Sensor Fusion and Online Calibration of an Ambulatory Backpack System for Indoor Mobile Mapping

By

Nicholas Giovanni Corso

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in
Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Avideh Zakhor, Chair
Professor Ronald Fearing
Professor Kyle Steinfeld

Spring 2016

Sensor Fusion and Online Calibration of an Ambulatory Backpack System for Indoor Mobile Mapping

Abstract

Sensor Fusion and Online Calibration of an Ambulatory Backpack System for Indoor Mobile Mapping

by

Nicholas Giovanni Corso

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Avideh Zakhor, Chair

GPS-denied indoor mobile mapping has been an active area of research for many years. With applications such as historical preservation, entertainment, and augmented reality, the demand for both fast and accurate scanning technologies has dramatically increased. In this thesis, we present two algorithmic pipelines for GPS-denied indoor mobile 3D mapping using an ambulatory backpack system. By mounting scanning equipment on a backpack system, a human operator can traverse the interior of a building to produce a high-quality 3D reconstruction. In each of our presented algorithmic pipelines, data from a number of 2D laser scanners, a camera, and an IMU is fused together to track the 3D position of the system as the operator traverses an unknown environment.

This thesis presents a number of novel contributions for indoor GPS-denied 2.5 and 3D mobile mapping using a number of 2D laser scanners, a camera, and an IMU. First, for 3D mapping we develop a tightly coupled EKF estimator for fusing data from all sensors into a single optimized 3D trajectory. By formulating each sensor's contributions independently, we demonstrate a modular algorithm that easily scales to an arbitrary number of 2D laser scanners. In contrast to existing work that either assumes a known fixed map or limits the environment to a set of axis aligned planes, we demonstrate the ability to map environments containing horizontal and vertical planes of arbitrary orientation with no a priori information. Additionally, through timing and complexity analysis, we demonstrate that the runtime of the proposed EKF estimator is only linear in the acquisition time. Secondly, by including in our EKF estimator the laser scanner's spatial and temporal calibration parameters, we present a novel laser calibration methodology. Through simulated and real-world data, we validate that the proposed algorithms are capable of calibrating both the extrinsic and temporal misalignments present in our system's laser data. Lastly, we address the scalability of the proposed approach by utilizing a graph optimization post processing step that overcomes any accumulated drift in the EKF estimator. We then validate the proposed 3D end-to-end localization system using 3 multi-story datasets collected from real-world environments. The system's reconstructions are compared against CAD drawings of the buildings and are shown to achieve an intersection over union of over 96% on all datasets. Lastly, we demonstrate accuracy improvements over our 2.5D methods using a comparison test against data collected

with a static scanner.

In addition to 3D mapping, we also present a methodology for 2.5D mapping with three novel contributions. First, we present a method for automatically segmenting barometric pressure data based on the floor of the building it was collected from. Specifically, by using Bayesian non-parametrics we are able to demonstrate simultaneous floor detection and the corresponding data segmentation. The data segmentation is then used to extend classical 2D particle filtering across any number of discrete building stories. Secondly, we demonstrate a genetic scan matching algorithm used to estimate loop closure constraints even without an accurate initial condition. Through simulation and real-world experiments we show an improvement over state of the art scan matching techniques. Next, we present two metrics that are used to validate the results of the genetic scan matching algorithm. We use both a correlation and shape metric to demonstrate robust and accurate validation of loop closure constraints in indoor environments. Lastly, we compare and characterize the performance of the proposed 3D and 2.5D mapping techniques developed in this thesis. Although the 2.5D mapping techniques are more computationally lightweight, we show that the accuracy of system is significantly improved using the 3D mapping algorithm.

To Sindy.

Contents

List of Figures

Acknowledgements

My time at the University of California, Berkeley has been nothing short of spectacular. I knew that a rewarding intellectual adventure was in store with me from the very first moment I stepped onto campus. The knowledgeable faculty, diverse base of students, and excellent lab mates have made my studies nothing short of fantastic.

