf{R}_{k-1}^j + \mathbf{p}_k^\times\mathbf{R}_{k-1}^j\mathbf{p}_k^{\times\mathsf{T}} \\
&= \mathbf{p}_k^\times\left(\mathbf{R}_k^j + \mathbf{R}_{k-1}^j\right)\mathbf{p}_k^{\times\mathsf{T}} - \left(\mathbf{R}_{k-1}^j - \mathbf{R}_k^j\right)\mathbf{p}_k^{\times\mathsf{T}} - \mathbf{p}_k^\times\left(\mathbf{R}_{k-1}^j - \mathbf{R}_k^j\right) + \left(\mathbf{R}_k^j + \mathbf{R}_{k-1}^j\right).
\end{aligned}$$

Letting

$$\begin{aligned}
\mathbf{S} &= \left(\mathbf{R}_k^j + \mathbf{R}_{k-1}^j\right), \\
\mathbf{T} &= \left(\mathbf{R}_{k-1}^j - \mathbf{R}_k^j\right), \\
\mathbf{U} &= \left(\mathbf{R}_k^j + \mathbf{R}_{k-1}^j\right), \\
\boldsymbol{\Theta} &= \mathbf{p}_k^{\times\mathsf{T}},
\end{aligned}$$

$\bar{\boldsymbol{W}}_k^j$ can be written and simplified via the following identity outlined in [54] where

$$\begin{aligned}
\bar{\boldsymbol{W}}_k^j &= \boldsymbol{\Theta}^\mathsf{T}\mathbf{S}\boldsymbol{\Theta} - \mathbf{T}\boldsymbol{\Theta} - \boldsymbol{\Theta}^\mathsf{T}\mathbf{T}^\mathsf{T} + \mathbf{U} \\
&= \left(\boldsymbol{\Theta} - \mathbf{S}^{-1}\mathbf{T}\right)^\mathsf{T}\mathbf{S}\left(\boldsymbol{\Theta} - \mathbf{S}^{-1}\mathbf{T}\right) + \mathbf{U} - \mathbf{T}^\mathsf{T}\mathbf{S}^{-1}\mathbf{T} \\
&\geq \mathbf{U} - \mathbf{T}^\mathsf{T}\mathbf{S}^{-1}\mathbf{T} \\
&= \left(\mathbf{R}_k^j + \mathbf{R}_{k-1}^j\right) - \left(\mathbf{R}_{k-1}^j - \mathbf{R}_k^j\right)\left(\mathbf{R}_k^j + \mathbf{R}_{k-1}^j\right)^{-1}\left(\mathbf{R}_{k-1}^j - \mathbf{R}_k^j\right) \\
&= \bar{\mathbf{W}}_k^j,
\end{aligned}$$

which is also to say that $\bar{\mathbf{W}}_k^j \geq \mathbf{W}_k^j = \bar{\boldsymbol{W}}_k^{j^{-1}}$, Then, by defining $\bar{J}_k^j(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) = \frac{1}{2}\bar{\mathbf{e}}_k^{j\mathsf{T}}\bar{\mathbf{W}}_k^j\bar{\mathbf{e}}_k^j$ it follows that

$$J_k^j(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) = \tfrac{1}{2}\bar{\mathbf{e}}_k^{j\mathsf{T}}\mathbf{W}_k^j\bar{\mathbf{e}}_k^j \leq \tfrac{1}{2}\bar{\mathbf{e}}_k^{j\mathsf{T}}\bar{\mathbf{W}}_k^j\bar{\mathbf{e}}_k^j = \bar{J}_k^j.$$

Thus, $\bar{J}_k^j(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1})$ overbounds $J_k^j(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1})$. Moreover, by minimizing $\bar{J}_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1})$, an upper bound on $J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1})$ is minimized.

5.3.2.3 Optimal Pose Using the New Cost Function

Given that $\bar{J}_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1})$ overbounds $J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1})$, the optimal $\mathbf{r}_k^{s_k z k-1}$ and $\mathbf{C}_{kk-1}$ can be obtained as follows. First, solve the weighted least squares problem

$$\begin{aligned}
\bar{J}_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) &= \sum_{j=1}^{M} \bar{J}_k^j(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) \\
&= \sum_{j=1}^{M} \tfrac{1}{2}\bar{\mathbf{e}}_k^{j\mathsf{T}}\bar{\mathbf{W}}_k^{j^{-1}}\bar{\mathbf{e}}_k^j \\
&= \tfrac{1}{2}(\mathbf{b}_k - \mathbf{A}_k^j\mathbf{x}_k)^\mathsf{T}\bar{\mathbf{W}}_k^{-1}(\mathbf{b}_k - \mathbf{A}_k^j\mathbf{x}_k) \\
&= \tfrac{1}{2}\mathbf{b}_k^\mathsf{T}\bar{\mathbf{W}}_k^{-1}\mathbf{b}_k + \mathbf{b}_k^\mathsf{T}\bar{\mathbf{W}}_k^{-1}\mathbf{A}_k^j\mathbf{x}_k + \tfrac{1}{2}\mathbf{x}_k^\mathsf{T}\mathbf{A}_k^{j\mathsf{T}}\bar{\mathbf{W}}_k^{-1}\mathbf{A}_k^j\mathbf{x}_k.
\end{aligned}$$

Where the expressions for $\mathbf{b}_k$ and $\mathbf{A}_k^j$ are given in (5.5). The solution, after simplification, is

$$\mathbf{x}_k = (\mathbf{A}_k^{j\mathsf{T}}\bar{\mathbf{W}}_k^{-1}\mathbf{A}_k^j)^{-1}\bar{\mathbf{W}}_k^{-1}\mathbf{A}_k^{j\mathsf{T}}\mathbf{b}_k, \tag{5.11}$$

where the states of interest can be extracted via

$$\mathbf{r}_k^{s_k s_{k-1}} = -(\mathbf{1} + \mathbf{p}_k^\times)^{-1}\mathbf{t}_k, \tag{5.12}$$

$$\mathbf{C}_{kk-1} = (\mathbf{1} + \mathbf{p}_k^\times)^{-1}(\mathbf{1} - \mathbf{p}_k^\times). \tag{5.13}$$

Because an estimate for $\mathbf{C}_{kk-1}$ is now available, in practice, one would go back to (5.10) to compute a better estimate for the weight, $\mathbf{W}_k^j$ and iterate until convergence.

5.3.3 Point-to-Plane Registration Using OLATE

The standard approach to solving point-to-plane registration was first proposed by Chen and Medioni in [28] and later derived in detail in [55]. This method involves linearizing the problem by making small angle approximations. For a derivation of this method, please see Appendix A. Another method is just to leave the cost function in its nonlinear form and apply GN or LM. As noted in [46], at the time the paper was written, there were no closed-form solutions to the point-to-plane problem. In recent years, a closed-form solution has been found in [56] though this method solves a linear system of 12 parameters. Furthermore, when measurements are noisy, the rotation matrix obtained in [56] is not necessarily orthogonal so an additional SVD step is needed to enforce orthogonality.

The solution obtained via OLATE is not only a closed-form solution but also solves for 3 parameters of the Rodriguez parameter $\mathbf{p}$ that can be mapped back directly to $SO(3)$, thereby naturally guaranteeing orthogonality of the rotation matrix.

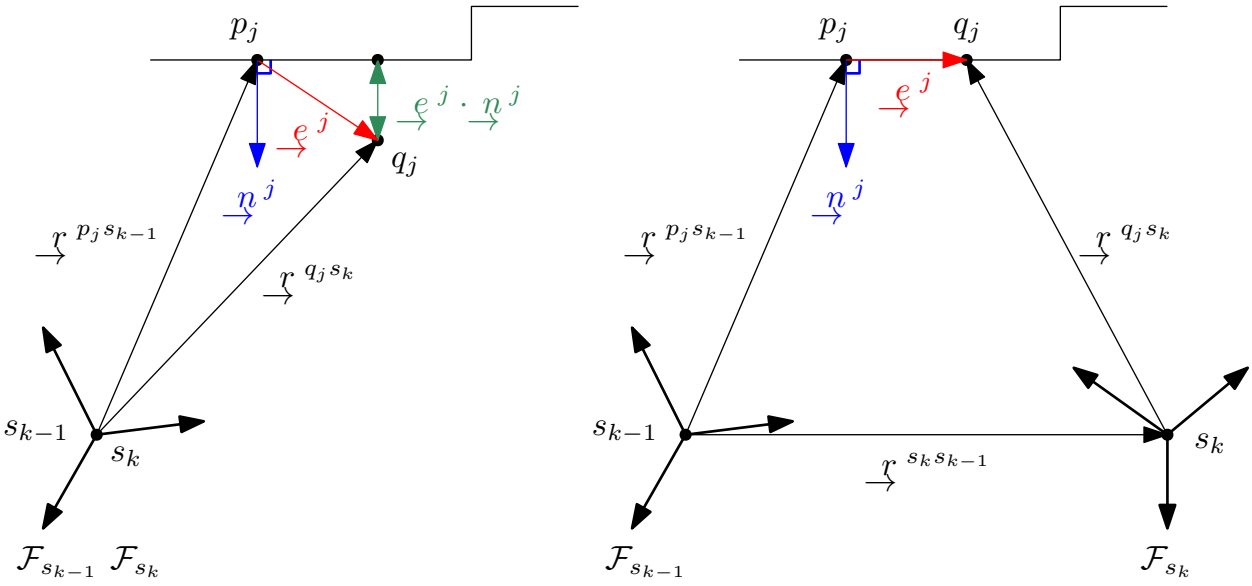


Figure 5.4: Point-to-plane registration problem statement. Left: Frame-centric view before registration completion. Right: After point q_j has been registered.

Consider the diagram in Fig. 5.4 where point p_j is seen at timestep $k-1$ and point q_j, a point near point p_j, is seen at timestep k.

Instead of minimizing the euclidean distance between the measured points p_j and q_j, that is $\underrightarrow{e}^j \cdot \underrightarrow{e}^j$, the new objective is to minimize the distance between point q_j and the local

surface around point p_j, that is $\underrightarrow{e}^j \cdot \underrightarrow{n}^j$. Let $\hat{e}_k^j = \mathbf{n}_k^{j\mathsf{T}} \mathbf{e}_k^j$, then, the new cost function to minimize is

$$\begin{aligned} J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) &= \sum_{j=1}^{M} \tfrac{1}{2} \hat{e}_k^{j\,2} \\ &= \sum_{j=1}^{M} \tfrac{1}{2} \mathbf{e}_k^{j\mathsf{T}} \mathbf{n}_k^j \mathbf{n}_k^{j\mathsf{T}} \mathbf{e}_k^j \\ &= \sum_{j=1}^{M} \tfrac{1}{2} \mathbf{e}_k^{j\mathsf{T}} \mathbf{N}_k^j \mathbf{e}_k^j \end{aligned} \tag{5.14}$$

where just as before, the residual is

$$\begin{aligned} \mathbf{e}_k^j &= \mathbf{r}_k^{q_j s_k} - \mathbf{C}_{kk-1} \mathbf{r}_{k-1}^{p_j s_{k-1}} + \mathbf{r}_k^{s_k s_{k-1}} \\ &= \mathbf{r}_k^{q_j s_k} - (\mathbf{1} + \mathbf{p}_k^{\times})^{-1} (\mathbf{1} - \mathbf{p}_k^{\times}) \mathbf{r}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{s_k s_{k-1}}. \end{aligned}$$

Note that $\mathbf{N}_k^j$ is a projection matrix that is always singular, which brings forth complications when attempting to invert the weight associated with this point-to-plane problem.

Next, define a new residual $\bar{\mathbf{e}}_k^j = (\mathbf{1} + \mathbf{p}_k^{\times}) \mathbf{e}_k^j$, and after the same simplification steps made in (5.5), the expression for $\bar{\mathbf{e}}_k^j$ becomes

$$\begin{aligned} \bar{\mathbf{e}}_k^j &= (\mathbf{1} + \mathbf{p}_k^{\times}) \tilde{\mathbf{r}}_k^{jk} - (\mathbf{1} - \mathbf{p}_k^{\times}) \tilde{\mathbf{r}}_{k-1}^{jk-1} - (\mathbf{1} + \mathbf{p}_k^{\times}) \mathbf{r}_k^{kk-1} \\ &= \mathbf{b}_k^j - \mathbf{A}_k^j \mathbf{x}_k. \end{aligned}$$

Using the relationship

$$\mathbf{e}_k^j = (\mathbf{1} + \mathbf{p}_k^{\times})^{-1} \bar{\mathbf{e}}_k^j$$

equation (5.14) can be rewritten as

$$J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) = \sum_{j=1}^{M} \tfrac{1}{2} \bar{\mathbf{e}}_k^{j\mathsf{T}} \underbrace{(\mathbf{1} + \mathbf{p}_k^{\times})^{-\mathsf{T}} \mathbf{N}_k^j (\mathbf{1} + \mathbf{p}_k^{\times})^{-1}}_{\mathbf{W}_k^j} \bar{\mathbf{e}}_k^j.$$

Since $(\mathbf{1}+\mathbf{p}_k^{\times})^{-\mathsf{T}}\mathbf{N}_k^j(\mathbf{1}+\mathbf{p}_k^{\times})^{-1} \leq \mathbf{N}_k^j = \bar{\mathbf{W}}_k^j$, it follows that

$$\begin{aligned} J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) &= \sum_{j=1}^{M} \tfrac{1}{2}\bar{\mathbf{e}}_k^{j\mathsf{T}}(\mathbf{1}+\mathbf{p}_k^{\times})^{-\mathsf{T}}\mathbf{N}_k^j(\mathbf{1}+\mathbf{p}_k^{\times})^{-1}\bar{\mathbf{e}}_k^j \\ &\leq \sum_{j=1}^{M} \bar{\mathbf{e}}_k^{j\mathsf{T}}\mathbf{N}_k^j\bar{\mathbf{e}}_k^j = \bar{J}_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}). \end{aligned}$$

In other words, $\bar{J}_k$ overbounds $J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1})$ and minimizing $\bar{J}_k$ minimizes an upper bound of $J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1})$. The solution procedure follows similar steps as those derived for point-to-point WOLATE. Namely, solve (5.11) and (5.13), then recompute $\mathbf{W}_k = (\mathbf{1}+\mathbf{p}_k^{\times})^{-\mathsf{T}}\mathbf{N}_k^j(\mathbf{1}+\mathbf{p}_k^{\times})^{-1}$ and iterate until convergence.

5.3.4 Weighted OLATE-Based Point-to-Plane Registration

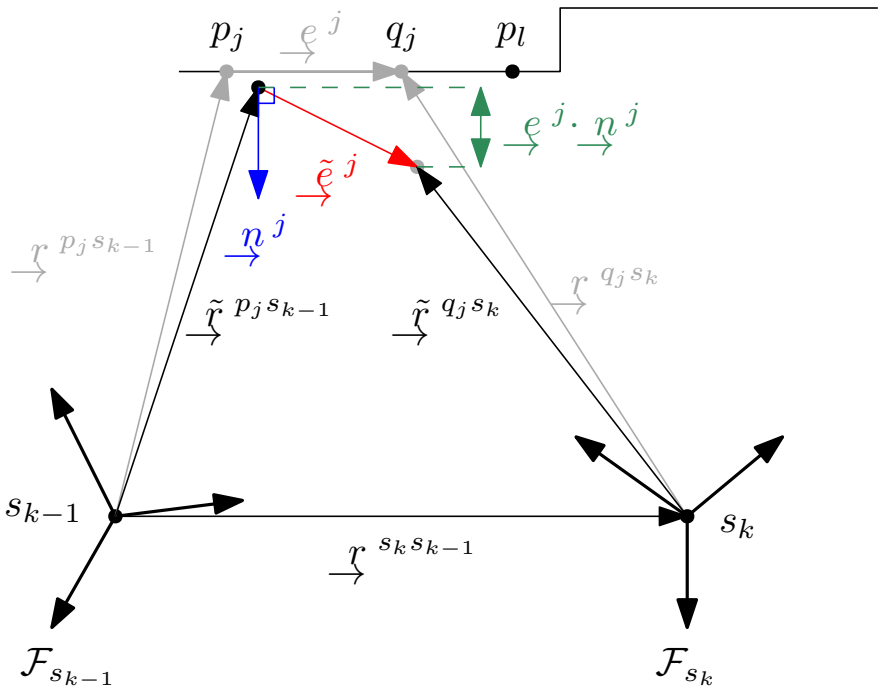


Figure 5.5: Point-to-plane registration problem statement.

Consider now the weighted case where the measurements are no longer deterministic, that is

$$\begin{aligned} \tilde{\mathbf{r}}_k^{q_j s_k} &= \mathbf{r}_k^{q_j s_k} + \mathbf{w}_k^{q_j}, \\ \tilde{\mathbf{r}}_k^{p_j s_{k-1}} &= \mathbf{r}_{k-1}^{p_j s_{k-1}} + \mathbf{w}_{k-1}^{p_j}. \end{aligned}$$

where $\mathbf{w}_{k-1}^{p_j} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{k-1}^{j})$ and $\mathbf{w}_{k}^{q_j} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{k}^{j})$.

The cost function to minimize is given by

$$J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) = \sum_{j=1}^{M} \frac{1}{2} (\underbrace{\tilde{\mathbf{e}}_k^{j\mathsf{T}} \mathbf{n}_k^j}_{\hat{e}_k^{j\mathsf{T}}})^2 \tag{5.15}$$

where $\hat{e}_k^j = \mathbf{n}_k^{j\mathsf{T}} \tilde{\mathbf{e}}_k^j = \tilde{\mathbf{e}}_k^{j\mathsf{T}} \mathbf{n}_k^j$ is the projection of vector $\tilde{\mathbf{e}}_k^j$ along the normal direction $\mathbf{n}_k^j$ of the plane.

At this point, it is worth deriving the covariance matrix associated with this error where the covariance is defined as $E[\hat{e}_k^{j\,2}]$. First note that the error residual and noise vectors are related via

$$\begin{aligned}
\tilde{\mathbf{e}}_k^j &= \tilde{\mathbf{r}}_k^{q_j s_k} - (\mathbf{C}_{kk-1} \tilde{\mathbf{r}}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{s_k s_{k-1}}) \\
&= \mathbf{r}_k^{q_j s_k} - (\mathbf{C}_{kk-1} \mathbf{r}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{s_k s_{k-1}}) + \mathbf{w}_k^{q_j} - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^{p_j} \\
&= \mathbf{e}_k^j + \mathbf{w}_k^{q_j} - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^{p_j}.
\end{aligned}$$

The variance associated with the error is then

$$\begin{aligned}
\sigma_k^{j\,2} &= E[\hat{e}_k^{j\,2}] \\
&= E[(\mathbf{n}_k^{j\mathsf{T}} \tilde{\mathbf{e}}_k^j)(\tilde{\mathbf{e}}_k^{j\mathsf{T}} \mathbf{n}_k^j)] \\
&= E[\mathbf{n}_k^{j\mathsf{T}} (\mathbf{e}_k^j + \mathbf{w}_k^{q_j} - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^{p_j})(\mathbf{e}_k^j + \mathbf{w}_k^{q_j} - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^{p_j})^{\mathsf{T}} \mathbf{n}_k^j].
\end{aligned}$$

Notice that $\mathbf{n}_k^{j\mathsf{T}} \mathbf{e}_k^j = \mathbf{e}_k^{j\mathsf{T}} \mathbf{n}_k^j = 0$. Continuing,

$$\begin{aligned}
\sigma_k^{j\,2} &= E[\mathbf{n}_k^{j\mathsf{T}} (\mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}} - \mathbf{w}_k^j \mathbf{w}_{k-1}^{j\,\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}} - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\,\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}) \mathbf{n}_k^j] \\
&= \mathbf{n}_k^{j\mathsf{T}} E[\mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}}] \mathbf{n}_k^j - \mathbf{n}_k^{j\mathsf{T}} E[\mathbf{w}_k^j \mathbf{w}_{k-1}^{j\,\mathsf{T}}] \mathbf{C}_{kk-1}^{\mathsf{T}} \mathbf{n}_k^j \\
&\quad - \mathbf{n}_k^{j\mathsf{T}} \mathbf{C}_{kk-1} E[\mathbf{w}_{k-1}^j \mathbf{w}_k^{j\mathsf{T}}] \mathbf{n}_k^j + \mathbf{n}_k^{j\mathsf{T}} \mathbf{C}_{kk-1} E[\mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\,\mathsf{T}}] \mathbf{C}_{kk-1}^{\mathsf{T}} \mathbf{n}_k^j \\
&= \mathbf{n}_k^{j\mathsf{T}} \mathbf{R}_k^j \mathbf{n}_k^j + \mathbf{n}_k^{j\mathsf{T}} \mathbf{C}_{kk-1} \mathbf{R}_{k-1}^{jk-1} \mathbf{C}_{kk-1}^{\mathsf{T}} \mathbf{n}_k^j \\
&= \mathbf{n}_k^{j\mathsf{T}} (\mathbf{R}_k^j + \mathbf{C}_{kk-1} \mathbf{R}_{k-1}^{jk-1} \mathbf{C}_{kk-1}^{\mathsf{T}}) \mathbf{n}_k^j
\end{aligned}$$

It is possible to derive an expression for the error covariance, or more accurately, variance in this case, without making the point-to-point association assumption made during the point-to-point WOLATE derivation. This is a critical piece of information that arises naturally in our derivation. It implies that in cases where there is no guarantee that points observed at one instance are the same points observed in another instance, such as the case with

LIDAR measurements, point-to-plane registration should always be chosen over point-to-point registration.