I would first like to thank my advisor Professor Avideh Zakhor. Her guidance and mentorship always drove me to overcome every challenge I faced. My knowledge base and understanding of the academic process flourished under her guidance. Furthermore, I would like to thank Professors Pieter Abbeel, Ronald Fearing, and Kyle Steinfeld for serving on my qualification exam and dissertation committees. Each one of them provided unique insight and feedback that helped to shape the direction of my dissertation.

This research was made possible through numerous government grants including: AFOSR Grants FA9550-08-1-0168 and FA9550-12-1-0299, Army Research Office Grants W911NF-07-1-0471 and W911NF-11-1-088, DOE Grant 2011-DN-ARI049-03, DOE Grant DE-AR0000331, and NSF Grant CMMI-1427096. I am extremely grateful for the financial support that made this dissertation possible.

Next, I would like to thank all my colleagues in the Video and Image Processing Lab. John Kua, Michael Krishnan, Ricardo Garcia, Shangliang Jiang, Shicong Yang, Richard Zhang, Joe Menke, David Fridovich-Keil, Erik Nelson, and Brian Nemsick. I would like to specially recognize Eric Turner for being my partner in crime throughout the graduate school experience.

Finally I would like to thank my family for all of their support over the years. My parents Matthew and Jeri always made sure that I had the drive to continue on and the perspective to remember what is important in life. Most importantly, I would like to thank my wife Sindy for supporting me through the long hours, stress, and general shirking of household duties. Without you, I would never have survived.

Chapter 1

Introduction

Recent years has seen a great interest in the modeling of urban indoor and outdoor environments using mapping technology. By utilizing sensor data captured from a suite of sensors and tracking the position of the scanner, a full featured reconstruction of the environment can be created. Modeling, mapping, and reconstruction of indoor and outdoor environments has many applications in the historical preservation, emergency response, mapping, navigation, energy simulation, and entertainment industries. As such, there has been significant motivation to increase accuracy and decrease the acquisition time for mapping systems.

Indoor mapping solutions cover a wide variety of platforms and applications. The available modalities for indoor mapping can be viewed as a trade-off between portability and data fidelity. Figure 1.1 illustrates the spectrum of indoor mapping solutions. On the far left of the spectrum, static scanning provides the highest data quality but is the least portable. During static scanning, a 3D laser scanning station is placed on a tripod and a small section of the environment is captured with high detail. The tripod is then moved around and the process is repeated until the entire area has been captured. The small 3D point clouds can then be stitched together to build a single, unified representation of the environment. Stitching is typically achieved either by placing small markers throughout the environment or a combination of manual intervention and point matching techniques. While this process is accurate and reliable, it can be both slow and invasive.

Moving more towards portability, wheeled platforms, such as push-carts, can carry heavy equipment such as long range laser scanners [1] or high quality inertial systems [2] in order or provide high accuracy reconstructions of the environment. Furthermore, wheel encoders have be employed to provide accurate, low-cost odometry for positioning in mapping applications [3]. While wheeled systems provide many advantages, complex terrain, such as staircases, often provide significant challenges for data capture and reconstruction.

To address the shortcomings of cart-based solutions, scanning equipment is often mounted on human operators and carried through the environment [4–8]. By mounting the scanning equipment on an ambulatory platform, the operator is able to map any environment a human is able to safely traverse. This allows the mapping systems to easily scale to multistory buildings in a single data collection. Human mounted mobile mapping systems also present many unique challenges. The human gait is considerably more complex than a wheeled system and thus fusing data from multiple sensors can be more challenging. Additionally, ambulatory systems are not capable of mapping locations, such as a collapsed building, that

Figure 1.1: The spectrum of indoor mapping solutions. The existing solutions can be viewed as a trade-off between portability and data fidelity. In one extreme, static scanning solutions provide the highest quality data but are slow and cumbersome to use. On the other end of the spectrum, R-GBD depth cameras are small and mobile, but often contain noisy sensing technologies.

are unsafe for a human operator.