The cost function of (5.15) can now be rewritten as

$$\begin{aligned} J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) &= \sum_{j=1}^{M} \frac{1}{2\sigma_k^{j\,2}} \hat{e}_k^{j\,2} \\ &= \sum_{j=1}^{M} \tfrac{1}{2} \hat{e}_k^j (\mathbf{n}_k^{j\,\mathsf{T}} (\mathbf{R}_k^j + \mathbf{C}_{kk-1} \mathbf{R}_{k-1}^{jk-1} \mathbf{C}_{kk-1}^{\mathsf{T}}) \mathbf{n}_k^j)^{-1} \hat{e}_k^j \\ &= \sum_{j=1}^{M} \tfrac{1}{2} \tilde{\mathbf{e}}_k^{j\mathsf{T}} \mathbf{n}_k^j (\mathbf{n}_k^{j\,\mathsf{T}} (\mathbf{R}_k^j + \mathbf{C}_{kk-1} \mathbf{R}_{k-1}^{jk-1} \mathbf{C}_{kk-1}^{\mathsf{T}}) \mathbf{n}_k^j)^{-1} \mathbf{n}_k^{j\,\mathsf{T}} \tilde{\mathbf{e}}_k^j \end{aligned}$$

Rewriting the Cost Function Using the Caley Transformation

Applying the Cayley Transformation (5.3) to $\mathbf{C}_{kk-1}$, the error residual term that was previously defined as

$$\tilde{\mathbf{e}}_k^j = \tilde{\mathbf{r}}_k^{q_j s_k} - (\mathbf{C}_{kk-1} \tilde{\mathbf{r}}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{s_k s_{k-1}})$$

can be rewritten as

$$\tilde{\mathbf{e}}_k^j = \tilde{\mathbf{r}}_k^{q_j s_k} - \left((\mathbf{1} + \mathbf{p}_k^\times)^{-1} (\mathbf{1} - \mathbf{p}_k^\times) \tilde{\mathbf{r}}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{s_k s_{k-1}} \right).$$

Similarly, by defining a new residual $\bar{\mathbf{e}}_k^j = (\mathbf{1} + \mathbf{p}_k^\times) \tilde{\mathbf{e}}_k^j$

$$\begin{aligned} \bar{\mathbf{e}}_k^j &= (\mathbf{1} + \mathbf{p}_k^\times) \tilde{\mathbf{r}}_k^{q_j s_k} - (\mathbf{1} - \mathbf{p}_k^\times) \tilde{\mathbf{r}}_{k-1}^{p_j s_{k-1}} - (\mathbf{1} + \mathbf{p}_k^\times) \mathbf{r}_k^{s_k s_{k-1}} \\ &= \mathbf{b}_k^j - \mathbf{A}_k^j \mathbf{x}_k. \end{aligned}$$

Using the relationship

$$\tilde{\mathbf{e}}_k^j = (\mathbf{1} + \mathbf{p}_k^\times)^{-1} \bar{\mathbf{e}}_k^j$$

The above cost function can be rewritten as

$$\begin{aligned} J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) &= \sum_{j=1}^{M} \frac{1}{2} \tilde{\mathbf{e}}_k^{j\mathsf{T}} \mathbf{n}_k^j {\sigma_k^j}^{-1} {\mathbf{n}_k^j}^{\mathsf{T}} \tilde{\mathbf{e}}_k^j \\ &= \sum_{j=1}^{M} \frac{1}{2} \bar{\mathbf{e}}_k^{j\mathsf{T}} (\mathbf{1} + \mathbf{p}_k^{\times})^{-\mathsf{T}} \mathbf{n}_k^j {\sigma_k^j}^{-1} {\mathbf{n}_k^j}^{\mathsf{T}} (\mathbf{1} + \mathbf{p}_k^{\times})^{-1} \bar{\mathbf{e}}_k^j \\ &= \sum_{j=1}^{M} \frac{1}{2} \bar{\mathbf{e}}_k^{j\mathsf{T}} \mathbf{W}_k^j \bar{\mathbf{e}}_k^j \end{aligned}$$

where $\mathbf{W}_k^j = (\mathbf{1} + \mathbf{p}_k^{\times})^{-\mathsf{T}} \mathbf{n}_k^j {\sigma_k^j}^{-1} {\mathbf{n}_k^j}^{\mathsf{T}} (\mathbf{1} + \mathbf{p}_k^{\times})^{-1}$.

5.3.5 Point-to-Line Registration Using OLATE

To the knowledge of the author, the only known mention of 3D point-to-line registration in literature is provided in [22], which formulates the point-to-line distance using a cross product and requiring nonlinear optimization to solve. In [57] a 2D point-to-line registration method is described that is solved by enforcing the constraint $\sin\theta^2 + \cos\theta^2 = 1$ using the Lagrange Multiplier method. However, no 3D solution is provided. The Iterative Closest Line (ICL) approach described in [29] is in fact a line-to-line registration method but is likely the most similar method to the method presented in this section. In deriving the OLATE-based point-to-line method, it will be shown that the 2D version is actually equivalent to 2D point-to-plane registration. Because no literature available to date describes 3D point-to-line registration in a rigorous way, the derivation provided in this section is particularly novel.

In the point-to-line formulation, the objective is to minimize the euclidean distance between q_j and the line formed by point p_j and p_l. Let $\hat{\mathbf{e}}_k^j = \mathbf{e}_k^j - \mathbf{a}_k^j {\mathbf{a}_k^j}^{\mathsf{T}} \mathbf{e}_k^j$. The cost function to minimize is

$$\begin{aligned} J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) &= \sum_{j=1}^{M} {\hat{\mathbf{e}}_k^j}^{\mathsf{T}} \hat{\mathbf{e}}_k^j \\ &= \sum_{j=1}^{M} (\mathbf{e}_k^j - \underbrace{\mathbf{a}_k^j {\mathbf{a}_k^j}^{\mathsf{T}}}_{\mathbf{D}_k^j} \mathbf{e}_k^j)^{\mathsf{T}} (\mathbf{e}_k^j - \underbrace{\mathbf{a}_k^j {\mathbf{a}_k^j}^{\mathsf{T}}}_{\mathbf{D}_k^j} \mathbf{e}_k^j) \\ &= \sum_{j=1}^{M} {\mathbf{e}_k^j}^{\mathsf{T}} (\mathbf{1} - \mathbf{D}_k^j)^{\mathsf{T}} (\mathbf{1} - \mathbf{D}_k^j) \mathbf{e}_k^j \end{aligned} \tag{5.16}$$

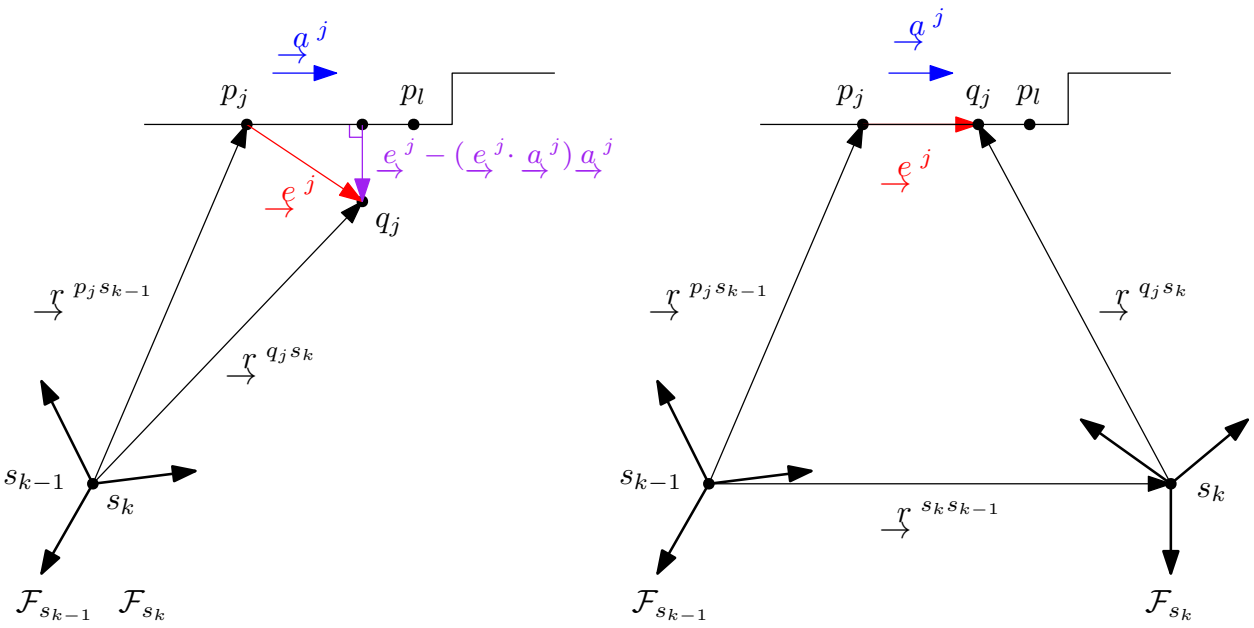


Figure 5.6: Point-to-line registration problem statement. Left: Frame-centric view before registration completion. Right: After point q_j has been registered.

where the residual is

$$\mathbf{e}_k^j = \mathbf{r}_k^{q_j s_k} - (\mathbf{1} + \mathbf{p}_k^\times)^{-1}(\mathbf{1} - \mathbf{p}_k^\times)\mathbf{r}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{s_k s_{k-1}},$$

Just as before, define a new residual $\bar{\mathbf{e}}_k^j = (\mathbf{1} + \mathbf{p}_k^\times)\mathbf{e}_k^j$, and after the same simplification steps made in (5.5), the expression for $\bar{\mathbf{e}}_k^j$ becomes

$$\begin{aligned}\bar{\mathbf{e}}_k^j &= (\mathbf{1} + \mathbf{p}_k^\times)\tilde{\mathbf{r}}_k^{q_j s_k} - (\mathbf{1} - \mathbf{p}_k^\times)\tilde{\mathbf{r}}_{k-1}^{p_j s_{k-1}} - (\mathbf{1} + \mathbf{p}_k^\times)\mathbf{r}_k^{s_k s_{k-1}} \\ &= \mathbf{b}_k^j - \mathbf{A}_k^j \mathbf{x}_k.\end{aligned}$$

Using the relationship

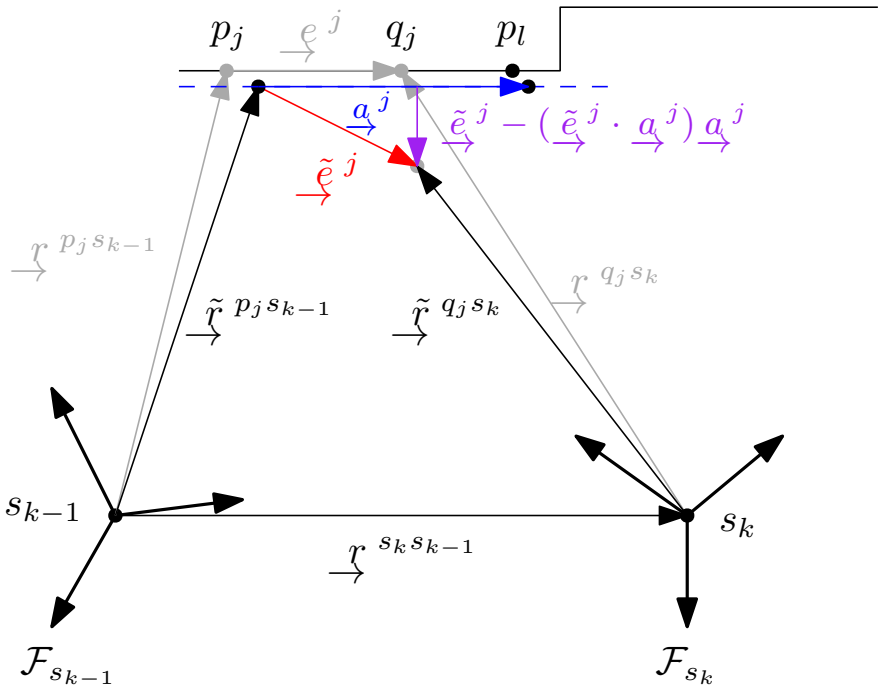
$$\mathbf{e}_k^j = (\mathbf{1} + \mathbf{p}_k^\times)^{-1}\bar{\mathbf{e}}_k^j,$$

the cost function of (5.16) becomes

$$\begin{aligned}J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) &= \sum_{j=1}^{M} \bar{\mathbf{e}}_k^{j\mathsf{T}}(\mathbf{1} + \mathbf{p}_k^\times)^{-\mathsf{T}}(\mathbf{1} - \mathbf{D}_k^j)^\mathsf{T}(\mathbf{1} - \mathbf{D}_k^j)(\mathbf{1} + \mathbf{p}_k^\times)^{-1}\bar{\mathbf{e}}_k^j \\ &= \sum_{j=1}^{M} \bar{\mathbf{e}}_k^{j\mathsf{T}}(\mathbf{1} + \mathbf{p}_k^\times)^{-\mathsf{T}}(\mathbf{1} - \mathbf{D}_k^j)(\mathbf{1} + \mathbf{p}_k^\times)^{-1}\bar{\mathbf{e}}_k^j,\end{aligned}$$

where $(\mathbf{1} - \mathbf{D}_k^j)^\mathsf{T}(\mathbf{1} - \mathbf{D}_k^j) = (\mathbf{1} - \mathbf{D}_k^j)$. For the 2D case, $(\mathbf{1} - \mathbf{D}_k^j) = \mathbf{N}_k^j$ because every unit vector parallel to a line has a unique orthonormal complement that happens to be the normal vector of that line. This statement is no longer valid for the 3D case where planes and lines are different geometric entities and where every unit vector has an infinite number of orthonormal complements.

5.3.6 Weighted OLATE-Based Point-to-Line Registration



Once again, in the weighted formulation, probabilistic measurements are assumed, that is

$$\begin{aligned}\tilde{\mathbf{r}}_k^{q_j s_k} &= \mathbf{r}_k^{q_j s_k} + \mathbf{w}_k^{q_j},\\ \tilde{\mathbf{r}}_k^{p_j s_{k-1}} &= \mathbf{r}_{k-1}^{p_j s_{k-1}} + \mathbf{w}_{k-1}^{p_j}\end{aligned}$$

where $\mathbf{w}_{k-1}^{p_j} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{k-1}^j)$ and $\mathbf{w}_k^{q_j} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k^j)$.

Given the unit vector a^j parallel to the line of interest, the cost function to minimize can be formulated as follows.

$$J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) = \sum_{j=1}^{M} \frac{1}{2} (\underbrace{\tilde{\mathbf{e}}_k^j - \mathbf{a}_k^j \mathbf{a}_k^{j\mathsf{T}} \tilde{\mathbf{e}}_k^j}_{\hat{\mathbf{e}}_k^{j\mathsf{T}}})^\mathsf{T} (\underbrace{\tilde{\mathbf{e}}_k^j - \mathbf{a}_k^j \mathbf{a}_k^{j\mathsf{T}} \tilde{\mathbf{e}}_k^j}_{\hat{\mathbf{e}}_k^j})$$

where $\hat{\mathbf{e}}_k^j = \tilde{\mathbf{e}}_k^j - \mathbf{a}_k^j \mathbf{a}_k^{j\mathsf{T}} \tilde{\mathbf{e}}_k^j$ is the projection of vector $\tilde{\mathbf{e}}_k^j$ along the direction normal to the line direction $\mathbf{a}_k^j$.

Next, derive the covariance matrix associated with this error where the covariance is defined as $E[\hat{\mathbf{e}}_k^j \hat{\mathbf{e}}_k^{j\mathsf{T}}]$. First note that

$$\begin{aligned}
\tilde{\mathbf{e}}_k^j &= \tilde{\mathbf{r}}_k^{q_j s_k} - (\mathbf{C}_{kk-1} \tilde{\mathbf{r}}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{s_k s_{k-1}}) \\
&= \mathbf{r}_k^{q_j s_k} - (\mathbf{C}_{kk-1} \mathbf{r}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{s_k s_{k-1}}) + \mathbf{w}_k^{q_j} - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^{p_j} \\
&= \mathbf{e}_k^j + \mathbf{w}_k^{q_j} - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^{p_j}.
\end{aligned}$$

The covariance associated with the error is then

$$\begin{aligned}
\boldsymbol{\Sigma}_k^j &= E[\hat{\mathbf{e}}_k^j \hat{\mathbf{e}}_k^{j\mathsf{T}}] \\
&= E[(\tilde{\mathbf{e}}_k^j - \mathbf{a}_k^j \mathbf{a}_k^{j\mathsf{T}} \tilde{\mathbf{e}}_k^j)(\tilde{\mathbf{e}}_k^j - \mathbf{a}_k^j \mathbf{a}_k^{j\mathsf{T}} \tilde{\mathbf{e}}_k^j)^{\mathsf{T}}] \\
&= E[\tilde{\mathbf{e}}_k^j \tilde{\mathbf{e}}_k^{j\mathsf{T}} - \tilde{\mathbf{e}}_k^j \tilde{\mathbf{e}}_k^{j\mathsf{T}} \underbrace{\mathbf{a}_k^j \mathbf{a}_k^{j\mathsf{T}}}_{\mathbf{D}_k^j} - \underbrace{\mathbf{a}_k^j \mathbf{a}_k^{j\mathsf{T}}}_{\mathbf{D}_k^j} \tilde{\mathbf{e}}_k^j \tilde{\mathbf{e}}_k^{j\mathsf{T}} + \underbrace{\mathbf{a}_k^j \mathbf{a}_k^{j\mathsf{T}}}_{\mathbf{D}_k^j} \tilde{\mathbf{e}}_k^j \tilde{\mathbf{e}}_k^{j\mathsf{T}} \underbrace{\mathbf{a}_k^j \mathbf{a}_k^{j\mathsf{T}}}_{\mathbf{D}_k^j}] \\
&= E[(\mathbf{e}_k^j + \mathbf{w}_k^j - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j)(\mathbf{e}_k^j + \mathbf{w}_k^j - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j)^{\mathsf{T}} \\
&\quad - (\mathbf{e}_k^j + \mathbf{w}_k^j - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j)(\mathbf{e}_k^j + \mathbf{w}_k^j - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j)^{\mathsf{T}} \mathbf{D}_k^j \\
&\quad - \mathbf{D}_k^j (\mathbf{e}_k^j + \mathbf{w}_k^j - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j)(\mathbf{e}_k^j + \mathbf{w}_k^j - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j)^{\mathsf{T}} \\
&\quad + \mathbf{D}_k^j (\mathbf{e}_k^j + \mathbf{w}_k^j - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j)(\mathbf{e}_k^j + \mathbf{w}_k^j - \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j)^{\mathsf{T}} \mathbf{D}_k^j] \\
&= E[\mathbf{e}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{w}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{e}_k^{j\mathsf{T}} \\
&\quad + \mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}} - \cancel{\mathbf{w}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}} - \cancel{\mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_k^{j\mathsf{T}}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}] \\
&\quad - E[\mathbf{e}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{w}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{e}_k^{j\mathsf{T}} \\
&\quad + \mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}} - \cancel{\mathbf{w}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}} - \cancel{\mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_k^{j\mathsf{T}}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}] \mathbf{D}_k^j \\
&\quad - \mathbf{D}_k^j E[\mathbf{e}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{w}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{e}_k^{j\mathsf{T}} \\
&\quad + \mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}} - \cancel{\mathbf{w}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}} - \cancel{\mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_k^{j\mathsf{T}}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}] \\
&\quad + \mathbf{D}_k^j E[\mathbf{e}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{w}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{e}_k^{j\mathsf{T}} \\
&\quad + \mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}} - \cancel{\mathbf{w}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}} - \cancel{\mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_k^{j\mathsf{T}}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}] \mathbf{D}_k^j
\end{aligned}$$

At this stage, it was possible to cross out terms using the assumption that the two noise vectors are uncorrelated, that is $E[\mathbf{w}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}}] = E[\mathbf{w}_{k-1}^j \mathbf{w}_k^{j\mathsf{T}}] = \mathbf{0}$. Also note that $\mathbf{e}_k^{j\mathsf{T}} \mathbf{D}_k^j = \mathbf{e}_k^{j\mathsf{T}}$ and $\mathbf{D}_k^j \mathbf{e}_k^j = \mathbf{e}_k^j$, allowing an additional simplification to take place, that being

$$
\begin{aligned}
\boldsymbol{\Sigma}_k^j &= E[\mathbf{e}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{w}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{e}_k^{j\mathsf{T}} \\
&\quad + \mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}] \\
&\quad - E[\mathbf{e}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{w}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{e}_k^{j\mathsf{T}} \\
&\quad + \mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}] \mathbf{D}_k^j \\
&\quad - \mathbf{D}_k^j E[\mathbf{e}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{w}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{e}_k^{j\mathsf{T}} \\
&\quad + \mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}] \\
&\quad + \mathbf{D}_k^j E[\mathbf{e}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{w}_k^j \mathbf{e}_k^{j\mathsf{T}} + \mathbf{e}_k^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{e}_k^{j\mathsf{T}} \\
&\quad + \mathbf{w}_k^j \mathbf{w}_k^{j\mathsf{T}} + \mathbf{C}_{kk-1} \mathbf{w}_{k-1}^j \mathbf{w}_{k-1}^{j\mathsf{T}} \mathbf{C}_{kk-1}^{\mathsf{T}}] \mathbf{D}_k^j \\
&= (\mathbf{R}_k^j + \mathbf{C}_{kk-1} \mathbf{R}_{k-1}^{jk-1} \mathbf{C}_{kk-1}^{\mathsf{T}}) - (\mathbf{R}_k^j \mathbf{D}_k^j + \mathbf{C}_{kk-1} \mathbf{R}_{k-1}^{jk-1} \mathbf{C}_{kk-1}^{\mathsf{T}} \mathbf{D}_k^j) \\
&\quad - (\mathbf{D}_k^j \mathbf{R}_k^j + \mathbf{D}_k^j \mathbf{C}_{kk-1} \mathbf{R}_{k-1}^{jk-1} \mathbf{C}_{kk-1}^{\mathsf{T}}) + (\mathbf{D}_k^j \mathbf{R}_k^j \mathbf{D}_k^j + \mathbf{D}_k^j \mathbf{C}_{kk-1} \mathbf{R}_{k-1}^{jk-1} \mathbf{C}_{kk-1}^{\mathsf{T}} \mathbf{D}_k^j) \\
&= (\mathbf{1} - \mathbf{D}_k^j)(\mathbf{R}_k^j + \mathbf{C}_{kk-1} \mathbf{R}_{k-1}^{jk-1} \mathbf{C}_{kk-1}^{\mathsf{T}})(\mathbf{1} - \mathbf{D}_k^j).
\end{aligned}
$$

where terms that cancel with each other have been color coded for clarity.

Note that unlike the point-to-plane case where the variance σ_k^j is a scalar and therefore invertible, the covariance $\boldsymbol{\Sigma}_k^j$ in the point-to-line case is a matrix. Because the matrix $\mathbf{D}_k^j$ is a projection matrix, $\mathbf{1} - \mathbf{D}_k^j$ is singular, which is to say $\boldsymbol{\Sigma}_k^j$ is also singular and non-invertible. To get around this problem, simply perturb $\boldsymbol{\Sigma}_k^j$ with a small Symmetric Positive Definite matrix $\boldsymbol{\Delta}$, effectively rendering it invertible. Note that the two matrices to the left and right of the inverse covariance matrix, namely, $(\mathbf{1} - \mathbf{D}_k^j)^{\mathsf{T}}$ and $(\mathbf{1} - \mathbf{D}_k^j)$ must not be perturbed as they serve to project the weight matrix into the correct subspace.

Adding the aforementioned perturbation to the covariance matrix allows the cost function to be rewritten as

$$
\begin{aligned}
J_k(\mathbf{r}_k^{s_k s_{k-1}}, \mathbf{C}_{kk-1}) &= \sum_{j=1}^{M} \tfrac{1}{2} \hat{\mathbf{e}}_k^{j\mathsf{T}} (\boldsymbol{\Sigma}_k^j + \boldsymbol{\Delta})^{-1} \hat{\mathbf{e}}_k^j \\
&= \sum_{j=1}^{M} \tfrac{1}{2} \hat{\mathbf{e}}_k^{j\mathsf{T}} \left((\mathbf{1} - \mathbf{D}_k^j)(\mathbf{R}_k^j + \mathbf{C}_{kk-1} \mathbf{R}_{k-1}^{jk-1} \mathbf{C}_{kk-1}^{\mathsf{T}})(\mathbf{1} - \mathbf{D}_k^j) + \boldsymbol{\Delta} \right)^{-1} \hat{\mathbf{e}}_k^j \\
&= \sum_{j=1}^{M} \tfrac{1}{2} \tilde{\mathbf{e}}_k^{j\mathsf{T}} (\mathbf{1} - \mathbf{D}_k^j)^{\mathsf{T}} \left((\mathbf{1} - \mathbf{D}_k^j)(\mathbf{R}_k^j + \mathbf{C}_{kk-1} \mathbf{R}_{k-1}^{jk-1} \mathbf{C}_{kk-1}^{\mathsf{T}})(\mathbf{1} - \mathbf{D}_k^j) + \boldsymbol{\Delta} \right)^{-1} (\mathbf{1} - \mathbf{D}_k^j) \tilde{\mathbf{e}}_k^j
\end{aligned}
$$

Rewriting the Cost Function Using the Caley Transformation

Applying the Cayley Transformation (5.3) to $\mathbf{C}_{kk-1}$, the error residual term that was previously defined as

$$\tilde{\mathbf{e}}_k^j = \tilde{\mathbf{r}}_k^{q_j s_k} - (\mathbf{C}_{kk-1}\tilde{\mathbf{r}}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{s_k s_{k-1}})$$

can be rewritten as

$$\tilde{\mathbf{e}}_k^j = \tilde{\mathbf{r}}_k^{q_j s_k} - \left((\mathbf{1}+\mathbf{p}_k^\times)^{-1}(\mathbf{1}-\mathbf{p}_k^\times)\tilde{\mathbf{r}}_{k-1}^{p_j s_{k-1}} - \mathbf{r}_k^{kk-1}\right).$$

Similarly, by defining a new residual $\bar{\mathbf{e}}_k^j = (\mathbf{1}+\mathbf{p}_k^\times)\tilde{\mathbf{e}}_k^j$

$$\begin{aligned}\bar{\mathbf{e}}_k^j &= (\mathbf{1}+\mathbf{p}_k^\times)\tilde{\mathbf{r}}_k^{q_j s_k} - (\mathbf{1}-\mathbf{p}_k^\times)\tilde{\mathbf{r}}_{k-1}^{p_j s_{k-1}} - (\mathbf{1}+\mathbf{p}_k^\times)\mathbf{r}_k^{s_k s_{k-1}} \\ &= \mathbf{b}_k^j - \mathbf{A}_k^j \mathbf{x}_k.\end{aligned}$$

Using the relationship

$$\tilde{\mathbf{e}}_k^j = (\mathbf{1}+\mathbf{p}_k^\times)^{-1}\bar{\mathbf{e}}_k^j$$

The above cost function can be rewritten as

$$\begin{aligned}J &= \sum_{j=1}^{M} \frac{1}{2}\tilde{\mathbf{e}}_k^{j\mathsf{T}}(\mathbf{1}-\mathbf{D}_k^j)^\mathsf{T}\left(\mathbf{\Sigma}_k^j + \mathbf{\Delta}\right)^{-1}(\mathbf{1}-\mathbf{D}_k^j)\tilde{\mathbf{e}}_k^j \\ &= \sum_{j=1}^{M} \frac{1}{2}\bar{\mathbf{e}}_k^{j\mathsf{T}}(\mathbf{1}+\mathbf{p}_k^\times)^{-\mathsf{T}}(\mathbf{1}-\mathbf{D}_k^j)^\mathsf{T}\left(\mathbf{\Sigma}_k^j + \mathbf{\Delta}\right)^{-1}(\mathbf{1}-\mathbf{D}_k^j)(\mathbf{1}+\mathbf{p}_k^\times)^{-1}\bar{\mathbf{e}}_k^j \\ &= \sum_{j=1}^{M} \frac{1}{2}\bar{\mathbf{e}}_k^{j\mathsf{T}}\mathbf{W}_k^j\bar{\mathbf{e}}_k^j\end{aligned}$$

where $\mathbf{W}_k^j = (\mathbf{1}+\mathbf{p}_k^\times)^{-\mathsf{T}}(\mathbf{1}-\mathbf{D}_k^j)^\mathsf{T}\left(\mathbf{\Sigma}_k^j + \mathbf{\Delta}\right)^{-1}(\mathbf{1}-\mathbf{D}_k^j)(\mathbf{1}+\mathbf{p}_k^\times)^{-1}$.

5.4 Total Registration Using WOLATE

Now that the point-to-point, point-to-plane, and point-to-line WOLATE algorithms have been derived, an algorithm that encompasses all three problems into a single problem can be formulated. This method is termed Total Registration as it performs registration an all point types simultaneously. Before proceeding, a table summarizing the three methods is provided in Table 5.1.

Table 5.1: Summary of OLATE and WOLATE-based registration cost functions

	Regular Cost Function	Weighted Cost Function
pt-pt	$\frac{1}{2}{\mathbf{e}_k^j}^{\mathsf{T}}\mathbf{e}_k^j$	$\frac{1}{2}{\tilde{\mathbf{e}}_k^j}^{\mathsf{T}}{\mathbf{S}_k^j}^{-1}\tilde{\mathbf{e}}_k^j$
pt-line	$\frac{1}{2}{\mathbf{e}_k^j}^{\mathsf{T}}(\mathbf{1}-\mathbf{D}_k^j)\mathbf{e}_k^j$	$\frac{1}{2}{\tilde{\mathbf{e}}_k^j}^{\mathsf{T}}(\mathbf{1}-\mathbf{D}_k^j)^{\mathsf{T}}\left((\mathbf{1}-\mathbf{D}_k^j)\mathbf{S}_k^j(\mathbf{1}-\mathbf{D}_k^j)+\boldsymbol{\Delta}\right)^{-1}(\mathbf{1}-\mathbf{D}_k^j)^{\mathsf{T}}\tilde{\mathbf{e}}_k^j$
pt-plane	$\frac{1}{2}{\mathbf{e}_k^j}^{\mathsf{T}}\mathbf{N}_k^j\mathbf{e}_k^j$	$\frac{1}{2}{\tilde{\mathbf{e}}_k^j}^{\mathsf{T}}\mathbf{n}_k^j\left({\mathbf{n}_k^j}^{\mathsf{T}}\mathbf{S}_k^j\mathbf{n}_k^j\right)^{-1}{\mathbf{n}_k^j}^{\mathsf{T}}\tilde{\mathbf{e}}_k^j$

where $\mathbf{S}_k^j = \mathbf{R}_k^j + \mathbf{C}_{kk-1}\mathbf{R}_{k-1}^{jk-1}\mathbf{C}_{kk-1}^{\mathsf{T}}$ and $\boldsymbol{\Delta}$ is a small Positive Definite matrix.

Consider that for a given transformation between two poses $\mathbf{T}_{ib_{k-1}}$ and $\mathbf{T}_{ib_k}$, the point clouds viewed from the two poses, namely $\mathbf{r}_{b_{k-1}}^{p_j b_{k-1}}$ and $\mathbf{r}_{b_k}^{p_j b_k}$ are given. Also suppose that after undergoing the feature extraction steps defined in Section 4.6, the edge points $\mathfrak{e}_{b_k}^{q_j b_k}$, surface points $\mathfrak{s}_{b_k}^{q_j b_k}$, and feature points $\mathbf{r}_{b_k}^{q_j b_k}$ along with their corresponding line vectors $\mathbf{a}_k^j$, normal vectors $\mathbf{n}_k^j$, and points $\mathbf{r}_{b_{k-1}}^{p_j b_{k-1}}$ are available. Note that here, the term "feature points" refers to points of interest that can be corresponded exactly to some other feature point. In 2D SLAM, these can be corner points. Finally, suppose that the uncertainty of the sensor used has been modelled according to Chapter 3 and the covariance matrices $\mathbf{R}_{k-1}^{jk-1}$ and $\mathbf{R}_k^{jk}$ of feature points, the covariance matrices $\boldsymbol{\mathcal{E}}_{k-1}^{jk-1}$ and $\boldsymbol{\mathcal{E}}_k^{jk}$ of edge points, and the covariance matrices $\boldsymbol{\mathcal{S}}_{k-1}^{jk-1}$ and $\boldsymbol{\mathcal{S}}_k^{jk}$ of surface points have been retrieved. The Total Registration algorithm is given in Algorithm 5.1.

Algorithm 5.1

function $\mathbf{C}_{kk-1}$, $\mathbf{r}_k^{kk-1}$ = TotalRegistration($\mathbf{r}_k^{q_j b_k}$, $\mathfrak{s}_k^{q_j b_k}$, $\mathfrak{e}_k^{q_j b_k}$, $\mathbf{r}_{k-1}^{p_j b_{k-1}}$, $\mathfrak{s}_{k-1}^{p_j b_{k-1}}$, $\mathfrak{e}_{k-1}^{p_j b_{k-1}}$, $\mathbf{n}_k^j$, $\mathbf{a}_k^j$, $\mathbf{R}_{k-1}^{jk-1}$, $\mathbf{R}_k^{jk}$,$\boldsymbol{\mathcal{S}}_{k-1}^{jk-1}$, $\boldsymbol{\mathcal{S}}_k^{jk}$,$\boldsymbol{\mathcal{E}}_{k-1}^{jk-1}$, $\boldsymbol{\mathcal{E}}_k^{jk}$)

Form the column matrix $\mathbf{v}_k = \begin{bmatrix} \mathbf{v}_k^{\mathbf{r}} \\ \mathbf{v}_k^{\mathfrak{s}} \\ \mathbf{v}_k^{\mathfrak{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{k-1}^{p_j b_{k-1}} + \mathbf{r}_k^{q_j b_k} \\ \mathfrak{s}_{k-1}^{p_j b_{k-1}} + \mathfrak{s}_k^{q_j b_k} \\ \mathfrak{e}_{k-1}^{p_j b_{k-1}} + \mathfrak{e}_k^{q_j b_k} \end{bmatrix}$

Form the matrix $\mathbf{A}_k$ where $\mathbf{A}_k = \begin{bmatrix} {\mathbf{A}_k}^{\mathbf{r}} \\ {\mathbf{A}_k}^{\mathfrak{s}} \\ {\mathbf{A}_k}^{\mathfrak{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & {\mathbf{v}_k^j}^{\mathbf{r}\times} \\ \mathbf{1} & {\mathbf{v}_k^j}^{\mathfrak{s}\times} \\ \mathbf{1} & {\mathbf{v}_k^j}^{\mathfrak{e}\times} \end{bmatrix}$

Form the column matrix $\mathbf{b}_k$ where $\mathbf{b}_k = \begin{bmatrix} \mathbf{b}_k^{\mathbf{r}} \\ \mathbf{b}_k^{\mathfrak{s}} \\ \mathbf{b}_k^{\mathfrak{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{k-1}^{p_j b_{k-1}} - \mathbf{r}_k^{q_j b_k} \\ \mathfrak{s}_{k-1}^{p_j b_{k-1}} - \mathfrak{s}_k^{q_j b_k} \\ \mathfrak{e}_{k-1}^{p_j b_{k-1}} - \mathfrak{e}_k^{q_j b_k} \end{bmatrix}$

iteration = 0

while condition $> \epsilon$ **do**

  for all feature points **do**

    if iteration == 1 **then**

      Compute $\boldsymbol{\Sigma}_k^j = \mathbf{R}_k^{jk} + \mathbf{R}_{k-1}^{jk-1}$

      $\mathbf{W}_k^j \leftarrow {\boldsymbol{\Sigma}_k^j}^{-1}$

    else

      Compute $\boldsymbol{\Sigma}_k^j = \mathbf{R}_k^{jk} + \mathbf{C}_{kk-1}\mathbf{R}_{k-1}^{jk-1}\mathbf{C}_{kk-1}^{\mathsf{T}}$

      $\mathbf{W}_k^j \leftarrow (\mathbf{1} + \mathbf{p}_k^{\times})^{-\mathsf{T}} {\boldsymbol{\Sigma}_k^j}^{-1} (\mathbf{1} + \mathbf{p}_k^{\times})^{-1}$

    end if

  end for

  for all surface points **do**

    if iteration == 1 **then**

      Compute $\sigma_k^j = {\mathbf{n}_k^j}^{\mathsf{T}} (\boldsymbol{\mathcal{S}}_k^{jk} + \boldsymbol{\mathcal{S}}_{k-1}^{jk-1}) \mathbf{n}_k^j$

      $\mathbf{W}_k^j \leftarrow \mathbf{n}_k^j {\sigma_k^j}^{-1} {\mathbf{n}_k^j}^{\mathsf{T}}$

    else

      Compute $\sigma_k^j = {\mathbf{n}_k^j}^{\mathsf{T}} (\boldsymbol{\mathcal{S}}_k^{jk} + \mathbf{C}_{kk-1}\boldsymbol{\mathcal{S}}_{k-1}^{jk-1}\mathbf{C}_{kk-1}^{\mathsf{T}}) \mathbf{n}_k^j$

      $\mathbf{W}_k^j \leftarrow (\mathbf{1} + \mathbf{p}_k^{\times})^{-\mathsf{T}} \mathbf{n}_k^j {\sigma_k^j}^{-1} {\mathbf{n}_k^j}^{\mathsf{T}} (\mathbf{1} + \mathbf{p}_k^{\times})^{-1}$

    end if

  end for

  for all edge points **do**

    if iteration == 1 **then**

      Compute $\boldsymbol{\Sigma}_k^j = (\mathbf{1} - \mathbf{D}_k^j)^{\mathsf{T}} (\boldsymbol{\mathcal{E}}_k^{jk} + \boldsymbol{\mathcal{E}}_{k-1}^{jk-1}) (\mathbf{1} - \mathbf{D}_k^j) + \boldsymbol{\Delta}$

      $\mathbf{W}_k^j \leftarrow (\mathbf{1} - \mathbf{D}_k^j)^{\mathsf{T}} {\boldsymbol{\Sigma}_k^j}^{-1} (\mathbf{1} - \mathbf{D}_k^j)$

    else

      Compute $\boldsymbol{\Sigma}_k^j = (\mathbf{1} - \mathbf{D}_k^j)^{\mathsf{T}} (\boldsymbol{\mathcal{E}}_k^{jk} + \mathbf{C}_{kk-1}\boldsymbol{\mathcal{E}}_{k-1}^{jk-1}\mathbf{C}_{kk-1}^{\mathsf{T}}) (\mathbf{1} - \mathbf{D}_k^j) + \boldsymbol{\Delta}$

      $\mathbf{W}_k^j \leftarrow (\mathbf{1} + \mathbf{p}_k^{\times})^{-\mathsf{T}} (\mathbf{1} - \mathbf{D}_k^j)^{\mathsf{T}} {\boldsymbol{\Sigma}_k^j}^{-1} (\mathbf{1} - \mathbf{D}_k^j) (\mathbf{1} + \mathbf{p}_k^{\times})^{-1}$

    end if

  end for

$\mathbf{W}_k = \begin{bmatrix} \mathbf{W}_k & & \\ & \mathbf{W}_k & \\ & & \mathbf{W}_k \end{bmatrix}$

$\mathbf{x} = \begin{bmatrix} \mathbf{t}_k \\ \mathbf{p}_k \end{bmatrix} = (\mathbf{A}_k^\top \mathbf{W}_k \mathbf{A}_k)^{-1} \mathbf{A}_k^\top \mathbf{W}_k \mathbf{b}_k$

$\mathbf{r}_k^{s_k s_{k-1}} = -(\mathbf{1} + \mathbf{p}_k^\times)^{-1} \mathbf{t}_k$

$\mathbf{C}_{kk-1} = (\mathbf{1} + \mathbf{p}_k^\times)^{-1}(\mathbf{1} - \mathbf{p}_k^\times)$

condition $\leftarrow$ ComputeConvergenceCondition()

iteration++

end while

end function

For 2D LIDAR odometry, it is recommended to treat corner points as feature points with point-to-point correspondence since the same point will appear as a corner regardless of pose. For 3D LIDAR odometry, only edge points and surface points should be used since regular points have no guarantee of being associated with their closest-neighbour.

5.4.1 Constrained OLATE

Another useful feature of a linear solution to the pose estimation problem, that being OLATE or WOLATE, is that it becomes easy to introduce constraints to the problem. Suppose the same least squares problem as the previous section, namely, $\min \|\mathbf{Ax} - \mathbf{b}\|_2^2$ where $\mathbf{A} \in \mathbb{R}^{m \times n}$ now has an equality constraint $\mathbf{Bx} = \mathbf{d}$, so that the optimization problem can be written as

$$\begin{aligned} &\min \|\mathbf{Ax} - \mathbf{b}\|_2^2 \\ &\text{s.t. } \mathbf{Bx} = \mathbf{d}. \end{aligned}$$

Such a constraint can arise when a better estimate for one of the states is available. For example, segmenting the ground plane from a point cloud and separately performing point-to-plane scan matching can yield a more accurate estimate of the z translation component. If this were the case, the predicted pose would be initialized with the a-priori known $\mathbf{r}_k^{s_k s_{k-1}}$ and the constrained would then be $\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ and $\mathbf{b} = (\mathbf{1} + \mathbf{p}^\times)\mathbf{r}_k^{s_k s_{k-1}} = \mathbf{0}$. Note that since this is usually applied within an ICP algorithm where the pose estimate is incrementally computed, setting $\mathbf{d}$ to zero is equivalent to preventing any translation in the z direction throughout the course of the ICP iterations.

This equality-constrained least squares problem can be solved a number of ways. The easiest and most natural way is via the Lagrange Multiplier method [58]. First, form the

Lagrangian function as

$$\begin{aligned} L(\mathbf{x}, \boldsymbol{\lambda}) &= \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \boldsymbol{\lambda}^\mathsf{T}(\mathbf{B}\mathbf{x} - \mathbf{d}) \\ &= \mathbf{x}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{x} - 2\mathbf{b}^\mathsf{T}\mathbf{A} - \mathbf{b}^\mathsf{T}\mathbf{b} + \boldsymbol{\lambda}^\mathsf{T}\mathbf{C}\mathbf{x} - \boldsymbol{\lambda}^\mathsf{T}\mathbf{d}. \end{aligned}$$

Next, minimize the Lagrangian with respect to $\mathbf{x}$ and $\boldsymbol{\lambda}$.

$$\begin{aligned} \frac{\delta L}{\delta \mathbf{x}} = 2\mathbf{x}^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A} - 2\mathbf{b}^\mathsf{T}\mathbf{A} + \boldsymbol{\lambda}^\mathsf{T}\mathbf{C} = \mathbf{0} \\ \frac{\delta L}{\delta \boldsymbol{\lambda}} = \mathbf{C}\mathbf{x} - \mathbf{d} = \mathbf{0}. \end{aligned}$$

Transposing and then simplifying this system of linear equations leads to

$$\begin{bmatrix} 2\mathbf{A}^\mathsf{T}\mathbf{A} & \mathbf{C}^\mathsf{T} \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} 2\mathbf{A}^\mathsf{T}\mathbf{b} \\ \mathbf{d} \end{bmatrix}, \tag{5.17}$$

yielding the Lagrange Multiplier solution, also known as the KKT solution to the equality-constrained least squares problem of constrained OLATE.

Another way to solve this equality-constrained least squares problem is to derive an equivalent unconstrained least squares problem of lower dimension. Using the method of direct elimination [42], first compute a QR decomposition of $\mathbf{B}$. Specifically, there exists an orthogonal matrix $\mathbf{Q}$ and a permutation matrix $\mathbf{P}$ such that

$$\mathbf{Q}^\mathsf{T}\mathbf{B}\mathbf{P} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

where $\mathbf{R}_{11} \in \mathbb{R}^{r \times r}$, $r = \text{rank}(\mathbf{B})$.

We can then substitute the factorized form of $\mathbf{B}$ into the equality constraint to obtain

$$\mathbf{Q} \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{P}^\mathsf{T}\mathbf{x} = \mathbf{d},$$

which can be simplified to

$$\begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \bar{\mathbf{x}} = \bar{\mathbf{d}}$$

where

$$\bar{\mathbf{d}} = \mathbf{Q}^{\mathsf{T}}\mathbf{d} = \begin{bmatrix} \bar{\mathbf{d}}_1 \\ \bar{\mathbf{d}}_2 \end{bmatrix} \quad , \quad \bar{\mathbf{d}}_1 = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \end{bmatrix} \bar{\mathbf{x}} \quad \text{, and} \quad \bar{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{x}. \tag{5.18}$$

The matrix $\mathbf{A}$ can then be permutated by substituting the expression for $\bar{\mathbf{x}}$, namely

$$\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{A}\mathbf{P}\bar{\mathbf{x}} - \mathbf{b} = \bar{\mathbf{A}}\bar{\mathbf{x}} - \mathbf{b} = \begin{bmatrix} \bar{\mathbf{A}}_1 & \bar{\mathbf{A}}_2 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_1 \\ \bar{\mathbf{x}}_2 \end{bmatrix} - \mathbf{b}. \tag{5.19}$$

Equation (5.18) can then be substituted into (5.19) to eliminate the $\bar{\mathbf{x}}_1$ variables. Noting that $\bar{\mathbf{x}}_1 = \mathbf{R}_{11}^{-1}(\bar{\mathbf{d}}_1 - \mathbf{R}_{12}\bar{\mathbf{x}}_2)$, proceed to isolate for $\bar{\mathbf{x}}_2$ to get

$$\begin{aligned} \mathbf{A}\mathbf{x} - \mathbf{b} &= \bar{\mathbf{A}}_1(\mathbf{R}_{11}^{-1}(\bar{\mathbf{d}}_1 - \mathbf{R}_{12}\bar{\mathbf{x}}_2)) + \bar{\mathbf{A}}_2\bar{\mathbf{x}}_2 - \mathbf{b} \\ &= (\bar{\mathbf{A}}_2 - \bar{\mathbf{A}}_1\mathbf{R}_{11}^{-1}\mathbf{R}_{12})\bar{\mathbf{x}}_2 - (\mathbf{b} - \bar{\mathbf{A}}_1\mathbf{R}_{11}^{-1}\bar{\mathbf{d}}_1). \end{aligned}$$

The reduced unconstrained least squares problem is thus

$$\min_{\bar{\mathbf{x}}_2} \left\| \hat{\mathbf{A}}_2\bar{\mathbf{x}}_2 - \hat{\mathbf{b}} \right\|^2 \tag{5.20}$$

where

$$\hat{\mathbf{A}}_2 = \bar{\mathbf{A}}_2 - \bar{\mathbf{A}}_1\mathbf{R}_{11}^{-1}\mathbf{R}_{12} \quad \text{and} \quad \hat{\mathbf{b}} = \mathbf{b} - \bar{\mathbf{A}}_1\mathbf{R}_{11}^{-1}\bar{\mathbf{d}}_1$$

If $\mathbf{B}$ has linearly independent rows, that is, $\text{rank}(\mathbf{B}) = p$ and $\text{rank}\begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix} = n$, it is possible to compute the QR decomposition of $\hat{\mathbf{A}}_2$ to get

$$\hat{\mathbf{A}}_2 = \mathbf{Q}\begin{bmatrix} \mathbf{R}_{22} \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{Q}^{\mathsf{T}}\hat{\mathbf{b}} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix}.$$

Next, solve for $\bar{\mathbf{x}}$ via

$$\begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} \bar{\mathbf{x}} = \begin{bmatrix} \bar{\mathbf{d}}_1 \\ \mathbf{c}_1 \end{bmatrix}.$$

Finally, $\mathbf{x} = \mathbf{P}\bar{\mathbf{x}}$. After solving for $\mathbf{x}$, $\mathbf{r}_k^{s_k s_{k-1}}$ and $\mathbf{C}_{kk-1}$ can be retrieved as usual.