Moving further right on the spectrum, autonomous robotic platforms have also been employed for mobile mapping applications. With the widespread availability of low-cost drones, unmanned aerial vehicles (UAVs) have become a popular choice [9–11]. UAVs are lightweight and capable of traversing both hazardous environments and locations humans may be unable to reach. UAVs intended for indoor applications are often limited to small payloads, and as a result battery life is typically a limiting factor for scalability. Furthermore many indoor environments, such as narrow doorways and HVAC equipment, present challenging aerodynamic problems that can disrupt a UAV's flight controller.

Finally, at the other extreme, hand-held devices have become a popular selection for fast mapping solutions [12–14]. Lightweight, hand-held devices offer unparalleled portability, but often only contain noisy and short range sensing technologies. To reduce weight and power consumption many systems utilize LED time-of-flight depth sensors or structured light projectors. These types of sensors are capable of capturing the 3D shape of the environment, but the high sensor noise and short range can limit their usefulness in many mapping applications.

In this thesis, we utilize an ambulatory backpack data collection system to map indoor GPS-denied environments. By using a backpack system, we are able to strike a balance between portability and data fidelity. Backpack systems are capable of carrying a large suite of high-quality sensors while still being able to seamlessly traverse complicated terrain without being limited by short battery life. For large-scale indoor mapping, it is crucial that the data acquisition process be both rapid and provide the flexibility to handle many types of environments.

1.1 Simultaneous Localization and Mapping

In contrast to static scanning technologies, mobile scanning systems are generally faster than static scanning techniques. While data collection is more rapid, the process of stitching together sensor data is considerably more complex. Mobile systems are almost constantly in motion and thus the position and orientation of the system must be accurately tracked in order to assemble the sensor readings during reconstruction. For many outdoor mapping systems, GPS/INS technology is used to solve the tracking problem [15–18]. However, in dense urban environments the "urban canyon" effect can limit the accuracy of GPS sensors. Furthermore, environments such as building interiors suffer from poor signal strength and multi-path interference making GPS/INS tracking solutions impractical. In these environments, mobile mapping systems typically rely on simultaneous localization and mapping (SLAM) algorithms for positioning. Common to all SLAM algorithms, the mobile mapping system must both create a map of the environment while simultaneously tracking its position within that map.

The SLAM problem has been studied extensively in the academic literature and many algorithms have been proposed to solve the problem [19–23]. SLAM solutions can logically be split into two mostly orthogonal steps. First, new sensor data must be referenced against the current map of the environment using a data association algorithm. Then, a back-end optimization framework is required to fuse information from the available sensors and update the system's estimates of its surrounding environment and position relative to it. As the data association algorithm is generally defined by the type of sensing modality, more emphasis is generally placed on the back-end optimization in the SLAM literature. For the remainder of this section, the term SLAM will refer to the back-end optimization step.

SLAM solutions can be roughly grouped into four categories: particle filtering, graph optimization, least-squares filtering, and database based solutions. Sometimes termed "fingerprinting" techniques, database based solutions maintain a sparse representation of the environment in an external database that can be queried similar to GPS technologies. Common fingerprinting signals include examples such as 802.11 WiFi [22], FM radio [24], and GSM cellular [25]. While database techniques have been proven to provide reliable positioning information in indoor environments, the need to pre-survey a location and build a database of fingerprints makes it unattractive for large-scale mapping.

Least-squares filtering techniques, such as the sliding window filter [26] or Kalman filter variants [27–29], are another extremely popular choice in the SLAM community due to straightforward implementation and natural application to real-time systems. Least-squares filtering techniques however rely on two assumptions: the system dynamics are locally linear and all sensor noise is Gaussian. To this end, the SLAM problem is modeled as a system of Gaussian random variables that are operated on by a series of linear transforms. As Gaussian are characterized by a mean vector and covariance matrix, least-squares filters must keep an $N \times N$ matrix of correlations. As the number of variables increases, the complexity of least-squares filters can become intractably large for real-time operation.