More commonly, it is possible to obtain accurate pitch and roll estimates via an IMU. In such a case, it may be advantageous to constrain the rodriguez parameters associated

with roll and pitch. To do this, first convert the desired Euler angles to a DCM, followed by another conversion to Rodriguez parameters. In the case where roll and pitch are constrained,

$$\text{yaw} = \psi, \quad \text{pitch} = \theta = 0, \quad \text{roll} = \phi = 0$$

Their respective basic rotation matrices being

$$\mathbf{C}_1(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \quad \mathbf{C}_2(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix},$$

$$\mathbf{C}_3(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The full DCM is thus

$$\mathbf{C}_a b = \mathbf{C}_3(\psi)\mathbf{C}_2(\theta)\mathbf{C}_1(\phi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

From the DCM, the Rodriguez parameters can be obtained as

$$\begin{aligned} \phi &= \cos^{-1}\left(\frac{\mathbf{C}_{11} + \mathbf{C}_{22} + \mathbf{C}_{33} - 1}{2}\right) = \psi \\ a_1 &= \frac{\mathbf{C}_{32} - \mathbf{C}_{23}}{2\sin(\phi)} = 0 \\ a_2 &= \frac{\mathbf{C}_{13} - \mathbf{C}_{31}}{2\sin(\phi)} = 0 \\ a_3 &= \frac{\mathbf{C}_{21} - \mathbf{C}_{12}}{2\sin(\phi)} = 1 \end{aligned}$$

5.5 Outlier Rejection

Outlier rejection addresses the fact that every LIDAR sweep may contain unreliable points. These are typically removed altogether in the preprocessing step but in the event they are not, a number of additional methods can be used.

For a single registration step, the most common method is to evaluate the distance of point-to-point correspondences and to establish a threshold for this distance, typically set near the expected translation. Corresponding points with point-to-point distances greater

than this threshold are rejected. This method can work reliably if there is no significant change in measurement uncertainty with respect to range but if further away points have consistently higher uncertainty, these points will always be rejected, a feature that is not always desired.

Another common method is to use a RANSAC scheme. Pick a random subset of points and, along with their correspondences, compute the relative transformation. Re-project all points in the matching point cloud using the estimated transform and compute all error residuals. Iterate until a desired residual has been attained, at which point outliers have been identified and disregarded during the computation of the transform. However, since ICP is already an iterative scheme, adding an additional iterative algorithm within it may not be computationally ideal.

For multiple ICP registration steps, it is more difficult to reject outliers based on point-to-point distances as these are generally small for every incremental transformation. Instead, one method involves accepting that a certain percentage of points are outliers and always picking the top percentile of points with the least point-to-point residual. In reality, the percent of outliers is never constant and with no reliable way of dynamically varying this percentage, important measurements can be left out if the threshold is set too low or outliers can be left in if the threshold is set too high.

As mentioned earlier, outlier rejection is performed naturally when using the Weighted OLATE method. During the pre-processing step, unreliable points are attributed a greater uncertainty and will hence play less of a role in the registration process. However, all the aforementioned outlier rejection methods can still be applied for robustness.

5.6 LIDAR Odometry and Mapping

Odometry refers to the estimation of pose over time and is the next step of accomplishing local SLAM. A naive approach to LIDAR Odometry is to string pose-to-pose estimates obtained from the ICP algorithm described in the previous section together. However, since each pose-to-pose estimate is inaccurate, errors are accumulated over time and drift, much like drift encountered in IMU integration. One way to mitigate drift is to build a local map of points and scan match every new sweep with that map, initializing the pose with pose-to-pose scan matching estimates. The point cloud representing the local map needs to be filtered regularly such that its size grows linearly with motion and not with time. This can be achieved by using a Voxel Filter. The general procedure of LIDAR Odometry is shown in Fig. 5.7. To begin, always use the first scan to initialize the local map. This can be done by setting the first body frame $\mathcal{F}_{v_1}$ as the local frame $\mathcal{F}_i$ and setting the pose of the robot at the

first timestep to zero. Next, perform pose-to-pose scan matching between the second point cloud and the previous point cloud using the ICP algorithm to obtain $\tilde{\mathbf{T}}_{v_1\tilde{v}_2}$, a rough estimate of the pose of the robot at the second timestep. Because the map currently contains only the first point cloud, $\tilde{\mathbf{T}}_{v_1\tilde{v}_2} = \mathbf{T}_{v_1v_2}$. Project the second point cloud into the local map frame and use a Voxel filter to downsample the map point cloud. With an incoming third scan, perform pose-to-pose scan matching once again, this time between the third point cloud and the previous point cloud using ICP to obtain $\tilde{\mathbf{T}}_{v_2\tilde{v}_3}$. At this point, use this pose estimate to initialize the pose-to-map scan matching by projecting the third point cloud into the local map frame and use ICP to compute $\mathbf{T}_{v_2v_3}$. Finally, reproject the third point cloud into the local map frame and merge it with the map point cloud via voxel filtering.

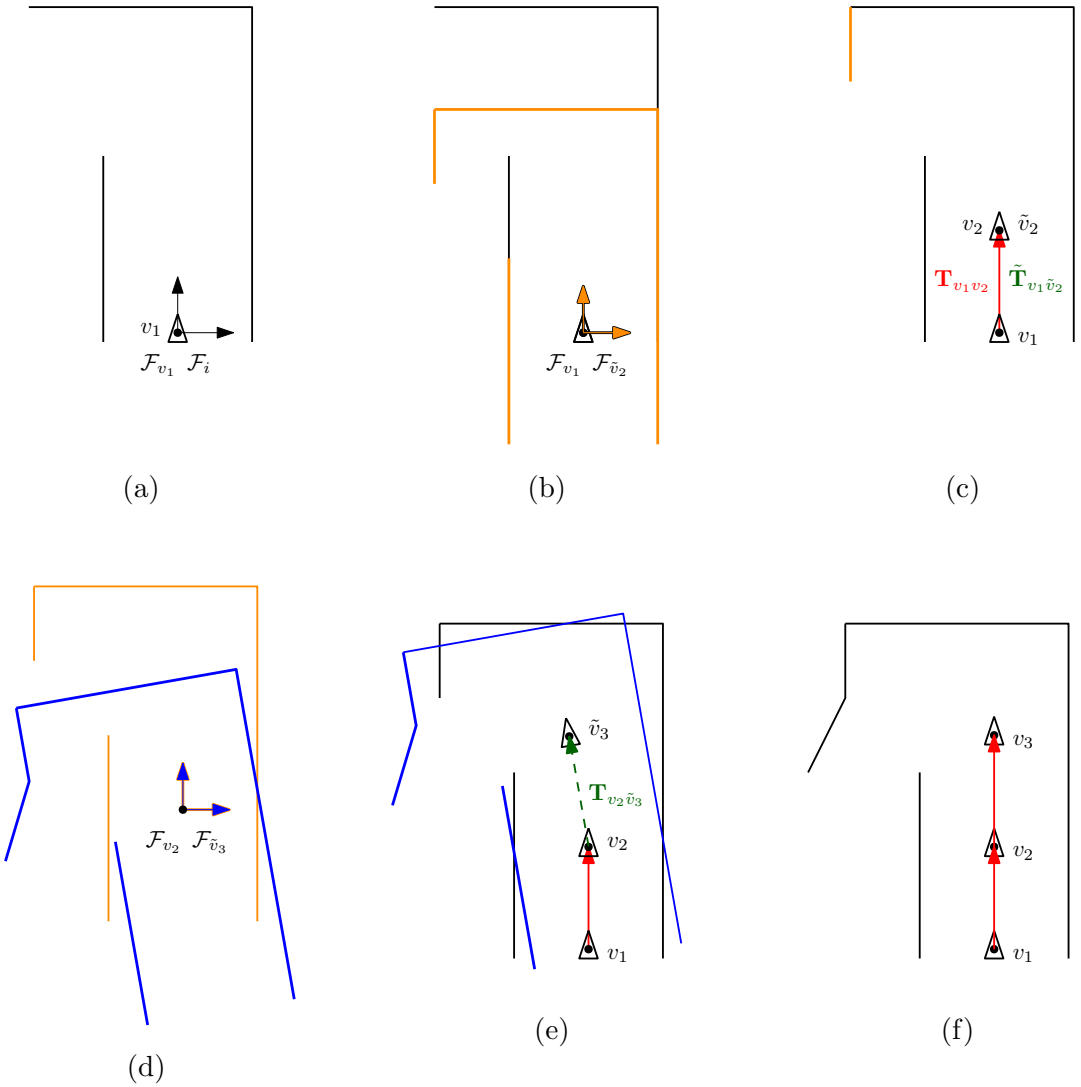


Figure 5.7: Lidar Odometry procedure. a) Initialize first scan at local map origin. b) Compute $\tilde{\mathbf{T}}_{v_1\tilde{v}_2}$ using pose-to-pose scan matching via ICP. c) Project the second scan onto the local map using $\tilde{\mathbf{T}}_{v_1\tilde{v}_2}$ and downsample and merge with map point cloud. d) Compute $\tilde{\mathbf{T}}_{v_2\tilde{v}_3}$ via pose-to-map scan matching. e) Project third scan onto map using $\tilde{\mathbf{T}}_{v_2\tilde{v}_3}$, perform pose-to-map scan matching to obtain $\mathbf{T}_{v_2\tilde{v}_3}$. f) reproject third scan onto map using $\mathbf{T}_{v_2\tilde{v}_3}$, downsample and merge with map point cloud.

Chapter 6

Results and Discussions

The registration algorithms derived in Chapter 5 are tested in simulation, and then in an experimental setting. Testing in both simulation and in an experimental setting is done for a number of reasons. Though the underlying principle of WOLATE, that is weighted least squares, should theoretically provide a more accurate estimate than its unweighed counterpart, the magnitude of these gains when applied to realistic measurements corrupted by realistic levels of noise can only be assessed using experimental data. There is also never any guarantee that LIDAR points correspond to close-by points in a subsequent scan, even when features are extracted. In the best of cases, one can be fairly certain that a feature point, such as a surface point, most likely corresponds to a nearby surface point, and thus belong to the same plane. Furthermore, the addition of weights introduces a number of parameters that need to be tuned to optimize performance. Testing in simulation provides a means to tune parameters and also provides groundtruth data that are more difficult to obtain in real-life. Experimental testing is also necessary to provide more realistic measurement noise and outliers that are difficult to model in a simulation setting.

6.1 Simulation Results

In this section, WOLATE is rigorously tested such that the degree of improvement over Horn's registration method and the linearized point-to-plane registration method can be better understood. First, point clouds having perfect point-to-point correspondences will be assumed. This case can arise in visual odometry applications using camera features assuming correct feature correspondence and no outliers. Next, WOLATE's performance within an ICP framework will be evaluated in the case where point-to-point correspondences are not assumed. This is the more realistic case encountered by LIDAR-based scan-matching methods that extract features as described in Chapter 4.

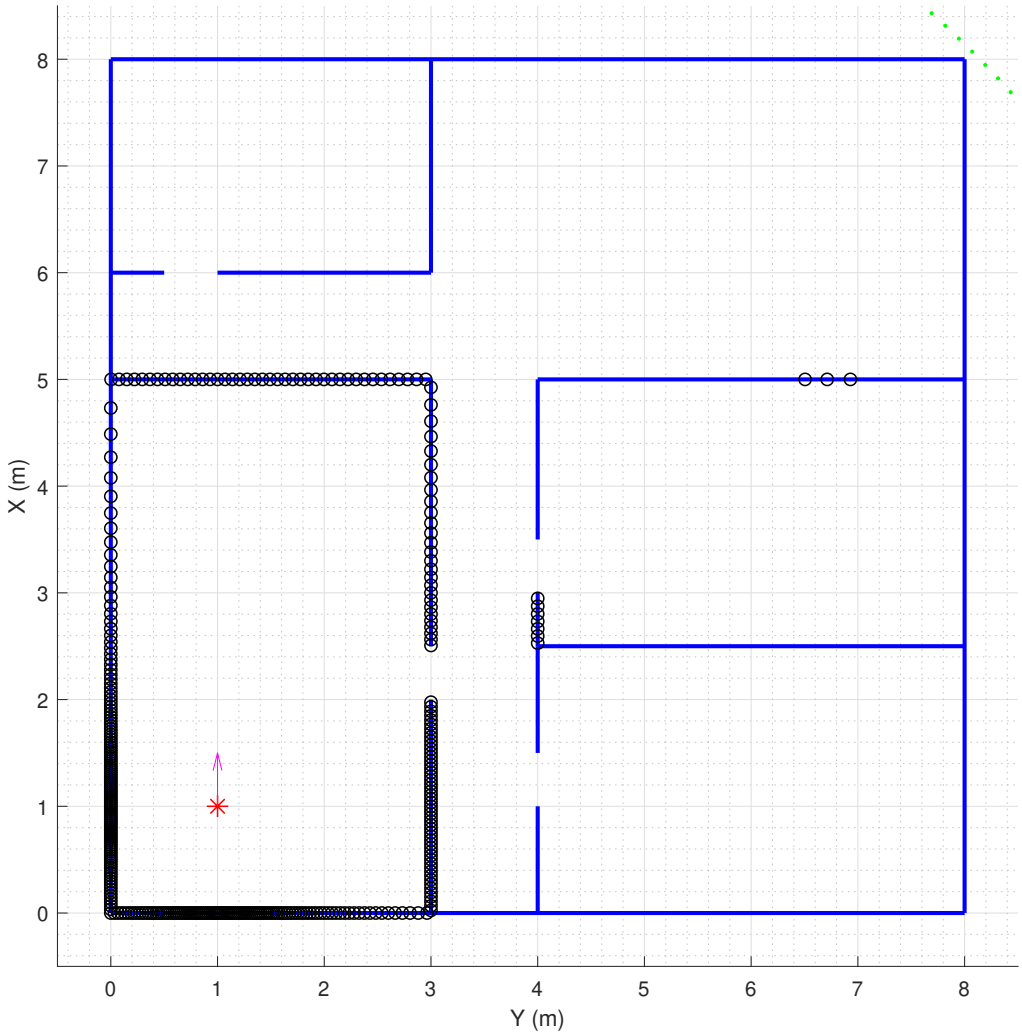
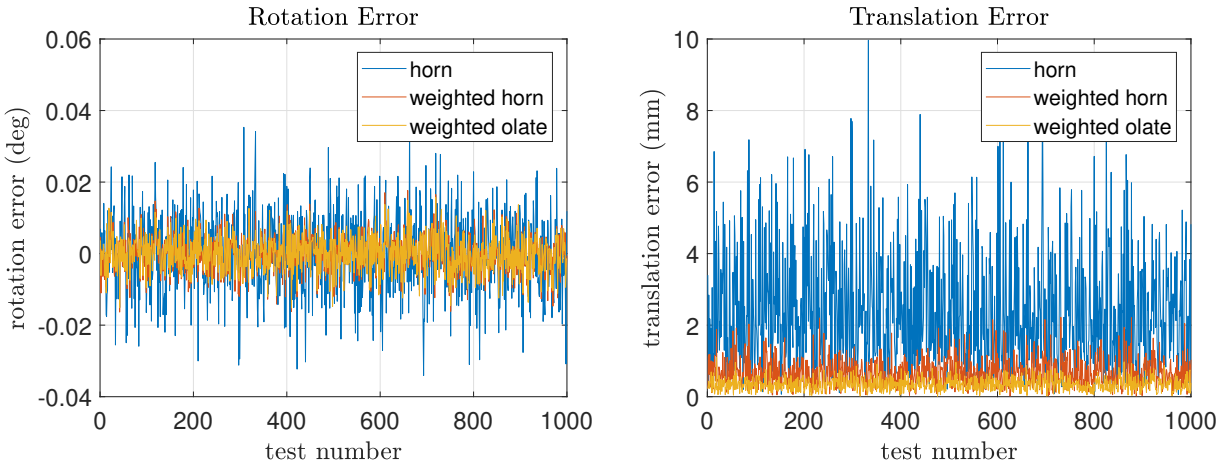


Figure 6.1: Simulated LIDAR measurement of a room.

6.1.1 2D Registration with Perfect Correspondence

The simulation and test setup is as follows. A virtual map resembling the floor of a building with many rooms such as the one in Fig. 6.1 is created. A LIDAR scan is taken and is subsequently transformed to mimic motion, assuming that each point is somehow tracked, thus yielding perfect point-to-point correspondence. This is analogous to a case where a camera tracks feature points as it moves. Range and incidence angle of each LIDAR beam is computed and Gaussian noise with zero mean and covariance as modelled in Chapter 3 is injected into each point cloud. Fig. 6.2 compares the performance of point-to-point Horn's



(a) Rotation error.

(b) Translation error.

Figure 6.2: Point-to-point registration: Rotation and Translation error for 1000 runs. $\theta_{true} = 10°$ $t_{true} = 1$m.

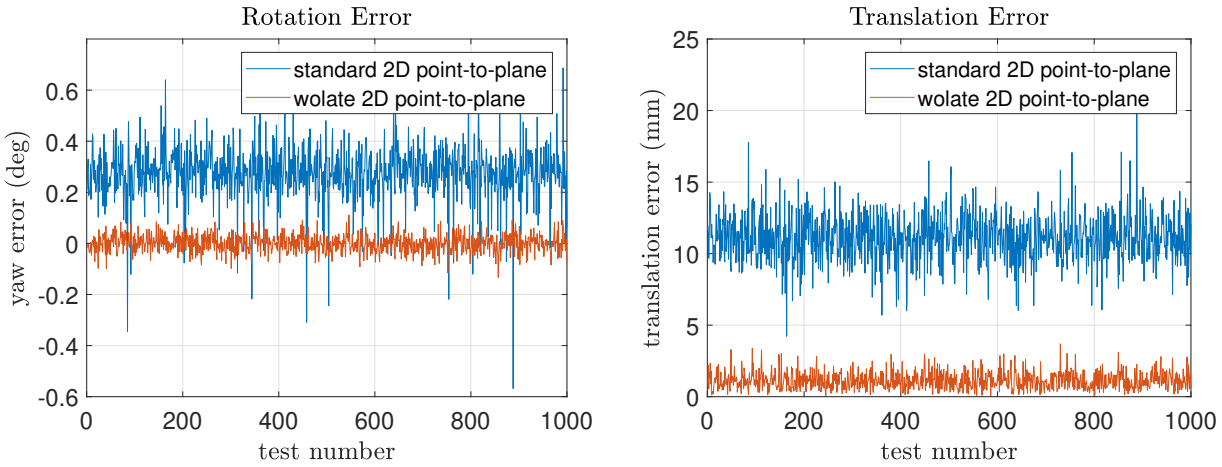
Method, Horn's Method with scalar weights, and WOLATE on this point-cloud pair. It can be seen that although rotation errors for WOLATE and weighted Horn's are similar, WOLATE estimates the translation more accurately, thus making it a better estimator for overall pose.

For the point-to-plane case, a similar trend is observed though the benefits are more pronounced, as shown in Fig. 6.3. This is in large due to the fact that the standard linearized point-to-plane registration method makes the small angle approximation that point-to-plane WOLATE does not make. Recall that the 2D point-to-line problem is equivalent to the point-to-plane problem so its results are omitted from this analysis.

These results suggest that if a good initial point correspondence is provided a priori, then point-to-plane WOLATE-based ICP will be more accurate than standard point-to-plane ICP. To test this theory, a new simulation scenario is constructed where the matching point cloud is no longer obtained by transforming the reference point cloud but is itself, a unique measurement taken at a different pose, much like what one would expect with a real LIDAR.

6.1.2 2D Registration with Realistic Conditions

Before commencing the more realistic simulations, a way is needed to simulate initialization noise for the ICP algorithm. Suppose a LIDAR is mounted on a UAV with a maximum angular velocity of 100°/sec and a sampling speed of 600RPM or 10sweeps/sec. A single sweep has a period of 0.1sec, which is enough time for the UAV to rotate no more than 10°. Similarly, assuming a maximum operational speed of 5m/s, in the same time frame,



(a) Rotation error.

(b) Translation error.

Figure 6.3: Point-to-plane registration: Rotation and Translation error for 1000 runs. $\theta_{true} = 10^{\circ}$ $t_{true} = 1$m.

Table 6.1: IMU measurement noise

	acceleration $(\frac{m}{s^2})$	gyroscrope (rad/sec)	magnetometer (Gauss)
bias [x y z]	[0.2 -0.38 0.14]	[0.02 -0.01 0.05]	[0.015 0.25 -0.12]
covariance	0.75	5e-4	1e-6

the UAV will have travelled no more than 0.5m. Suppose now that the UAV is operating indoors and thus does not have GPS measurements to correct IMU bias and error. The IMU itself has a sampling rate of 1000 Hz for the accelerometer and gyro and 50 Hz for the magnetometer. The expected translation and yaw error are simulated by keeping the UAV fixed and observing the sweep-to-sweep drift in yaw and translation when the measurements are corrupted with worse case bias and noise typically observed in practice. The worse case bias and noise values in Table 6.1 were provided by ARA Robotique.

Figure 6.4 shows that the accrued drift in yaw is mostly contained within its 3σ-bounds of ± 1.2678deg while Fig. 6.5 shows that the accrued drift in x and y position is mostly contained within its 3σ-bounds of $\pm 0.78m$ and $\pm 0.84m$ respectively. In other words, the pose prediction supplied by this simulated IMU should not deviate beyond these 3σ-bound values.

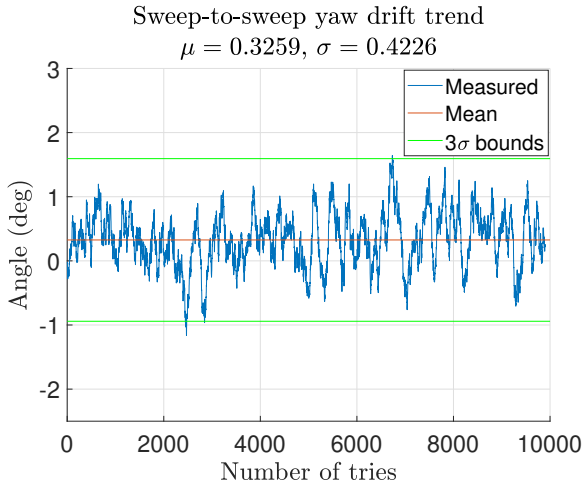
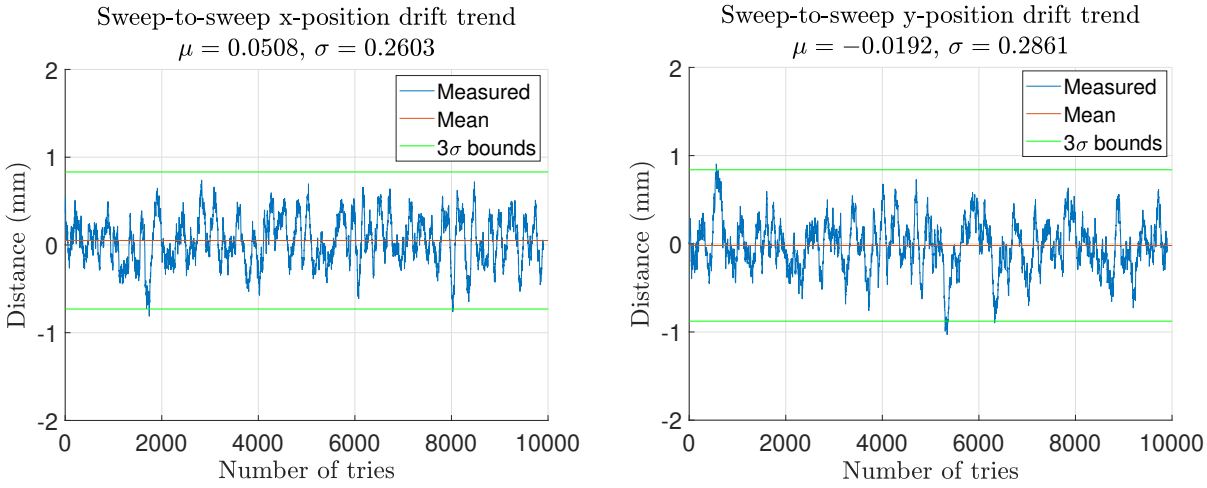


Figure 6.4: Yaw drift accrued over $0.1sec$, the sweep-to-sweep time for this particular LIDAR.



(a) Drift in x-position.

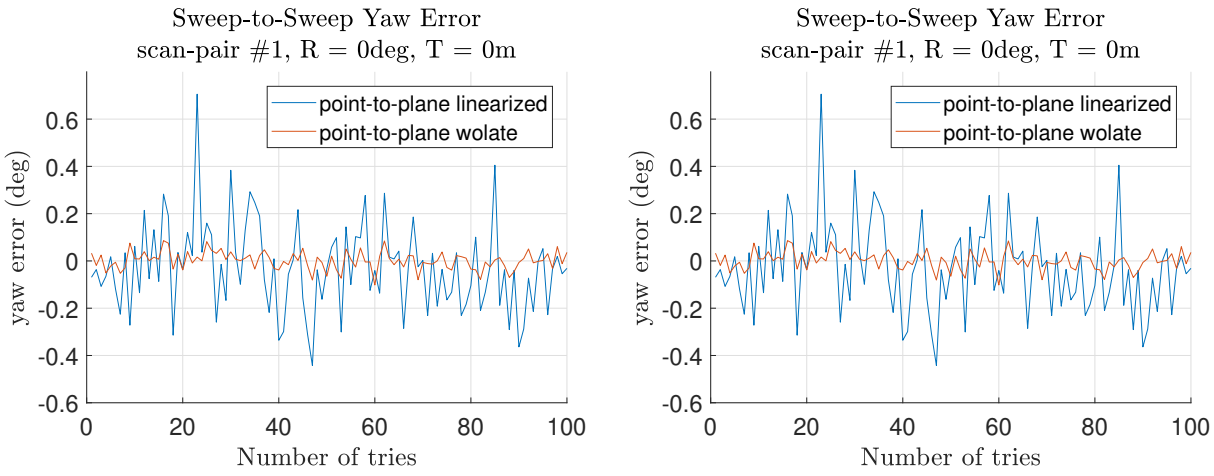
(b) Drift in y-position.

Figure 6.5: Accrued position drift accrued over $0.1sec$, the sweep-to-sweep time for this particular LIDAR.

.

Next, the error in scan matching is evaluated when the sweep to be matched is initialized with the mean and standard deviations for translation and yaw drift between successive sweeps obtained in Fig. 6.4, 6.5a, and 6.5b. Specifically, 1000 sweep-to-sweep LIDAR scan pairs are generated with translation and rotation randomly selected within the aforementioned worst cases, namely, 10° yaw and 0.5m translation. The results of Fig. 6.6 indicate that point-to-plane WOLATE is generally more accurate in translation and yaw estimates compared to its linearized counterpart and also has less error variation. Further simulation

with different sweep-to-sweep pairs validate these results and are presented in Fig. 6.7.



(a) Sweep-to-sweep yaw error.

(b) Sweep-to-sweep translation error.

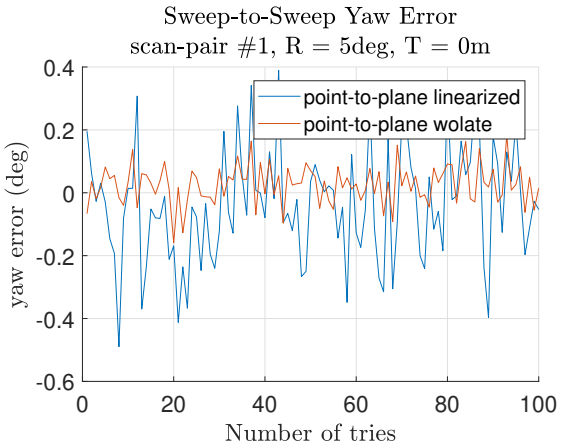
Figure 6.6: Scan matching error using ICP between two scan pairs, where points do not necessarily correspond with each other and where each try uses a randomly generated pose prediction that falls within worse case IMU estimates of 10° yaw and 0.5m translation.

Scan-to-scan registration alone is not enough to construct a cohesive trajectory estimate. In most cases, the transformation obtained through scan-to-scan registration is only used to initialize scan-to-map registration. The next test focuses on LIDAR-based odometry, where scan-to-map registration is performed to obtain pose estimates over time.

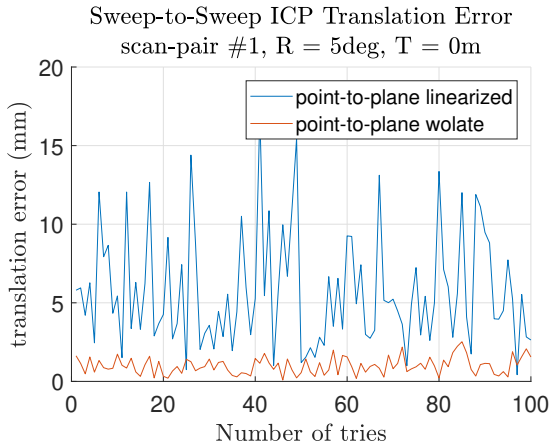
6.1.3 2D LIDAR Odometry

As previously introduced, LIDAR odometry involves two point cloud registration steps. Scan-to-scan matching is performed to obtain a rough estimate of the next pose while scan-to-map matching uses the rough pose estimate as an initialization to refine the pose estimate with respect to a locally built map. Since ICP is used twice in the LIDAR odometry method, there are two opportunities where the use of WOLATE-based point-to-plane ICP, which was shown to be more accurate than its linearized counterpart, can make a difference in terms of accuracy of the overall odometry solution. In light of this, the point-to-plane WOLATE algorithm of Section 5.3.4 is directly implemented within the odometry method described in Section 5.6 and applied to a series of simulated maps and trajectories. Some popular benchmark maps such as the Intel Lab dataset and the ACES3 Austin dataset were recreated in Matlab. Each test is repeated ten times, with different randomly generated noisy pose predictions that fall within worse case IMU estimates of 10° yaw and 0.5m translation.

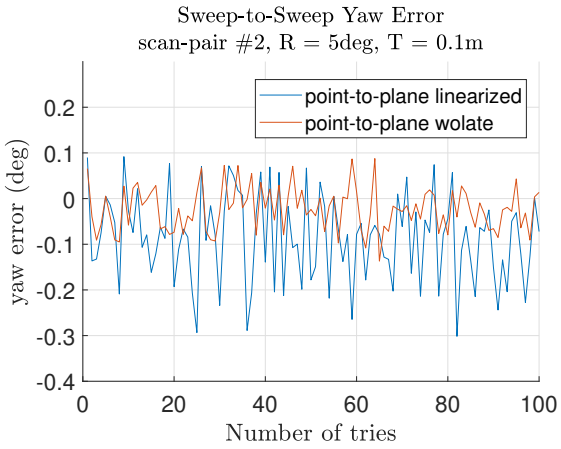
To gauge performance, for the set of results presented in Table 6.2, both rotation and translation errors with respect to the groundtruth have been extracted from the average of



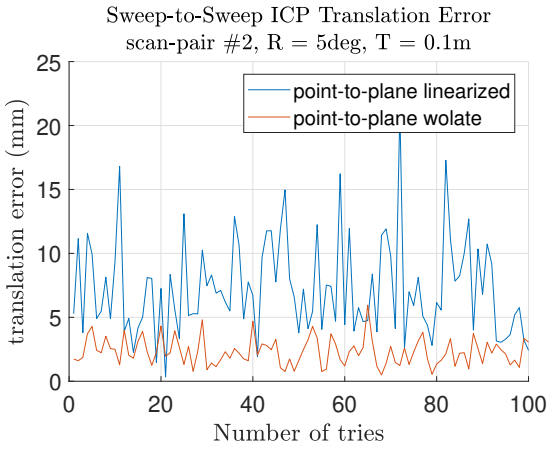
(a) Sweep-to-sweep yaw error.



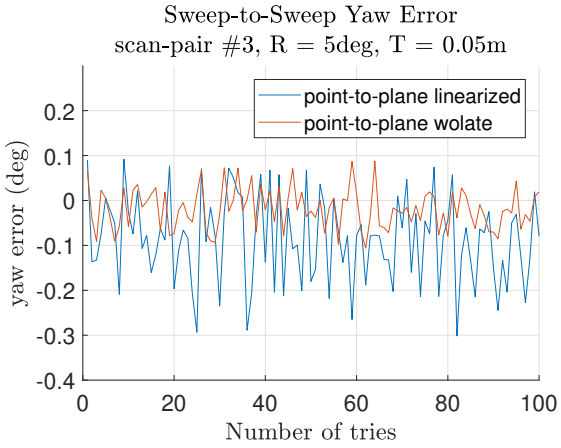
(b) Sweep-to-sweep translation error.



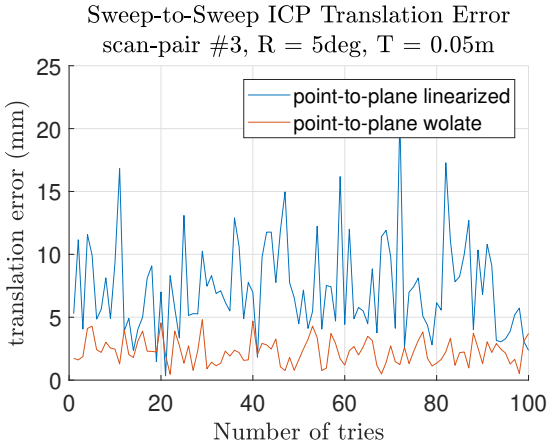
(c) Sweep-to-sweep yaw error.



(d) Sweep-to-sweep translation error.



(e) Sweep-to-sweep yaw error.



(f) Sweep-to-sweep translation error.

Figure 6.7: Comparison between linearized and WOLATE-based point-to-plane registration for three other scan pairs.

Table 6.2: Performance comparison of different registration methods in LIDAR odometry.

	Linearized pt-to-plane translation error averaged over 10 runs (m)	**WOLATE pt-to-plane translation error averaged over 10 runs (m)**	**Linearized feature-based pt-to-plane translation error averaged over 10 runs (m)**	**Total-reg translation error averaged over 10 runs (m)**	**Linearized pt-to-plane yaw error averaged over 10 runs (rad)**	**WOLATE pt-to-plane yaw error averaged over 10 runs (rad)**	**Linearized feature-based yaw error averaged over 10 runs (rad)**	**Total-reg yaw error averaged over 10 runs (rad)**
Corridor dataset	0.3547	0.2306	0.2067	**0.1454**	0.0176	0.0167	0.0116	**0.0108**
Intel lab dataset	**0.1502**	0.1985	0.1746	0.2200	**0.0069**	0.0139	0.0143	0.0092
ACES3 Austin	0.3858	**0.3499**	0.4559	0.3510	0.0156	**0.0119**	0.0219	0.0189

the pose estimate from ten separate tests, and accumulated over the entire trajectory via

$$\mathbf{e} = \sqrt{\frac{1}{M}\sum_{j=1}^{M} \mathbf{e}_j^{\mathsf{T}} \mathbf{e}_j}$$

$$\mathbf{e}_j = \left\| \hat{\mathbf{r}}_{i,j}^{vi} - \mathbf{r}_{i,j}^{vi} \right\|^2$$
$$\mathbf{e}_j = \left\| \ln\left(\hat{\mathbf{C}}_{iv,j}^{-1} \mathbf{C}_{iv,j} \right)^{\vee} \right\|^2 .$$

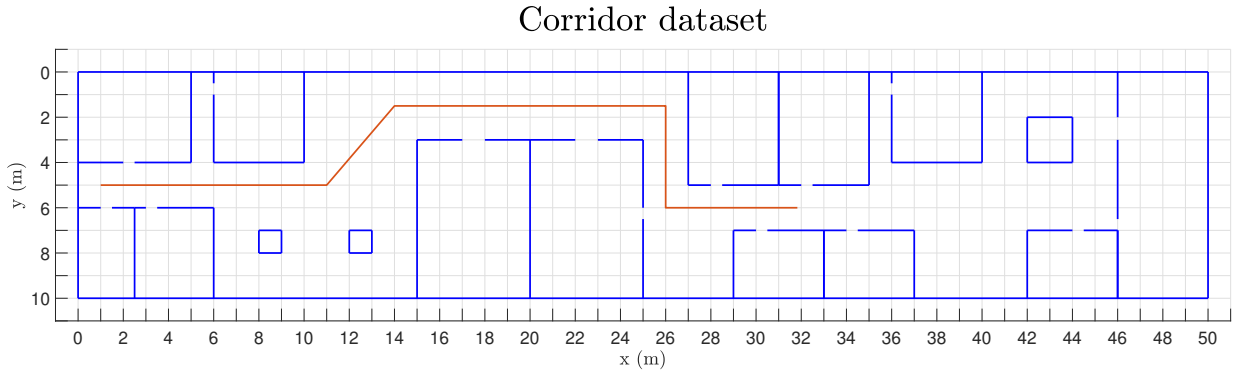
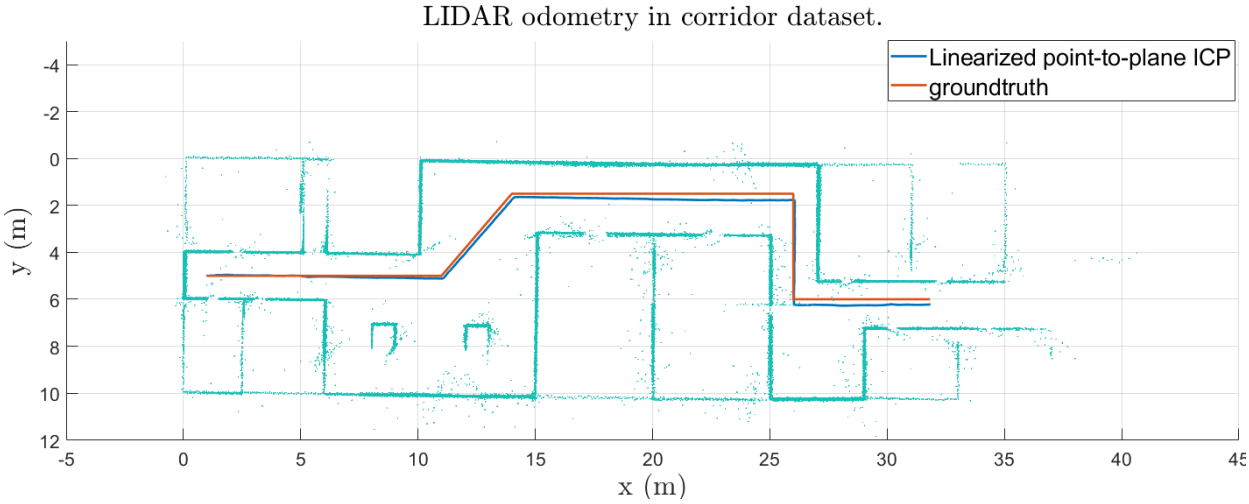
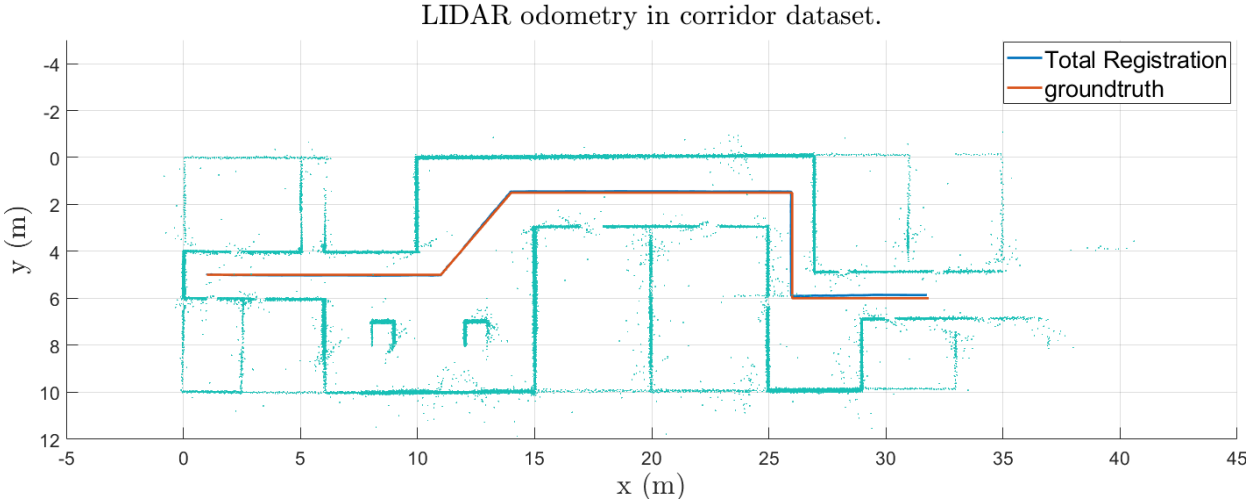


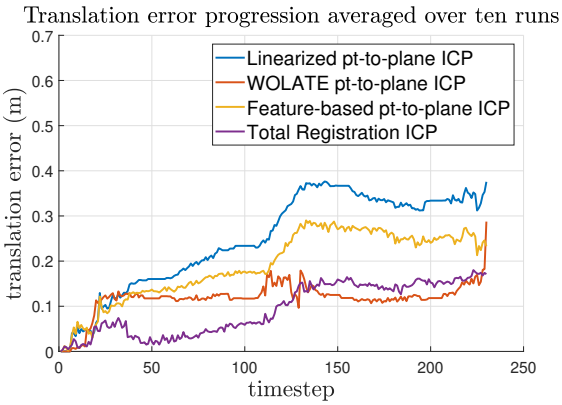
Figure 6.8: A simulated 2D corridor with rooms and square columns as features and the trajectory shown in red.



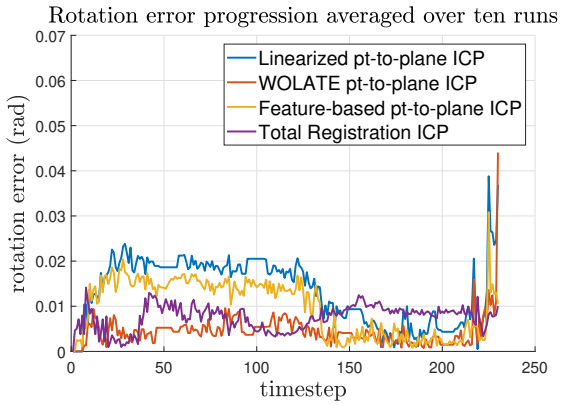
(a) LIDAR odometry using Linearized point-to-plane ICP.



(b) LIDAR odometry using Total Registration.



(c) Translation error along trajectory.



(d) Rotation error along trajectory.

Figure 6.9: Rotation and Translation error comparison between different registration methods.

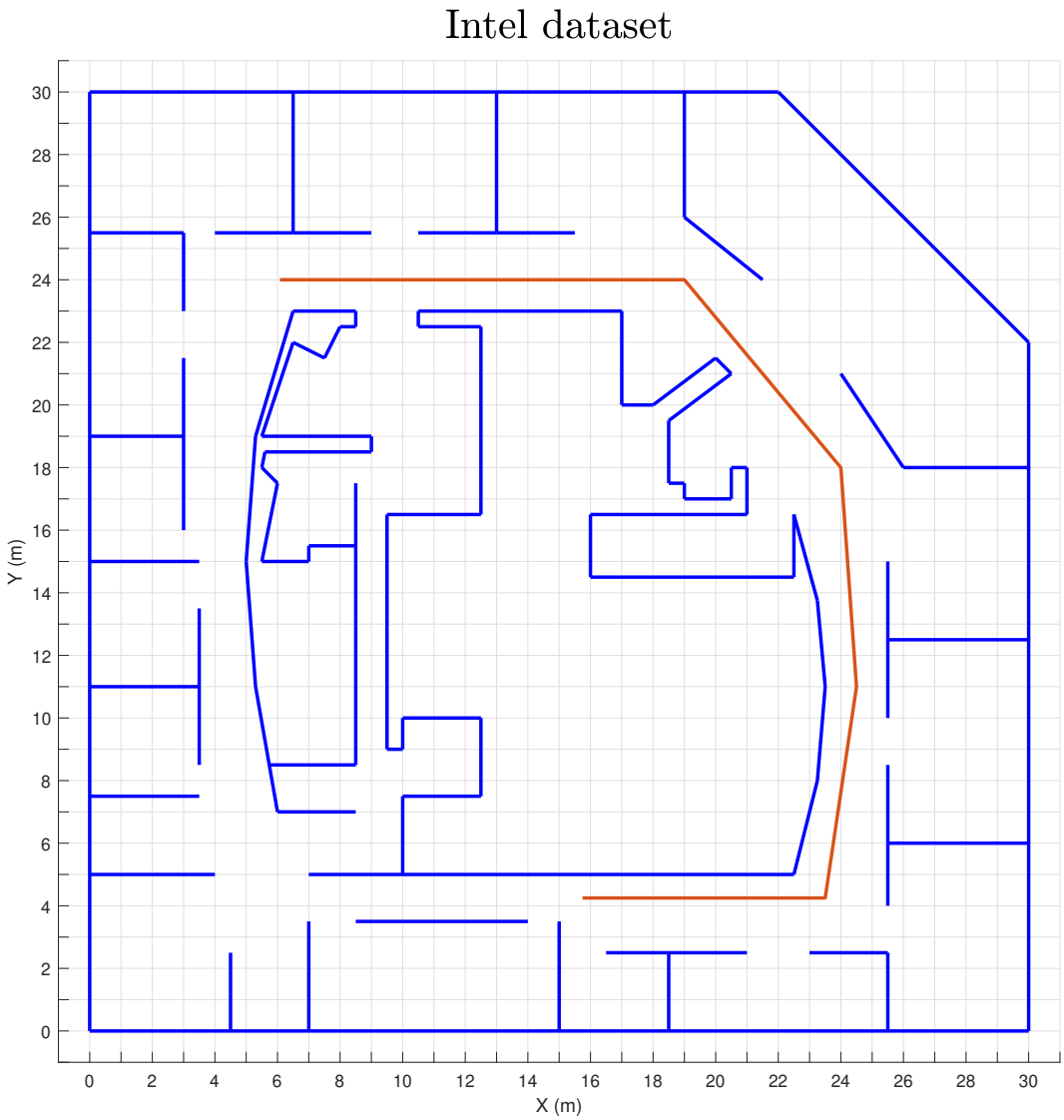
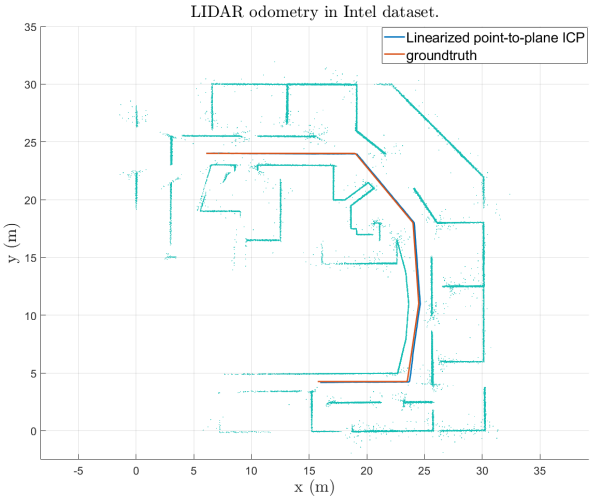
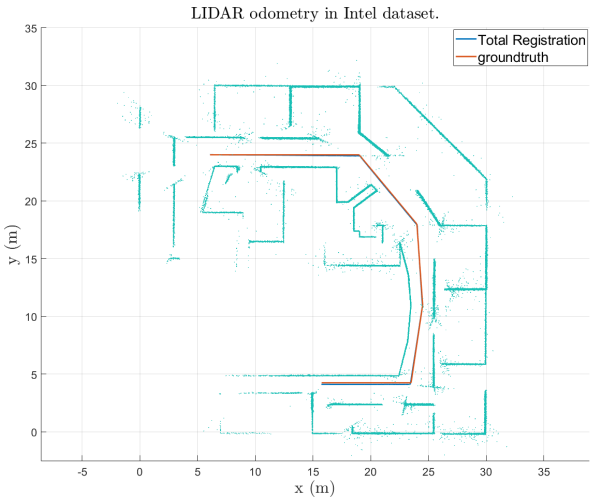


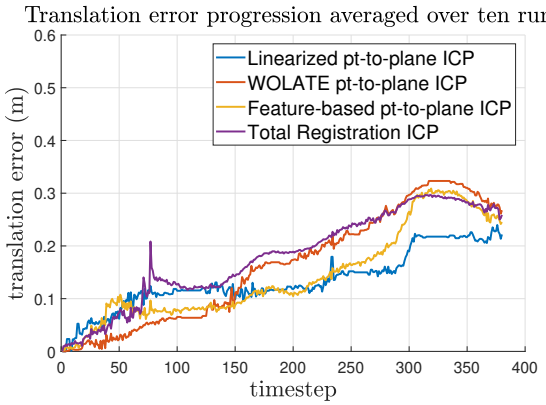
Figure 6.10: A reconstruction of the Intel dataset with trajectory shown in red.