In order to handle nonlinear or non-Gaussian dynamic models, particle filters have also been successfully used to solve the SLAM problem [30–32]. Instead of using an explicit formulation of the SLAM posterior, particle filters use Monte Carlo methods to estimate it from a set of discrete samples or "particles." Particle filters have been utilized in both 2D

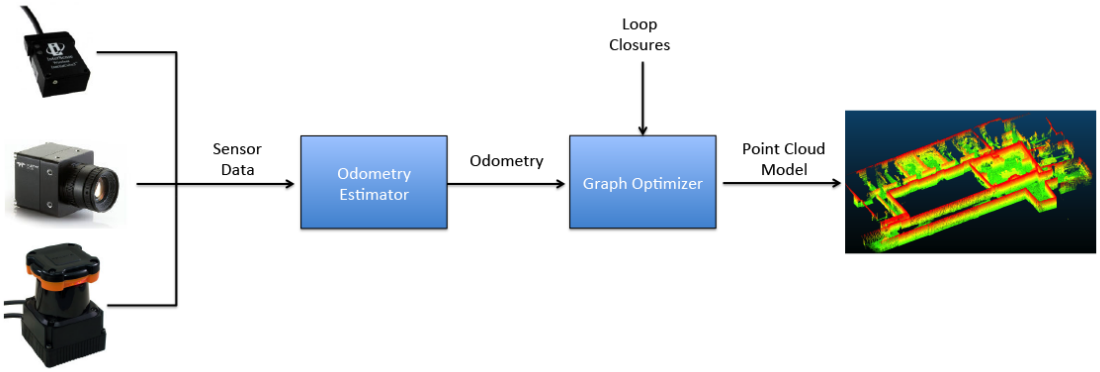


Figure 1.2: A high level flowchart of the two-stage SLAM pipeline. First, sensor data is fused into an initial estimate of the trajectory of the system using some odometry estimation algorithm. Then, loop closure constraints are detected and combined with the odometry estimated into a single optimized trajectory using a graph optimizer. Finally, a model is created by assembling the laser readings into a large point cloud.

and 3D grid mapping applications. While particle filters are robust to nonlinearities, as the dimensionality of the estimation problem increases the number of particles required increases substantially. This makes particle filters ill suited for estimation problems involving many variables.

More recently, non-linear graph based approaches have also been a popular choice to solve both incremental and batch SLAM problems [23, 33–35]. By formulating the estimation problem as a network of nodes connected by a sparse set of edge constraints, graph based optimizers are able to efficiently solve the SLAM problem. Batch optimizers such, as TORO [36] or g2o [34], or incremental solvers, such as iSAM [23, 33, 35], have been successfully utilized in both pose graph and landmark based SLAM applications.

Multi-modal approaches are also commonly used to solve the SLAM problem. By combining high-accuracy least-squares algorithms with sparse graph optimization methods, SLAM systems are able to naturally fuse sensor data with algorithms such as Kalman filters without succumbing to the curse of dimensionality. In this thesis, we present two SLAM algorithms for an ambulatory backpack system. In both instances, we take a two-stage approach to the SLAM process. A high level flowchart of this processes is shown in Fig. 1.2. First, data from the individual sensors are combined to create an initial estimate of the path the operator walked through the environment. Then, we detect locations that were revisited and combine those with the initial estimate via graph optimization. Using the optimized trajectory, we combine the laser data readings into a 3D representation of the environment.

1.2 Odometry Algorithms

Odometry estimation algorithms are an important component in the two-stage SLAM pipeline. Non-linear graph optimizers require an accurate initial condition, and thus accurate odometry is an important consideration when designing a SLAM algorithm. Furthermore, due to

computational constraints many estimators such as the EKF and sliding window approaches must marginalize out old observations to maintain scalability in large environments. In doing so, the estimator looses the ability to perform long-term loop closure and the reconstructed trajectory begins to accumulate drift.

Odometry algorithms are a widely studied problem in the SLAM literature. With the rise of low-cost cameras-IMU systems, visual-inertial odometry (VIO) has become a popular method for high-accuracy odometry [37–42]. These types of odometry systems have been shown to provide high-accuracy odometry estimates even with extremely low-quality sensors. In addition, the VIO are often combined with online calibration algorithms to auto-calibrate the extrinsic, intrinsic, and temporal aspects of the system. Unfortunately, because these algorithms rely on tracking interesting points in the camera imagery, they often have trouble in areas that contain few visual features or have extremely poor lighting.