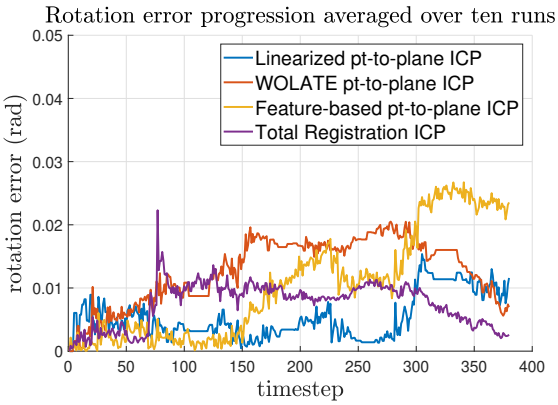
(a) LIDAR odometry using linearized point-to-plane ICP.



(b) LIDAR odometry using WOLATE point-to-plane ICP.



(c) Translation error along trajectory.



(d) Rotation error along trajectory.

Figure 6.11: LIDAR odometry performed on the reconstructed Intel dataset.

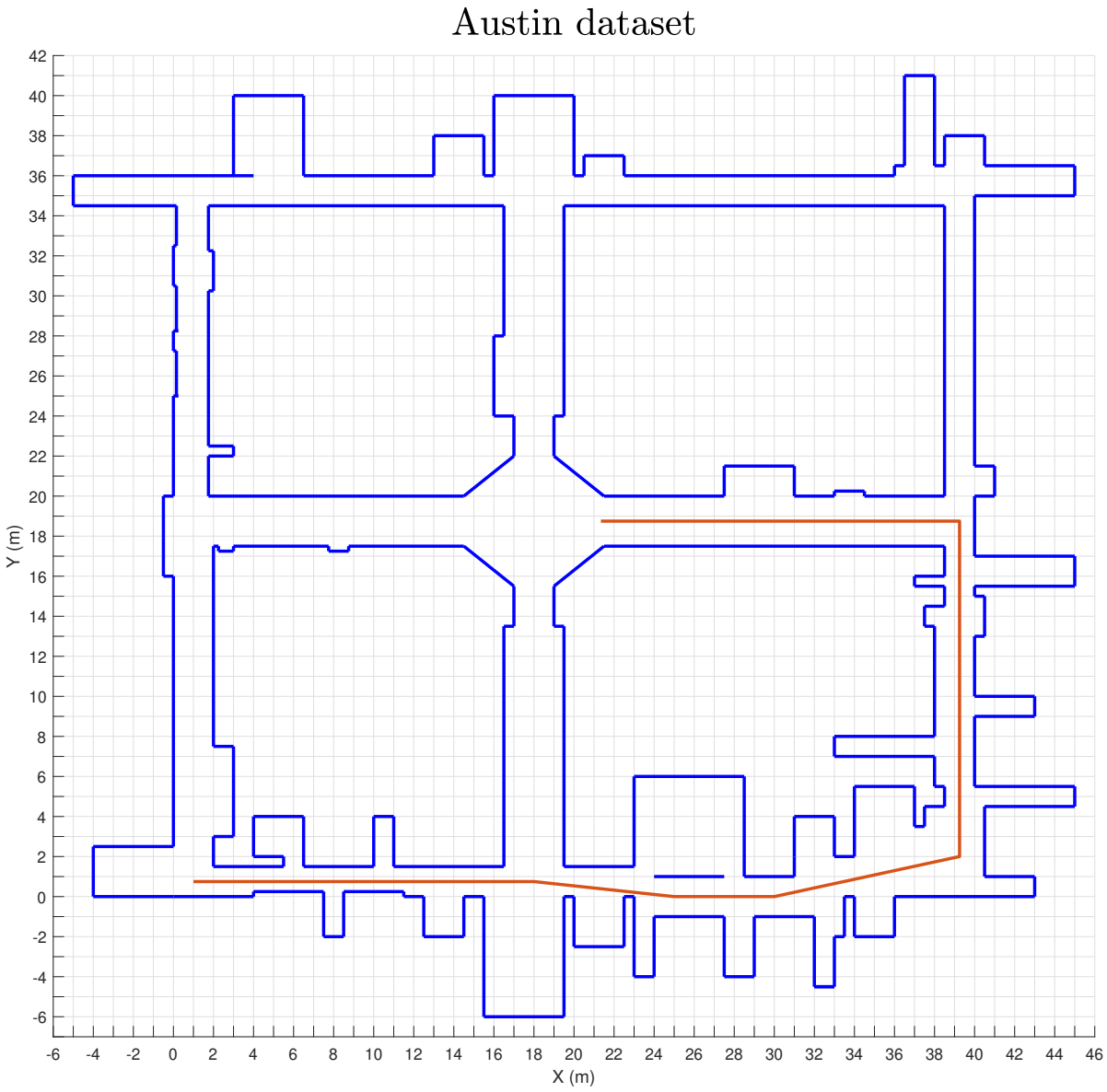
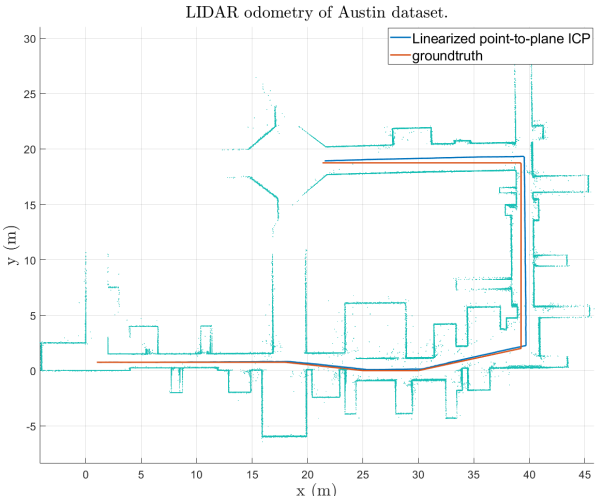
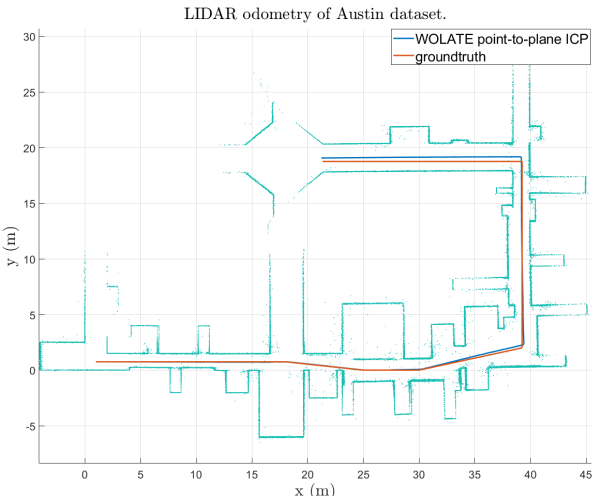


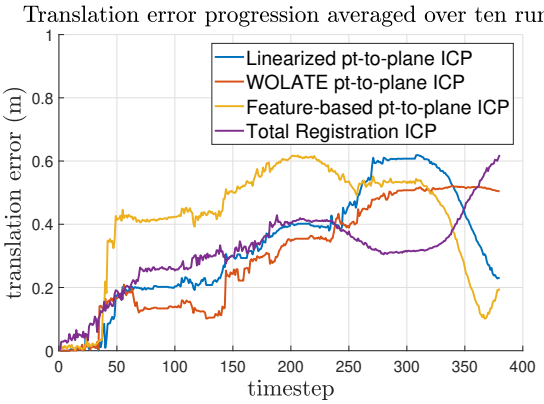
Figure 6.12: A reconstruction of the ACES3 Austin dataset with trajectory shown in red.



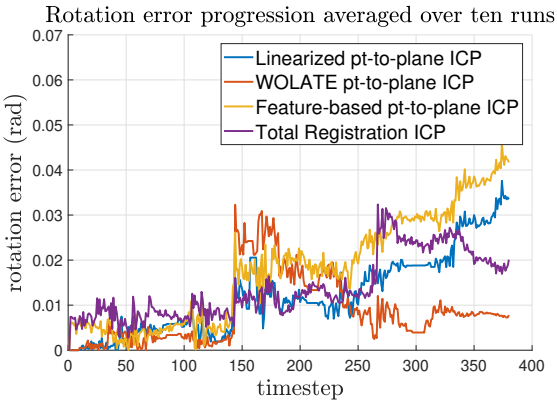
(a) LIDAR odometry using Linearized point-to-plane ICP.



(b) LIDAR odometry using WOLATE point-to-plane ICP



(c) Translation error along trajectory.



(d) Rotation error along trajectory.

Figure 6.13: Rotation and Translation error comparison between different registration methods.

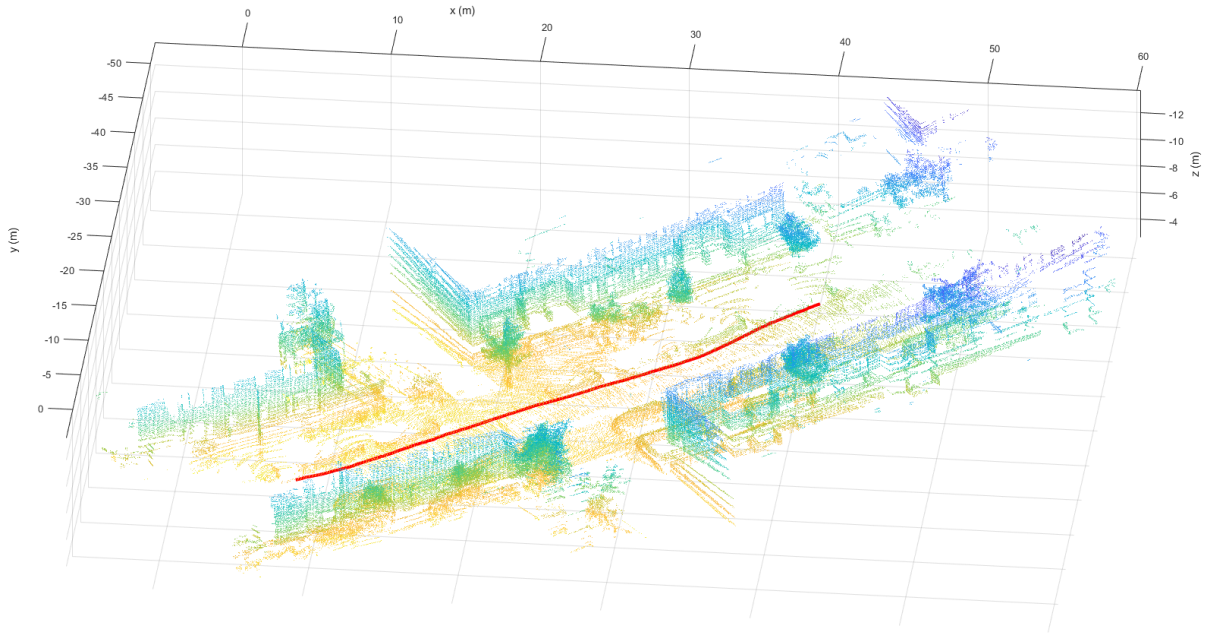
Results indicate that for the Corridor dataset and the Austin dataset, shown in Fig. 6.8 and Fig. 6.12, either point-to-plane WOLATE or Total Registration accumulate less rotation and translation drift than their counterparts while for the Intel dataset show in Fig. 6.10, the Linearized point-to-plane registration method accumulates less rotation and translation error over the prescribed trajectory. This is likely because the Intel dataset is feature-rich compared to the other two datasets while inaccurate corner-point detection and matching of Total Registration hinders its performance. Indeed, depending on the noise that is injected, Total Registration can sometimes outperform the other methods, as seen in Fig. 6.11b. Nevertheless, WOLATE-based registration tends to accrue less rotation error as evidenced by Fig. 6.9d, Fig. 6.11d and Fig. 6.13d, thanks to the fact that WOLATE point-to-plane registration does not make the small angle approximation that Linearized point-to-plane registration makes.

6.2 Experimental Results

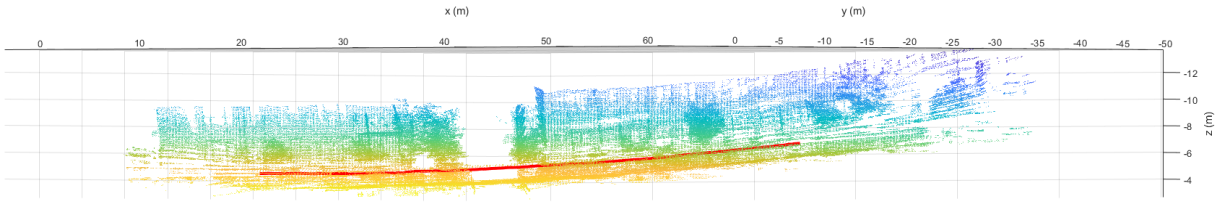
To conclude the experiments, Total Registration is applied on experimental data for validation. A Velodyne Puck LITE LIDAR, an ARA Skymate$^{\text{TM}}$, that houses an INS, and an RTK are mounted on a push-cart and rolled around a city block while collecting data. For this test, the 3D Total Registration algorithm of Algorithm 5.1 is used for scan-to-scan and scan-to-map matching. Notice from Fig.6.14b that without constraining certain states with priors, drift along the z-direction will occur, primarily due to drift in pitch and roll. Because the on-board INS can estimate roll and pitch with minimal drift, thanks to the Kalman Filter correction step, and because the RTK can provide drift-free z-position estimates, it is possible to constrain roll, pitch and z-position using these estimates as priors. Using the constrained formulation described in Section 5.4.1, notice from Fig. 6.15b that the constrained Total Registration-based LIDAR odometry no longer suffers from z-position drift. Also note that the resulting map is a more accurate representation of the intersection than the Google Earth image of Fig. 6.16 that was reconstructed using bundle adjustment of aerial images.

In a separate test, LIDAR data was gathered using an ARA Robotique drone flying along a predetermined GPS-guided path above a quarry plateau. Reconstructing the map using RTK and INS pose estimates produced suboptimal results as seen in Fig. 6.17a due to the vibration-sensitive gyroscope that produced inaccurate attitude estimates. Because the position estimates of the RTK are a lot more reliable, this is a suitable scenario where constrained OLATE can be used to constrain x and y positions. Plotting the point-to-map error where the groundtruth map is built using Bentley [59], a powerful post-processing

software that uses RTK/INS and camera images, reveals that using Constrained OLATE in ICP scan matching produces a more accurate map than using pure RTK/INS estimates Fig. 6.17.

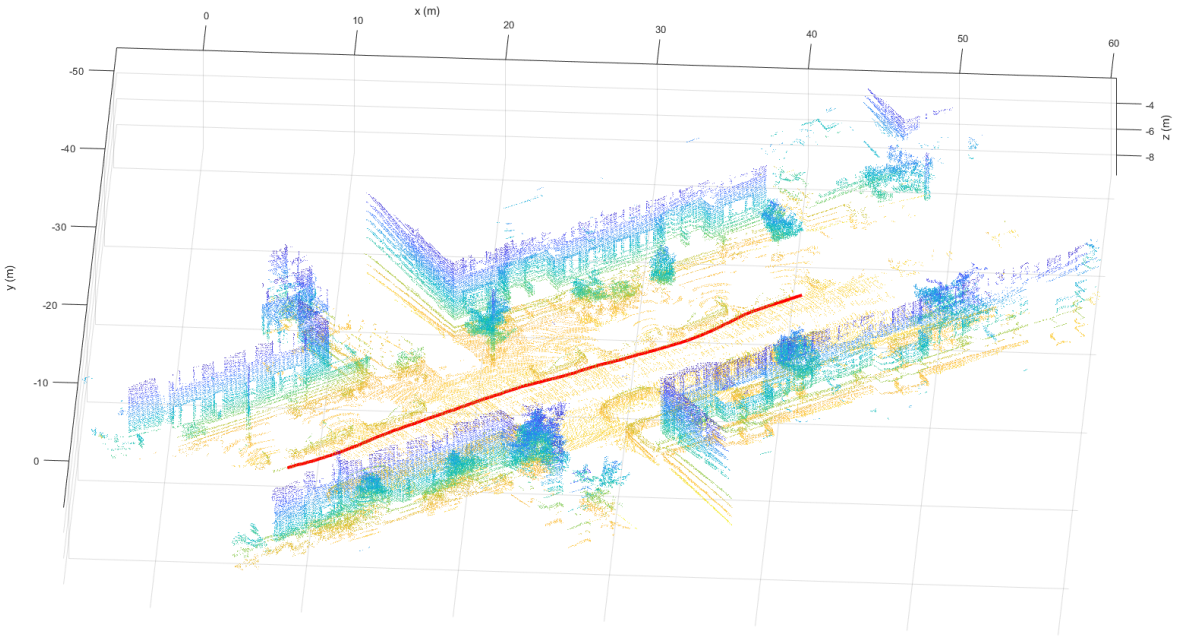


(a) Isometric view

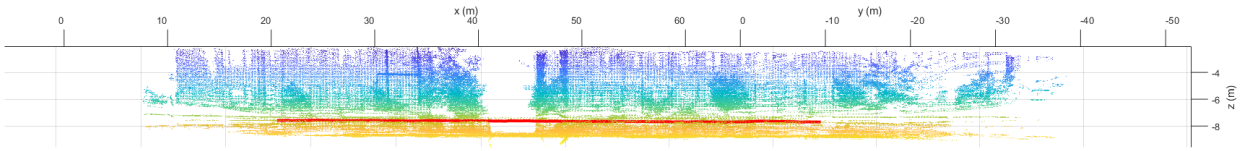


(b) Side view.

Figure 6.14: 3D LIDAR odometry using unconstrained Total Registration with estimated Trajectory in red.

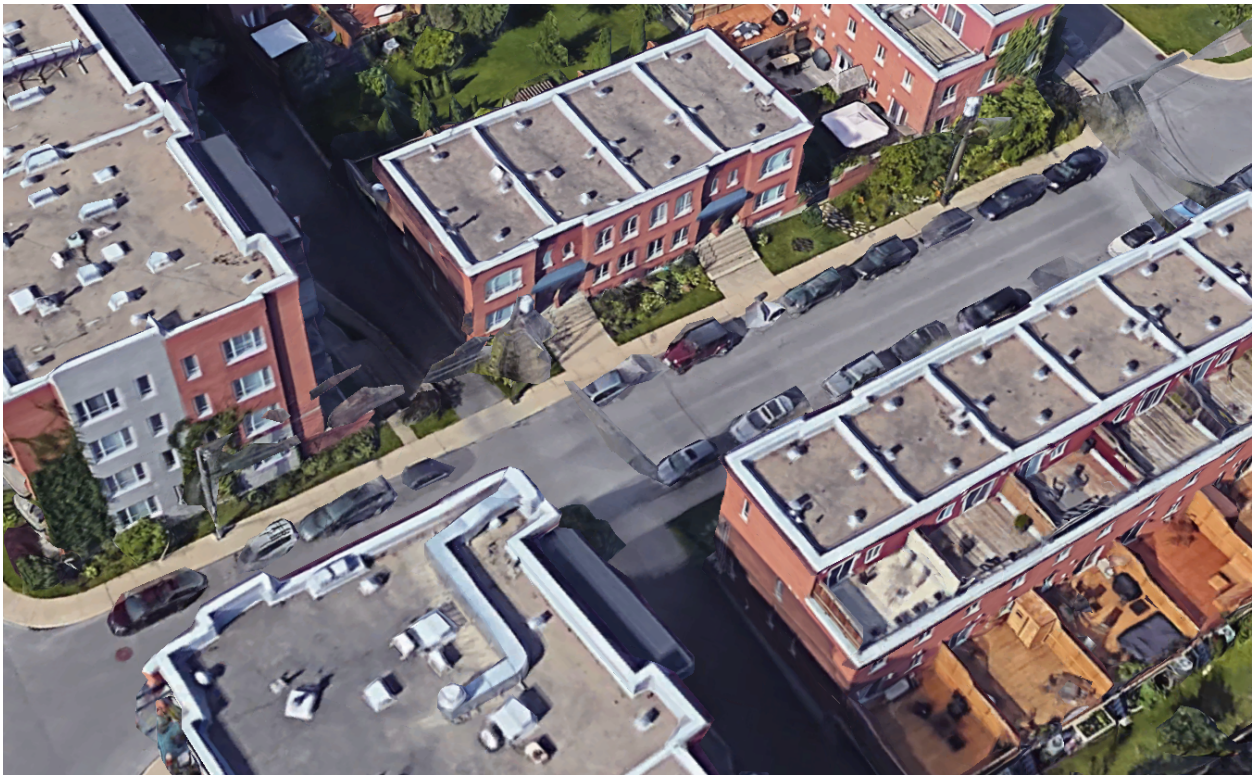


(a) Isometric view.

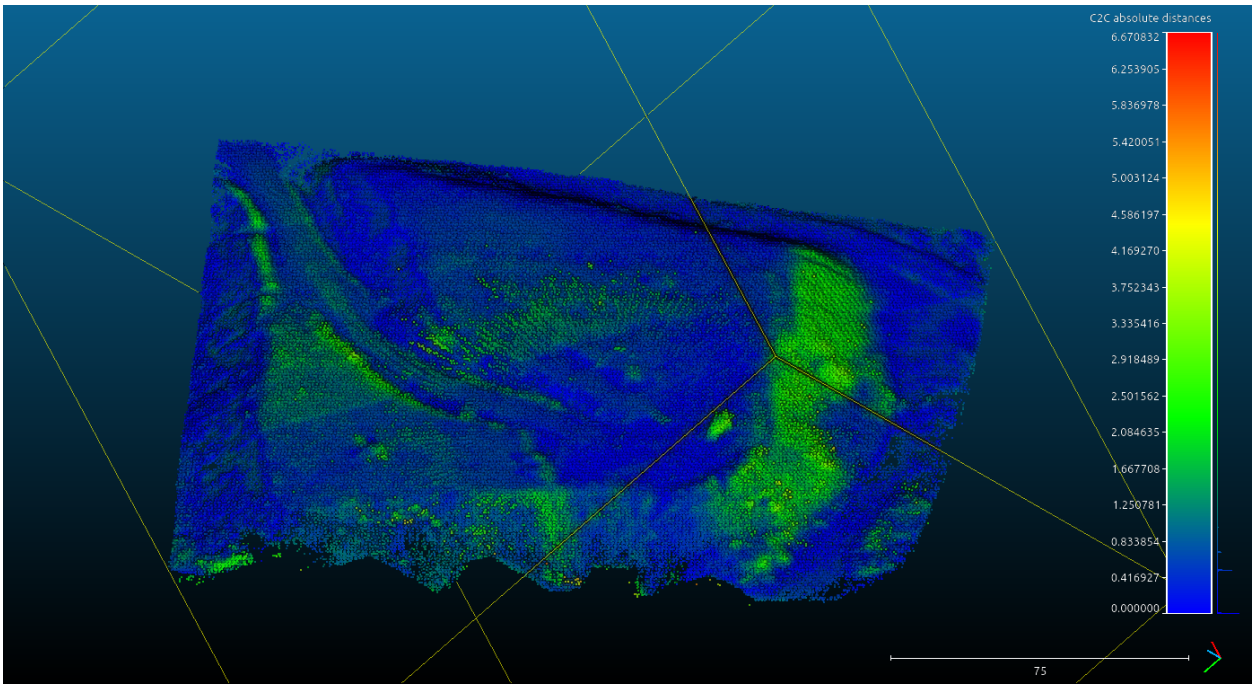


(b) Side view.

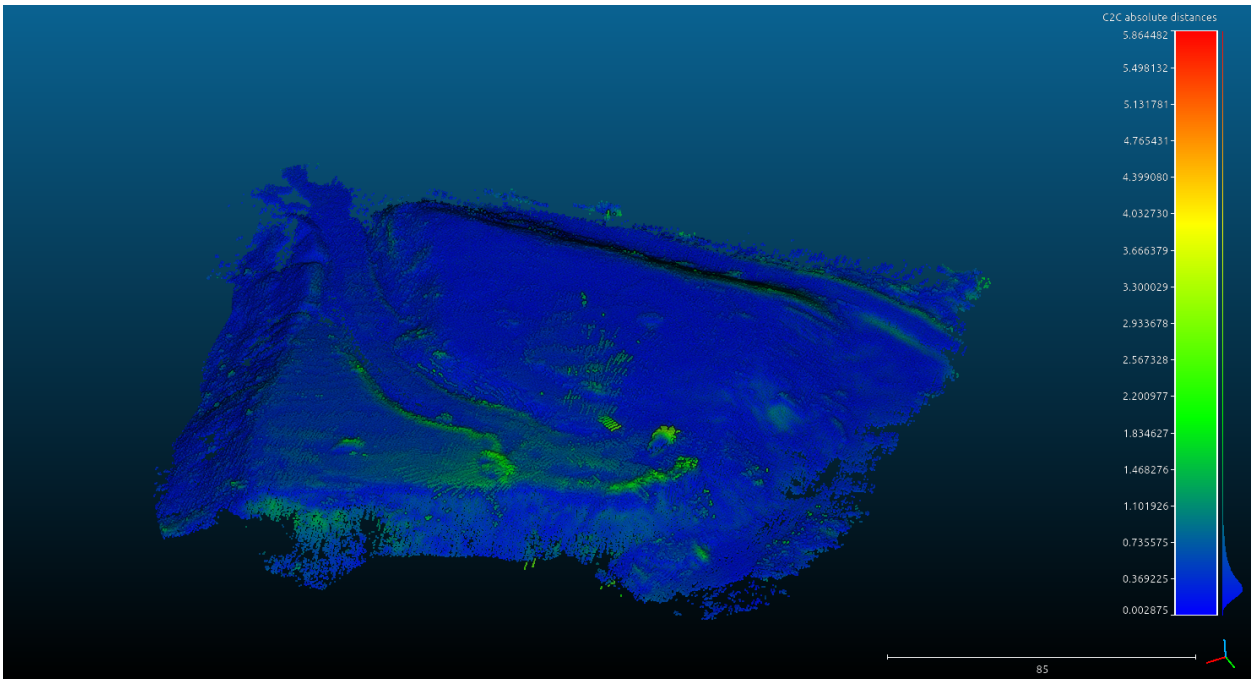
Figure 6.15: 3D LIDAR odometry using constrained Total Registration with estimated Trajectory in red.



Source: **Google Earth**. Version 9.2.64.2. Montreal, Canada. 45°28′22″N, 73°34′57″W. August 14, 2018.
Figure 6.16: Google Earth 3D reconstruction of the same intersection.



(a) Plateau reconstructed using RTK pose estimates.



(b) Plateau reconstructed using Constrained OLATE.

Figure 6.17: Reconstructed map of quarry plateau using RTK and Constrained OLATE

Chapter 7

Closing Remarks and Future Work

7.1 Conclusions

This thesis has contributed to the field of point-cloud registration and LIDAR-based navigation. Thanks to a clever application of the Cayley transform, the traditionally nonlinear point cloud registration problem can be converted to a linear least squares problem. Building on this idea, the novel point-to-point, point-to-plane, and point-to-line WOLATE registration algorithms have been rigorously derived. These registration methods are shown to attain greater registration accuracy compared to Horn's method or linearized point-to-plane registration, mostly because WOLATE takes measurement uncertainty into account and partly because it naturally rejects outliers. To test the effectiveness of WOLATE-based registration, a model of 2D LIDAR uncertainty was formulated based on the RP-LIDAR. This model was used to inject noise into simulated point cloud pairs, which were subsequently registered. Registering point clouds with both perfect and imperfect point-to-point correspondence using WOLATE resulted in more accurate pose estimates. Incorporating WOLATE within an ICP algorithm also produced similar results. Finally, WOLATE-based ICP was applied within a LIDAR odometry simulation and shown to outperform standard point-to-point and point-to-plane ICP, in terms of accuracy, in most situations. Specifically, because WOLATE is linear, the three aforementioned registration methods were elegantly combined into a single registration algorithm, Total Registration, which when used alongside a feature extraction algorithm, ensured better point correspondence with their respective pair. WOLATE-based registration has also been shown to be more versatile than its counterparts, not only because different types of points can be leveraged in different situations, but also because it can be easily constrained when reliable priors are available, as seen from the experimental test. These properties make WOLATE-based ICP and Total Registration excellent alternatives to standard ICP registration methods when individual point uncertainties can be estimated

and when the robot frequently rotates. As mentioned in the introduction, the robotics field is entering a new age characterized by robustness to unpredictable real-world environments. The Total Registration approach is one that attempts to tackle this problem by being adaptive to different feature types and perform registration by incorporating as much information as possible under the bounds described in this thesis.

7.2 Recommendations for Future Work

The proposed LIDAR odometry method is still prone to drift after extended periods of time. In practice, old scans are kept in memory so that when the robot arrives at a previously visited location, loop closure can be detected. A loop closure algorithm can then be used to globally optimize all prior pose estimates. A future study should evaluate whether WOLATE has a place in the loop closure scan matching step of global optimization.

Though Chapter 3 presents a way to model 2D LIDAR uncertainty, modelling of 3D LIDAR should also be studied. As range uncertainty is mostly affected by angle of incidence in the 2D case, it is expected that the same holds for the 3D case, though the angle of incidence is between the LIDAR beam and the normal vector of the local surface of contact.

The corner detection method described in Section 4.6 cannot always accurately find corners, particularly those that are further away. This is because the radial resolution of the LIDAR beams is sometimes not enough to accurately capture the exact location of a corner along a scan ring. Recently, Ji and Cheng [60] proposed fitting line segments and extracting corner points by finding the intercept between these line segments. This method deserves further investigation as it is more robust to scale change and potentially more accurate than the corner extraction method proposed in this thesis.

Due to a lack of experimental data with groundtruth, such as that obtained using motion capture systems, the proposed WOLATE-based algorithms were not rigorously tested in a controlled experimental setting. A study using experimental data with groundtruth can yield valuable information on real-life accuracy improvements, which in turn can dictate whether it is suitable for applications that require a certain level of accuracy.

The presented simulations were coded in Matlab and thus, runtime comparisons between WOLATE-based ICP and standard ICP were not considered. A future study should code the WOLATE-based algorithms derived in this thesis in C++ and perform a runtime analysis. In particular, it is hypothesized that WOLATE requires less ICP iterations to attain the same level of accuracy as standard ICP methods. This is however potentially offset by the fact that WOLATE requires additional iterations to estimate the weight matrix.

The simulated 2D LIDAR odometry results were sensitive to the voxel filter size as well

as corner point registration. This is primarily because the corner detection method used in this study was suboptimal. In a future study, corner detection within a LIDAR scan should be improved, which in turn will aid Total Registration. Furthermore, an adaptive voxel filter should be used when dealing with maps of varying scale.

Appendices

Appendix A

Miscellaneous Derivations

A.0.1 Principle Component Analysis

Suppose 2D measurement data in the form of a point cloud is correlated in x and y. Principle Component Analysis is a method that converts this set of correlated variables into linearly uncorrelated ones, thereby extracting a set of principle axes. The first axis is along the direction that produces the most variance in the data in the squared error sense and subsequent axes are orthogonal to the previous axis. The procedure of PCA is as follows. First, compute the centroid of the data. Next compute the distance to the centroid of each point along with the covariance matrix of these distances, given by

$$\mathbf{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^{\mathsf{T}}$$

where there are N points in the data set and $\bar{\mathbf{p}}$ is the centroid. The resulting covariance matrix will most likely show a correlation in x and y. By performing an Eigen decomposition on $\mathbf{\Sigma}$, two eigenvalue-eigenvector pairs can be found such that

$$\mathbf{\Sigma}\mathbf{v}_j = \lambda_j \mathbf{v}_j$$

for $j \in 1, 2$. The eigenvector of the smallest eigenvalue corresponds to the principle axis along which there is the least variance in the data, which is coincidentally the normal direction of a line observed as discrete point measurements. In practice, PCA is useful in computing local surface normals for point-to-plane registration. It can be also applied as a measure of local surface curvature by analyzing the magnitude of the eigenvector along the normal direction.

A.0.2 Proof of Proposition 5.1

These two additional propositions will be needed to prove proposition 5.1.

Proposition A.1. *If two matrices* $\mathbf{A}$ *and* $\mathbf{B}$ *are simultaneously diagonalizable, then the eigenvalues of* $(\mathbf{A}+\mathbf{B})$ *is the sum of the diagonal matrices containing the eigenvalues of each individual matrix.*

Proof. If $\mathbf{A}$ and $\mathbf{B}$ are simultaneously diagonalizable, there exists a common matrix $\mathbf{P}$ such that

$$\mathbf{A} = \mathbf{P}^{-1}\mathbf{D}^{A}\mathbf{P}, \quad \mathbf{B} = \mathbf{P}^{-1}\mathbf{D}^{B}\mathbf{P}$$

where $\mathbf{D}^{A}$ and $\mathbf{D}^{B}$ are diagonal matrices contain the eigenvalues of matrices $\mathbf{A}$ and $\mathbf{B}$ respectively. Then,

$$\mathbf{A} + \mathbf{B} = \mathbf{P}^{-1}(\mathbf{D}_A + \mathbf{D}_B)\mathbf{P}$$

Proposition A.2. *The eigenvalues of a real skew-symmetric matrix are either zeros or complex numbers.*

Proof. If matrix $\mathbf{A}$ is skew symmetric, then

$$\mathbf{A}^{\mathsf{T}} = -\mathbf{A}.$$

Furthermore, the eigenvalues λ and eigenvectors $\mathbf{x}$ of matrix $\mathbf{A}$ are related via

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}.$$

Multiply the left side by $\bar{\mathbf{x}}^{\mathsf{T}}$, the complex conjugate of $\mathbf{x}$. This leads to

$$\bar{\mathbf{x}}^{\mathsf{T}}\mathbf{A}\mathbf{x} = \lambda\bar{\mathbf{x}}^{\mathsf{T}}\mathbf{x} = \lambda \left\|\mathbf{x}\right\|^2 \tag{A.1}$$

where the left hand side can be simplified to

$$\bar{\mathbf{x}}^{\mathsf{T}}\mathbf{A}\mathbf{x} = (\mathbf{A}\mathbf{x})^{\mathsf{T}}\bar{\mathbf{x}} = \mathbf{x}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}\bar{\mathbf{x}} = -\mathbf{x}^{\mathsf{T}}\mathbf{A}\bar{\mathbf{x}}. \tag{A.2}$$

Taking the complex conjugate of $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, the expression

$$\mathbf{A}\bar{\mathbf{x}} = \bar{\lambda}\bar{\mathbf{x}},$$

can be substituted into (A.1) and (A.2) to obtain

$$\bar{\mathbf{x}}^{\mathsf{T}}\mathbf{A}\mathbf{x} = -\mathbf{x}^{\mathsf{T}}\mathbf{A}\bar{\mathbf{x}} = -\bar{\lambda}\mathbf{x}^{\mathsf{T}}\bar{\mathbf{x}} = -\bar{\lambda}\left\|\mathbf{x}\right\|^2 = \lambda\left\|\mathbf{x}\right\|^2.$$

Since $\mathbf{x}$ is a non-zero eigenvector, it can be concluded that $-\bar{\lambda} = \lambda$ implies that λ is either zero or a complex number.

Using A.1 and A.2, the eigenvalues of $(\mathbf{1}+\mathbf{A})$ are of the form $1+\lambda = 1+ib \neq 0$. Since $(\mathbf{1}+\mathbf{A})$ has no zero eigenvectors, it follows that $(\mathbf{1}+\mathbf{A})$ is nonsingular and therefore invertible.

A.0.3 Wahba's Problem and Point-to-Point Registration via SVD

Wahba's Problem, first formulated by Grace Wahba [47], is an optimization problem where the design variables are elements of SO(3). Specifically, given a set of vectors $\underrightarrow{r}^{p_i w}$ resolved in $\mathcal{F}_a$ and $\mathcal{F}_b$, $\mathbf{r}_a^{p_i w}$ and $\mathbf{r}_b^{p_i w}$ respectively, Wahba's Problem is to find $\mathbf{C}_{ba} \in SO(3)$ that minimizes

$$J(\mathbf{C}_{ba}) = \tfrac{1}{2}\sum_{i=1}^{N} w_i \left(\mathbf{r}_b^{p_i w} - \mathbf{C}_{ba}\mathbf{r}_a^{p_i w}\right)^{\mathsf{T}}\left(\mathbf{r}_b^{p_i w} - \mathbf{C}_{ba}\mathbf{r}_a^{p_i w}\right) \tag{A.3}$$

where $0 < w_i < \infty$ are positive weights for $i = 1, 2, \ldots, N$. Notice that

$$\begin{aligned}
J(\mathbf{C}_{ba}) &= \tfrac{1}{2}\sum_{i=1}^{N} w_i \left(\mathbf{r}_b^{p_i w} - \mathbf{C}_{ba}\mathbf{r}_a^{p_i w}\right)^{\mathsf{T}}\left(\mathbf{r}_b^{p_i w} - \mathbf{C}_{ba}\mathbf{r}_a^{p_i w}\right) \\
&= \tfrac{1}{2}\sum_{i=1}^{N} w_i \left(\mathbf{r}_b^{p_i w^{\mathsf{T}}}\mathbf{r}_b^{p_i w} - 2\mathbf{r}_b^{p_i w^{\mathsf{T}}}\mathbf{C}_{ba}\mathbf{r}_a^{p_i w} + \mathbf{r}_a^{p_i w^{\mathsf{T}}}\mathbf{C}_{ba}^{\mathsf{T}}\mathbf{C}_{ba}\mathbf{r}_a^{p_i w}\right) \\
&= \tfrac{1}{2}\sum_{i=1}^{N} w_i \left(\mathbf{r}_b^{p_i w^{\mathsf{T}}}\mathbf{r}_b^{p_i w} - 2\mathbf{r}_b^{p_i w^{\mathsf{T}}}\mathbf{C}_{ba}\mathbf{r}_a^{p_i w} + \mathbf{r}_a^{p_i w^{\mathsf{T}}}\mathbf{r}_a^{p_i w}\right) \\
&= \tfrac{1}{2}\sum_{i=1}^{N} w_i \left(\mathbf{r}_b^{p_i w^{\mathsf{T}}}\mathbf{r}_b^{p_i w} + \mathbf{r}_a^{p_i w^{\mathsf{T}}}\mathbf{r}_a^{p_i w}\right) - \sum_{i=1}^{N} w_i \mathbf{r}_b^{p_i w^{\mathsf{T}}}\mathbf{C}_{ba}\mathbf{r}_a^{p_i w}
\end{aligned}$$

where $\mathbf{C}_{ba}^{\mathsf{T}}\mathbf{C}_{ba} = \mathbf{1}$. Since only the last term is a function of $\mathbf{C}_{ba}$, minimizing $J(\mathbf{C}_{ba})$ is equivalent to maximizing $\hat{J}(\mathbf{C}_{ba})$ where

$$\hat{J}(\mathbf{C}_{ba}) = \mathrm{tr}\left[\mathbf{C}_{ba}\mathbf{B}^{\mathsf{T}}\right]$$

and

$$\mathbf{B}^{\mathsf{T}} = \sum_{i=1}^{N} w_i \mathbf{r}_a^{p_i w}\mathbf{r}_b^{p_i w^{\mathsf{T}}}.$$

Now, consider N physical vectors $\underrightarrow{r}^{p_i w}$, $\underrightarrow{r}^{p_i z} \in \mathbb{P}$ and $\underrightarrow{r}^{zw} \in \mathbb{P}$ resolved in $\mathcal{F}_a$, $\mathcal{F}_b$, and

$\mathcal{F}_a$, yielding $\mathbf{r}_a^{p_i w}$, $\mathbf{r}_a^{p_i z}$, and $\mathbf{r}_a^{zw}$ respectively. The Point Could Alignment problem is to find $\mathbf{C}_{ba} \in SO(3)$ and $\mathbf{r}_a^{zw} \in \mathbb{R}^3$ that minimizes

$$J(\mathbf{C}_{ba}, \mathbf{r}_a^{zw}) = \tfrac{1}{2} \sum_{i=1}^{N} w_i \left(\mathbf{r}_b^{p_i z} - \mathbf{C}_{ba} \left(\mathbf{r}_a^{p_i w} - \mathbf{r}_a^{zw}\right)\right)^{\mathsf{T}} \left(\mathbf{r}_b^{p_i z} - \mathbf{C}_{ba} \left(\mathbf{r}_a^{p_i w} - \mathbf{r}_a^{zw}\right)\right) \tag{A.4}$$

where $0 < w_i < \infty$ are positive weights for $i = 1, 2, \ldots, N$.

This optimization problem, sometimes referred to as "Horn's method", was posed by Horn (1987) [48, 61] as well as Arun, Huang, and Blostein (1987) [49]. Note that

$$\begin{aligned} \mathbf{r}_a^{zw} &= -\mathbf{C}_{ba}^{\mathsf{T}} \mathbf{r}_b^{p_i z} + \mathbf{r}_a^{p_i w}, \\ \sum_{i=1}^{N} w_i \mathbf{r}_a^{zw} &= \sum_{i=1}^{N} w_i \left(-\mathbf{C}_{ba}^{\mathsf{T}} \mathbf{r}_b^{p_i z}\right) + \sum_{i=1}^{N} w_i \mathbf{r}_a^{p_i w}, \end{aligned}$$

Let

$$w = \sum_{i=1}^{N} w_i, \quad \mathbf{r}_b = \frac{1}{w} \sum_{i=1}^{N} w_i \mathbf{r}_b^{p_i z}, \quad \mathbf{r}_a = \frac{1}{w} \sum_{i=1}^{N} w_i \mathbf{r}_a^{p_i w}, \tag{A.5}$$

so that

$$\mathbf{r}_a^{zw} = -\mathbf{C}_{ba}^{\mathsf{T}} \mathbf{r}_b + \mathbf{r}_a.$$

The objective function can then be written as [48, 61]

$$\begin{aligned} J(\mathbf{C}_{ba}, \mathbf{r}_a^{zw}) &= \tfrac{1}{2} \sum_{i=1}^{N} w_i \left(\mathbf{r}_b^{p_i z} - \mathbf{C}_{ba} \left(\mathbf{r}_a^{p_i w} - \mathbf{r}_a^{zw}\right)\right)^{\mathsf{T}} \left(\mathbf{r}_b^{p_i z} - \mathbf{C}_{ba} \left(\mathbf{r}_a^{p_i w} - \mathbf{r}_a^{zw}\right)\right) \\ &= \tfrac{1}{2} \sum_{i=1}^{N} w_i \left(\mathbf{r}_b^{p_i z} - \mathbf{C}_{ba} \left(\mathbf{r}_a^{p_i w} + \mathbf{C}_{ba}^{\mathsf{T}} \mathbf{r}_b - \mathbf{r}_a\right)\right)^{\mathsf{T}} \left(\mathbf{r}_b^{p_i z} - \mathbf{C}_{ba} \left(\mathbf{r}_a^{p_i w} + \mathbf{C}_{ba}^{\mathsf{T}} \mathbf{r}_b - \mathbf{r}_a\right)\right) \\ &= \tfrac{1}{2} \sum_{i=1}^{N} w_i \left(\left(\mathbf{r}_b^{p_i z} - \mathbf{r}_b\right) - \mathbf{C}_{ba} \left(\mathbf{r}_a^{p_i w} - \mathbf{r}_a\right)\right)^{\mathsf{T}} \left(\left(\mathbf{r}_b^{p_i z} - \mathbf{r}_b\right) - \mathbf{C}_{ba} \left(\mathbf{r}_a^{p_i w} - \mathbf{r}_a\right)\right) \end{aligned}$$

The objective function is now no longer a function of $\mathbf{r}_a^{zw}$ explicitly,, allowing the cost function to be written as

$$J_1(\mathbf{C}_{ba}) = \tfrac{1}{2} \sum_{i=1}^{N} w_i \left(\left(\mathbf{r}_b^{p_i z} - \mathbf{r}_b\right) - \mathbf{C}_{ba} \left(\mathbf{r}_a^{p_i w} - \mathbf{r}_a\right)\right)^{\mathsf{T}} \left(\left(\mathbf{r}_b^{p_i z} - \mathbf{r}_b\right) - \mathbf{C}_{ba} \left(\mathbf{r}_a^{p_i w} - \mathbf{r}_a\right)\right).$$

This form of $J_1(\cdot)$ is of the same form as the objective function associated with Wahba's Problem. Minimizing $J_1(\cdot)$ as a function of $\mathbf{C}_{ba} \in SO(3)$ is equivalent to maximizing $\hat{J}_1(\cdot)$ as a function of $\mathbf{C}_{ba} \in SO(3)$ where

$$\hat{J}_1(\mathbf{C}_{ba}) = \text{tr}\left[\mathbf{C}_{ba}\mathbf{B}^\mathsf{T}\right]$$

and

$$\mathbf{B}^\mathsf{T} = \sum_{i=1}^{N} w_i \left(\mathbf{r}_a^{p_i w} - \mathbf{r}_a\right)\left(\mathbf{r}_b^{p_i z} - \mathbf{r}_b\right)^\mathsf{T}.$$

After finding the optimal $\mathbf{C}_{ba}$, the optimal $\mathbf{r}_a^{zw}$ is given by

$$\mathbf{r}_a^{zw} = -\mathbf{C}_{ba}^\mathsf{T}\mathbf{r}_b + \mathbf{r}_a,$$

where $\mathbf{r}_b$ and $\mathbf{r}_a$ are given in Equation (A.5).