RGB-D cameras have also been utilized for odometry estimation is SLAM systems [12, 13, 43–45]. RGB-D cameras are capable of provided both imagery and depth information for a camera using either structured light projectors or a LED time-of-flight sensor. RGB-D cameras are typically able to produce a high volume of noise data for estimation. Techniques, such as KinectFusion [12,13], intelligently average the noisy sensor data to produce high-quality meshes of scanned objects. These types of approaches however typically suffer from 2 shortcomings. In order to handle the large quantity of data, methods such as KinectFusion must maintain a sliding volume of interest which leads to drift in the final trajectory. Additionally, RGB-D sensors typically have ranges around 4-6m and thus are ill-suited for large-scale mapping.

Finally, laser scanners have also been considered for 2D odometry estimation [8, 46, 47]. By matching successive laser readings from a 2D laser scanner, a scan matching algorithm, such as ICP, can be used to estimate the 2D trajectory of a system. While these types of approaches have seen great success in 2D SLAM applications, they are not always appropriate for full 3D SLAM without making assumptions about the environment.

1.3 Sensor Calibration

Common to all multi-sensor sensors is the importance of calibrating sensors with respect to one another. Each sensor captures data referenced against its own internal coordinate system and timestamps. In order to accurately fuse data from multiple sensors, the transformations and timing offsets must be correctly incorporated into the modeling process. If incorrect calibration is used, the recovered trajectory can be erroneous and the reconstructed model will not be self-consistent.

Calibration is generally broken into three groups: extrinsic, intrinsic, and temporal. Extrinsic calibration refers to the physical location of the sensors relative to one another. Extrinsic calibration is parameterized using a rotation and translation between sensor coordinate frames. Next, intrinsic calibration describes the internal parameters of a sensor. For example, a camera's intrinsic calibration is characterized by the lens's focal length and distortion parameters. Finally, temporal calibration describes how the timestamps of a sensor relate against a common system clock. This is especially important when fusing inertial data with other sensor due to the high dynamic gyroscope motion an ambulatory platform

undergoes.

Calibration procedures have been explored in both offline and online contexts. For example, Zhang [48] presented a widespread offline calibration procedure that calculates a camera's intrinsic parameters using a planar target. Similarly, Zhang and Pless [49] presented an offline method for computing the extrinsic calibration between a camera and laser scanner also using a planar target. Using a specially created 3D target, Bok et al. [50] built upon the work of Zhang and Pless and presented an improved system of calibrating a laser and camera sensor.

Online calibration of both the intrinsic and extrinsic parameters between an IMU and camera has been extensively studied. In [51], Mirzaei et al. presented an Extended Kalman Filter estimator for the rotation and translation between an IMU and camera using a checkerboard target. Li et al. expanded upon this work by presenting a visual inertial odometry (VIO) system that estimated the rotation and translation with no predefined target. [40,52]. Building on this, Li. et al. further expanded the VIO estimator to also compute the timing offset between the IMU and camera, the camera intrinsic calibration model, and the readout time of the camera's rolling shutter.

Laser extrinsic and temporal calibration has also been previously explored in the literature. In [53] Levinson and Thurn presented a method for calibrating 3D laser scanners and cameras by detecting mis-calibrations online and updating the sensor parameters in an arbitrary environment. The temporal offset between a 2D laser scanner and IMU was considered by Rehder et al. in [54]. Although the method presented by Rehder et al. is capable of estimating the timing offset and extrinsic calibration between a 2D laser and an IMU, the authors assume that a VIO system is also available to compute a sufficiently accurate initial estimate of the systems trajectory. This makes the algorithm difficult to apply for a system that only contains lasers and an IMU. He et al. [55] explored direct extrinsic calibration between 2D laser scanners by optimizing for the rotation and translation during the point cloud reconstruction process.

Previous calibration methods do not consider warping that occurs when the laser scanner is moving. The majority of 2D laser scanners are constructed using a single laser and a mirror that rotates around a central axis. A scan is generated by taking sequential measurements across one rotation of the central mirror and is assigned a single timestamp. This creates an effect similar to the rolling shutter in most low-end CMOS camera sensors. When the laser scanner is subjected to high linear or rotational velocity during the scanning process, the resulting scans become significantly warped. By modeling the laser scanner as a rolling shutter sensor and calibrating the read-out time, the scanner's data can be correctly dewarped.

1.4 Contributions and Thesis Organization

In this dissertation we present two methods for localizing an ambulatory mobile mapping system for the purpose of building large-scale, accurate 3D models of GPS-denied indoor environments. The objective of the algorithms presented in this dissertation is to be able to walk around a building using the backpack systems shown in Chapter 2 and produce a high quality 3D point cloud model with no a priori information about the environment. Figure 1.3 shows an example of a point cloud created from a large warehouse environment

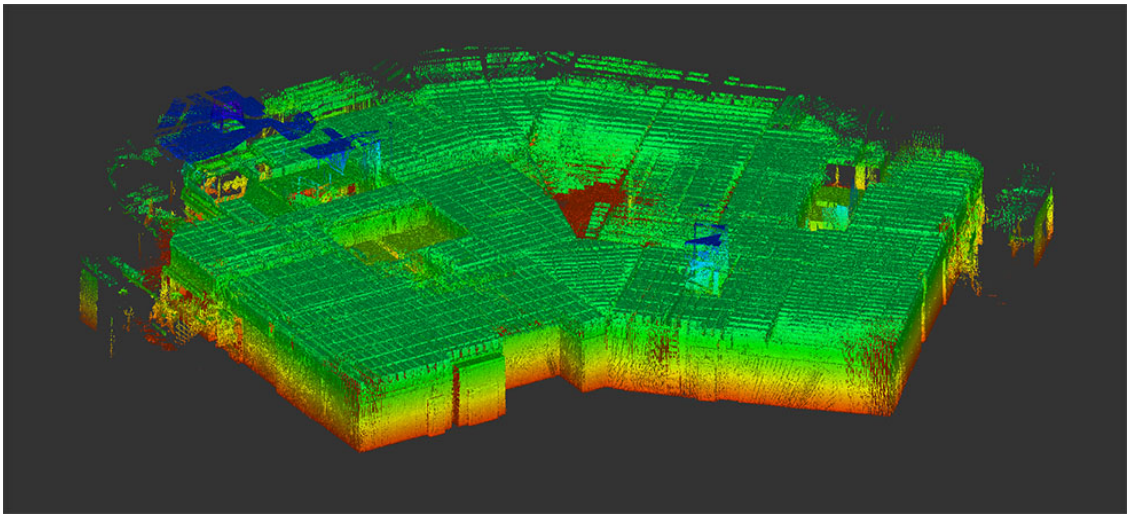


Figure 1.3: An example point cloud created using the methods presented in this thesis. Note that the points have been colored by height to increase the clarity of the screenshot.

using the methods described in this thesis. The points have been artificially colored by height to increase picture clarity.

The two algorithms in this thesis represent two approaches to the SLAM problem. Chapter 3 describes a 2.5D dimensional approach to mapping interior environments. By splitting the localization problem into a 2D localization problem followed by a height estimation problem, we create a series of 2D maps to represent the environment using an IMU and a horizontally mounted laser scanner. Following that, we use a downward facing laser scanner to estimate the height of the system relative to the ground. These two estimates are then fused together with barometer data to create a series of 2D occupancy grid maps via particle filtering. Finally, graph optimization is performed to create a high-rate optimized trajectory.