A.0.3.1 Solution via SVD [62, 63, 64, 65]

Solving for $\mathbf{C}_{ba}$ involves maximizing $\hat{J}(\cdot)$ as a function of $\mathbf{C}_{ba} \in SO(3)$ where

$$\hat{J}(\mathbf{C}_{ba}) = \text{tr}\left[\mathbf{C}_{ba}\mathbf{B}^\mathsf{T}\right].$$

Consider a SVD of $\mathbf{B}$, namely,

$$\mathbf{B} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\mathsf{T},$$

where $\mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{1}$, $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{1}$, and $\mathbf{\Sigma} = \text{diag}\{\sigma_1, \sigma_2, \sigma_3\}$ where $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$. Matrices $\mathbf{V}$ and $\mathbf{U}$ are not elements of $SO(3)$, rather they are elements of $O(3)$. As such, although $\mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{1}$ and $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{1}$, the determinants are $\det\mathbf{V} = \pm 1$ and $\det\mathbf{U} = \pm 1$. Note that $\det\mathbf{B} = \det\mathbf{V}\det\mathbf{\Sigma}\det\mathbf{U}^\mathsf{T} = \sigma_1\sigma_2\sigma_3\det\mathbf{V}\det\mathbf{U}$. Because $\sigma_1 \geq \sigma_1 \geq \sigma_3 \geq 0$, when $\sigma_3 > 0$ (i.e., rank $\mathbf{B} = 3$) it follows that $\text{sign}[\det\mathbf{B}] = \det\mathbf{V}\det\mathbf{U}$. When $\det\mathbf{B} > 0$, $\det\mathbf{V}\det\mathbf{U} = +1$; when $\det\mathbf{B} < 0$, $\det\mathbf{V}\det\mathbf{U} = -1$.

Consider the modified SVD of $\mathbf{B}$,

$$\mathbf{B} = \bar{\mathbf{V}}\bar{\mathbf{\Sigma}}\bar{\mathbf{U}}^\mathsf{T},$$

where $\bar{\mathbf{V}} = \mathbf{V}\text{diag}\{1, 1, \det\mathbf{V}\}$, $\bar{\mathbf{U}} = \mathbf{U}\text{diag}\{1, 1, \det\mathbf{U}\}$, $\bar{\mathbf{\Sigma}} = \text{diag}\{\bar{\sigma}_1, \bar{\sigma}_2, \bar{\sigma}_3\}$ and $\bar{\sigma}_1 = \sigma_1$, $\bar{\sigma}_2 = \sigma_2$, $\bar{\sigma}_3 = \sigma_3\det\mathbf{V}\det\mathbf{U}$. Now $\bar{\mathbf{V}}$ and $\bar{\mathbf{U}}$ are elements of $SO(3)$.

For the reminder of the derivation it is assumed that $\det\mathbf{B} \geq 0$ so that $\bar{\sigma}_3 = \sigma_3\det\mathbf{V}\det\mathbf{U} \geq$

0.

Rewrite the objective function as

$$\hat{J}(\mathbf{C}_{ba}) = \mathrm{tr}\left[\mathbf{C}_{ba}\bar{\mathbf{U}}\bar{\mathbf{\Sigma}}\bar{\mathbf{V}}^{\mathsf{T}}\right] = \mathrm{tr}\left[\bar{\mathbf{V}}^{\mathsf{T}}\mathbf{C}_{ba}\bar{\mathbf{U}}\bar{\mathbf{\Sigma}}\right] = \mathrm{tr}\left[\mathbf{S}\bar{\mathbf{\Sigma}}\right]$$

where $\mathbf{S} = \bar{\mathbf{V}}^{\mathsf{T}}\mathbf{C}_{ba}\bar{\mathbf{U}}$.

Notice that

$$\begin{aligned}\mathbf{S}^{\mathsf{T}}\mathbf{S} &= \bar{\mathbf{U}}^{\mathsf{T}}\mathbf{C}_{ba}^{\mathsf{T}}\bar{\mathbf{V}}\bar{\mathbf{V}}^{\mathsf{T}}\mathbf{C}_{ba}\bar{\mathbf{U}} = \mathbf{1},\\ \det\mathbf{S} &= \det(\bar{\mathbf{V}}^{\mathsf{T}}\mathbf{C}_{ba}\bar{\mathbf{U}}) = \det(\bar{\mathbf{V}}^{\mathsf{T}})\det\mathbf{C}_{ba}\det\bar{\mathbf{U}} = +1,\end{aligned}$$

and thus $\mathbf{S} \in SO(3)$.

It is possible to show that $\mathbf{S} = \mathbf{1}$ maximizes $\hat{J}(\mathbf{C}_{ba}) = \mathrm{tr}\left[\mathbf{S}\bar{\mathbf{\Sigma}}\right]$. First, write $\bar{\mathbf{\Sigma}}$ as

$$\bar{\mathbf{\Sigma}} = \sum_{j=1}^{3}\bar{\sigma}_j\mathbf{1}_j\mathbf{1}_j^{\mathsf{T}},$$

where $\mathbf{1}_1^{\mathsf{T}} = [1\ \ 0\ \ 0]$, $\mathbf{1}_2^{\mathsf{T}} = [0\ \ 1\ \ 0]$, $\mathbf{1}_3^{\mathsf{T}} = [0\ \ 0\ \ 1]$. Then,

$$\begin{aligned}\hat{J}(\mathbf{C}_{ba}) = \mathrm{tr}\left[\mathbf{S}\bar{\mathbf{\Sigma}}\right] &= \mathrm{tr}\left[\mathbf{S}\sum_{j=1}^{3}\bar{\sigma}_j\mathbf{1}_j\mathbf{1}_j^{\mathsf{T}}\right] = \mathrm{tr}\left[\sum_{j=1}^{3}\bar{\sigma}_j\mathbf{S}\mathbf{1}_j\mathbf{1}_j^{\mathsf{T}}\right]\\ &= \mathrm{tr}\left[\sum_{j=1}^{3}\bar{\sigma}_j\mathbf{1}_j^{\mathsf{T}}\mathbf{S}\mathbf{1}_j\right] = \sum_{j=1}^{3}\bar{\sigma}_j\mathbf{1}_j^{\mathsf{T}}\mathbf{S}\mathbf{1}_j.\end{aligned}$$

The Cauchy-Schwarz inequality, $|\mathbf{u}^{\mathsf{T}}\mathbf{v}| \leq \|\mathbf{u}\|_2\|\mathbf{v}\|_2$, and $\bar{\sigma}_1 \geq \bar{\sigma}_2 \geq \bar{\sigma}_3 \geq 0$, gives

$$\hat{J}(\mathbf{C}_{ba}) = \mathrm{tr}\left[\mathbf{S}\bar{\mathbf{\Sigma}}\right] = \sum_{j=1}^{3}\bar{\sigma}_j\mathbf{1}_j^{\mathsf{T}}\mathbf{S}\mathbf{1}_j \leq \sum_{j=1}^{3}\bar{\sigma}_j\|\mathbf{1}_j\|_2\|\mathbf{S}\mathbf{1}_j\|_2.$$

Because $\mathbf{S} \in SO(3)$,

$$\|\mathbf{S}\mathbf{1}_j\|_2 = \sqrt{\mathbf{1}_j^{\mathsf{T}}\mathbf{S}^{\mathsf{T}}\mathbf{S}\mathbf{1}_j} = \sqrt{\mathbf{1}_j^{\mathsf{T}}\mathbf{1}_j} = 1,$$

and as such,

$$\hat{J}(\mathbf{C}_{ba}) = \mathrm{tr}\left[\mathbf{S}\bar{\mathbf{\Sigma}}\right] \leq \sum_{j=1}^{3}\bar{\sigma}_j\|\mathbf{1}_j\|_2\|\mathbf{S}\mathbf{1}_j\|_2 = \sum_{j=1}^{3}\bar{\sigma}_j = \mathrm{tr}\bar{\mathbf{\Sigma}}.$$

Therefore, $\mathbf{S} = \mathbf{1}$ maximizes $\hat{J}(\cdot)$.

Now, recall that $\mathbf{S} = \bar{\mathbf{V}}^{\mathsf{T}}\mathbf{C}_{ba}\bar{\mathbf{U}}$. Thus,

$$\mathbf{1} = \bar{\mathbf{V}}^{\mathsf{T}}\mathbf{C}_{ba}\bar{\mathbf{U}},$$

$$\mathbf{C}_{ba} = \bar{\mathbf{V}}\bar{\mathbf{U}}^{\mathsf{T}} = \mathbf{V}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det\mathbf{V}\det\mathbf{U} \end{bmatrix}\mathbf{U}^{\mathsf{T}}.$$

A.0.4 Point-to-Plane Registration via Linearization

Given $\mathbf{r}_{z_k}^{p_j z_k}$ and $\mathbf{r}_{z_{k-1}}^{q_j z_{k-1}}$ the cost function to minimize is

$$J_k(\mathbf{r}_k^{s_k s_k - 1}, \mathbf{C}_{kk-1}) = \sum_{j=1}^{M}(\mathbf{e}_k^{j\,\mathsf{T}}\mathbf{n}_k^j)^2$$

where the residual, $\mathbf{e}_k^j$ is

$$\mathbf{e}_k^j = \mathbf{r}_{z_k}^{q_j z_k} - (\mathbf{C}_{kk-1}\mathbf{r}_{z_{k-1}}^{p_j z_{k-1}} - \mathbf{r}_{z_k}^{z_k z_{k-1}}), \tag{A.6}$$

Notice that this is a nonlinear least squares problem and therefore requires Jacobian computation to solve. If the rotation is small however, i.e. $\cos(\theta) \approx 1$ and $\sin(\theta) \approx 0$, the problem can be converted to a linear least squares problem as follows.

$$\mathbf{C}_{kk-1} \approx \begin{bmatrix} 1 & -\psi & \theta \\ \psi & 1 & -\phi \\ -\theta & \phi & 1 \end{bmatrix} \tag{A.7}$$

where ψ, θ, ϕ are yaw, pitch, roll respectively. Substitute A.7 into A.0.4 to obtain

$$
\begin{aligned}
J_k^j(\mathbf{r}_k^{s_k s_k-1}, \mathbf{C}_{kk-1}) &= \left(\left(\mathbf{r}_{z_k}^{q_j z_k} - \left(\begin{bmatrix} 1 & -\psi & \theta \\ \psi & 1 & -\phi \\ -\theta & \phi & 1 \end{bmatrix} \mathbf{r}_{z_{k-1}}^{p_j z_{k-1}} - \mathbf{r}_{z_k}^{z_k z_{k-1}} \right) \right)^{\mathsf{T}} \mathbf{n}_k^j \right)^2 \\
&= \begin{bmatrix} q_{jx} - (p_{jx} - p_{jy}\psi + p_{jz}\theta - t_x) \\ q_{jy} - (p_{jx}\psi + p_{jy} - p_{jz}\phi - t_y) \\ q_{jz} - (-p_{jx}\theta + \partial_{jy}\phi + p_{jz} - t_z) \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} n_{jx} \\ n_{jy} \\ n_{jz} \end{bmatrix} \\
&= n_{jx}q_{jx} - n_{jx}p_{jx} + n_{jx}p_{jy}\psi - n_{jx}p_{jz}\theta + n_{jx}t_x \\
&\quad + n_{jy}q_{jy} - n_{jy}p_{jx}\psi - n_{jy}p_{jy} + n_{jy}p_{jz}\phi + n_{jy}t_y \\
&\quad + n_{jz}q_{jz} + n_{jz}p_{jx}\theta - n_{jz}p_{jy}\phi - n_{jz}p_{jz} + n_{jz}t_z \\
&= [(n_{jy}p_{jz} - n_{jz}p_{jy})\phi + (n_{jz}p_{jx} - n_{jx}p_{jz})\theta + (n_{jx}p_{jy} - n_{jy}p_{jx})\psi \\
&\quad + n_{jx}t_x + n_{jy}t_y + n_{jz}t_z] \\
&\quad - [n_{jx}p_{jx} - n_{jx}q_{jx} + n_{jy}p_{jy} - n_{jy}q_{jy} + n_{jz}p_{jz} - n_{jz}q_{jz}] .
\end{aligned}
$$

Considering that there at M total pairs of point correspondences, stack all the point costs into the matrix expression

$$J_k(\mathbf{r}_k^{s_k s_k-1}, \mathbf{C}_{kk-1}) = \mathbf{A}\mathbf{x} - \mathbf{b}$$

where

$$
\mathbf{A} = \begin{bmatrix} (n_{1y}p_{1z} - n_{1z}p_{1y}) & (n_{1z}p_{1x} - n_{1x}p_{1z}) & (n_{1x}p_{1y} - n_{1y}p_{1x}) & n_{1x} & n_{1y} & n_{1z} \\ (n_{2y}p_{2z} - n_{2z}p_{2y}) & (n_{2z}p_{2x} - n_{2x}p_{2z}) & (n_{2x}p_{2y} - n_{2y}p_{2x}) & n_{2x} & n_{2y} & n_{2z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ (n_{My}p_{Mz} - n_{Mz}p_{My}) & (n_{Mz}p_{Mx} - n_{Mx}p_{Mz}) & (n_{Mx}p_{My} - n_{My}p_{Mx}) & n_{Mx} & n_{My} & n_{Mz} \end{bmatrix}
$$

$$
\mathbf{b} = \begin{bmatrix} n_{1x}p_{1x} - n_{1x}q_{1x} + n_{1y}p_{1y} - n_{1y}q_{1y} + n_{1z}p_{1z} - n_{1z}q_{1z} \\ n_{2x}p_{1x} - n_{2x}q_{2x} + n_{2y}p_{2y} - n_{2y}q_{2y} + n_{2z}p_{2z} - n_{2z}q_{2z} \\ \vdots \\ n_{Mx}p_{Mx} - n_{Mx}q_{Mx} + n_{My}p_{My} - n_{My}q_{My} + n_{Mz}p_{Mz} - n_{Mz}q_{Mz} \end{bmatrix},
$$

$$
\mathbf{x} = \begin{bmatrix} \phi & \theta & \psi & t_x & t_y & t_z \end{bmatrix}^{\mathsf{T}} .
$$

Bibliography

[1] M. Majji, J. Davis, J. Doebbler, J. Junkins, B. Macomber, M. Vavrina, and J. Vian, "Terrain Mapping and Landing Operations Using Vision Based Navigation Systems," *AIAA Guidance, Navigation, and Control Conference, Portland, OR, August 08-10, 2010*.

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

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, pp. 1309–1332, Dec 2016.

[4] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spacial relationships in robotics," *Autonomous Robot Vehicles*, pp. 167–193, January 1990.

[5] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics Automation Magazine*, vol. 18, pp. 80–92, Dec 2011.

[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov 2004.

[7] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 404–417, Springer Berlin Heidelberg, 2006.

[8] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, Nov 2011.

[9] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), (Berlin, Heidelberg), pp. 778–792, Springer Berlin Heidelberg, 2010.

[10] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234, Nov 2007.

[11] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the rgb-d slam system," in *2012 IEEE International Conference on Robotics and Automation*, pp. 1691–1696, May 2012.

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, pp. 1147–1163, Oct 2015.

[13] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *CoRR*, vol. abs/1610.06475, 2016.

[14] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "Pl-slam: Real-time monocular visual slam with points and lines," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4503–4508, May 2017.

[15] S. Maity, A. Saha, and B. Bhowmick, "Edge slam: Edge points based monocular visual slam," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 2408–2417, Oct 2017.

[16] D. Filliat, "A visual bag of words method for interactive qualitative localization and mapping," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3921–3926, April 2007.

[17] D. Galvez-Lpez and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, pp. 1188–1197, Oct 2012.

[18] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 International Conference on Computer Vision*, pp. 2320–2327, Nov 2011.

[19] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *2013 IEEE International Conference on Computer Vision*, pp. 1449–1456, Dec 2013.

[20] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 834–849, Springer International Publishing, 2014.

[21] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 611–625, March 2018.

[22] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, pp. 401–416, Feb 2017.

[23] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2174–2181, May 2015.

[24] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, June 2012.

[25] 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, May 2016.

[26] S. Agarwal, K. Mierle, and Others, "Ceres solver." http://ceres-solver.org.

[27] E. B. Olson, "Real-time correlative scan matching," in *2009 IEEE International Conference on Robotics and Automation*, pp. 4387–4393, May 2009.

[28] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pp. 2724–2729 vol.3, Apr 1991.

[29] M. Alshawa, "Icl: Iterative closest line a novel point cloud registration algorithm based on linear features," vol. 10, 07 2007.

[30] F. Lu and E. E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 935–938, Jun 1994.

[31] L. Montesano, J. Minguez, and L. Montano, "Probabilistic scan matching for motion estimation in unstructured environments," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3499–3504, Aug 2005.

[32] A. Diosi and L. Kleeman, "Laser scan matching in polar coordinates with application to slam," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3317–3322, Aug 2005.

[33] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[34] R. Eustice, M. Walter, and J. Leonard, "Sparse extended information filters: insights into sparsification," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3281–3288, Aug 2005.

[35] S. J. Julier and J. K. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 4, pp. 4238–4243 vol.4, 2001.

[36] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, pp. 1365–1378, Dec 2008.

[37] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb 2001.

[38] K. Konolige, G. Grisetti, R. Kmmerle, 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, Oct 2010.

[39] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[40] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[41] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[42] A. Bjrck, *Numerical Methods in Matrix Computations*. Springer International Publishing, 2015.

[43] B. Büttgen, T. Oggier, M. Lehmann, R. Kaufmann, and F. Lustenberger, "Ccd / cmos lock-in pixel for range imaging : Challenges , limitations and state-ofthe-art," 2005.

[44] C. L. Glennie and D. D. Lichti, "Static calibration and analysis of the velodyne hdl-64e s2 for high accuracy mobile scanning," *Remote Sensing*, vol. 2, pp. 1610–1624, 2010.

[45] R. Goeddel, C. Kershaw, J. Serafin, and E. Olson, "Flat2d: Fast localization from approximate transformation into 2d," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1932–1939, Oct 2016.

[46] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152, 2001.

[47] G. Wahba, "Problem 65-1, a least-squares estimate of satellite attitude," *SIAM Review*, vol. 7, p. 409, July 1965.

[48] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[49] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, 1987.

[50] J. L. Junkins, M. Majji, B. Macomber, J. Davis, J. Doebbler, and R. Nosterk, "Small Body Proximity Sensing With a Novel HD3D LADAR System," *ASS Guidance and Control Conference, Breckenridge, CO, Febuary 5-9, 2011.*

[51] X. I. Wong and M. Majji, "A Structured Light System for Relative Navigation Applications," *IEEE Sensors Journal*, vol. 16, pp. 6662–6679, Sept 2016.

[52] H. Schaub and J. L. Junkins, *Analytical Mechanics of Space Systems*. Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2nd ed., 2009.

[53] P. C. Hughes, *Spacecraft Attitude Dynamics*. Mineola, New York: Dover, 2nd ed., 2004.

[54] L. Ljung, ed., *System Identification (2Nd Ed.): Theory for the User*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[55] K. lim Low, "Linear least-squares optimization for point-toplane icp surface registration," tech. rep., 2004.

[56] K. Khoshelham, "Closed-form solutions for estimating a rigid motion from plane correspondences extracted from point clouds," pp. 78–91 vol.114, Apr 2016.

[57] A. Censi, "An icp variant using a point-to-line metric," in *2008 IEEE International Conference on Robotics and Automation*, pp. 19–25, May 2008.

[58] Y. Y. David G. Luenberger, *Linear and Nonlinear Programming*. Springer International Publishing, 2016.

[59] "Bentley systems, incorporated." https://www.bentley.com/en/products/product-line/reality-modeling-software/bentley-descartes.

[60] D. Ji, J. Cheng, and Y. Xu, "An extracting method of corner points from laser sensor readings," in *2018 37th Chinese Control Conference (CCC)*, pp. 4962–4967, July 2018.

[61] B. K. P. Horn, "Closed-form Solution of Absolute Orientation Using Orthonormal Matrices," *Journal of the Optical Society of America A*, vol. 5, no. 7, pp. 1127–1135, 1987.

[62] F. L. Markley, "Attitude Determination Using Vector Observations and the Singular Value Decomposition," *Journal of the Astronautical Sciences*, vol. 36, no. 3, pp. 245–258, 1988.

[63] K. Kanatani, "Analysis of 3-D Rotation Filtering," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 543–549, 1994.

[64] A. de Ruiter and J. R. Forbes, "On the Solution of Wahba's Problem on $SO(n)$," *Journal of the Astronautical Sciences*, vol. 60, no. 1, pp. 1–31, 2015.

[65] F. L. Markley and ohn L Crassidis, *Fundamentals of Spacecraft Attitude Determination and Control*. New York: Springer, 2014.