The presented 2.5D method however has some drawbacks. In making the 2.5D assumption we explicitly require that the environment be made up of mostly vertical planes. This assumption works for many types of buildings including office and academic buildings. Furthermore, performing online calibration of the sensors is challenging using a particle filter due to the curse of dimensionality. Intrinsic and extrinsic calibration parameters must be precomputed or inferred from CAD drawings. Lastly, sensor data is combined in an ad hoc manner. Data from camera sensors is not used in the localization process while IMU data is trusted

To address these concerns, in Chapter 4 we present a tightly-coupled EKF estimator that fuses data from all available sensors into a single optimized trajectory. By fusing data from both laser and camera sensors, we show that we are able to more accurately estimate the trajectory of the system than if we used only a single sensing modality. Secondly, by formulating the SLAM problem as an EKF estimator, data from individual sensors are causally fused together to estimate not only the position of the system and a low dimensional planar representation of the surrounding environment, but also an optimized estimate of the system's calibration parameters. Lastly, rather than explicitly requiring the environment be made of planar objects, the EKF algorithm detects areas that are planar and uses them as

feedback in the filter. In doing so, we are able to handle environments that are a mixture of planar and non-planar building elements without degrading the accuracy of the reconstructed trajectory. In the next sections, we detail the individual novel contributions presented in this thesis.

1.4.1 Automatic Barometric Floor Segmentation

The first novel contribution of this thesis is an automatic floor segmentation algorithm using barometric pressure data and Bayesian nonparametrics. Indoor floor detection via barometric pressure data has been studied before by Fallon et al. [7]. Using the pressure readings from a low-grade MEMS barometer, Fallon et al. was able to detect when an operator transitioned between floors of a building. However, in order to detect the number of floors in the building the authors used an incremental approach based on predefined heuristics.

In order to eliminate the need for predefined heuristics, we present a new methodology for automatic floor detection and segmentation based on Bayesian nonparametrics. By modeling the floor segmentation problem as a Chinese Restaurant Process [56, 57] (CRP) we are able to simultaneously partition the data while detecting the number of floors present in the dataset using a CRP mixture model. Furthermore, we show through experimentation that the CRP mixture model is extremely robust to choice of user defined parameters. The details of the algorithm are presented in Sec. 3.2.3 and the comparison results against both [7] and another unsupervised clustering algorithm [58] are presented in 3.5.1.

1.4.2 Fractional Genetic Scan Matcher

The second novel contribution of this thesis is an outlier-resistant, genetic scan matching algorithm that accurately matches scans despite a poor initial condition. The closest related previous work in genetic scan matching was presented by Lenac et al. [59]. In [59] the authors present a Hybrid Genetic Scan Matcher, which proposes to use a genetic search using a correlation based metric followed by a round of ICP to refine the solution.

We present two extensions to this algorithm. First, as our system's lasers often scan the ceiling, a large number of outliers can be present in the scan data and therefore a correlation based metric is not appropriate. To that end, we use the fractional root mean square distance metric [60] to provide robustness against outlier points. Secondly, we apply scan matching to force each chromosome into a local minima of the objective function. Under this formulation, the genetic scan matching algorithm efficiently searches only the local minima of the objective function. Although genetic scan matching can be a computationally expensive process, we present a solution space discretization method that significantly speeds up the algorithm without sacrificing accuracy. We call the extended algorithm the Fractional Genetic Scan Matcher (FGSM). The FGSM algorithm is detailed in Sec. 3.3.2 and the comparison results to [60] and [59] are presented in Sec. 3.5.3.

1.4.3 Loop Closure Validation Metrics

Next, in this thesis, we present two metrics to validate loop closure constraints for 2.5D laser-based SLAM systems. As graph optimizers are ill-equipped to handle outlier constraints, it is imperative that only high-quality loop closure constraints are used during pose-graph optimization. To that end we define two metrics based on the amount and complexity of overlapping geometry in order to vet the estimated loop closure constraints. Indoor environments contain many locations, such as long narrow hallways, where scan matching is ill-conditioned. By examining the quantity and complexity of overlapping geometry we automatically prevent erroneous loop closures from degrading the accuracy of the reconstructed trajectory.

The closest related work in this area was presented by Bosse and Zlot [47] which validates loop closure constraints based only on the correlation between matched 2D laser scans. In contrast to [47] which is focused on outdoor environments, our system is designed to work in indoors where scan matching is often ill-conditioned. We show that both a correlation and shape complexity metric are needed for reliable operation in indoor environments. Section 3.3.3 defines the metrics and provides visual examples to describe them. The validation metrics are tested using both automatic and manually defined loop closure constraints. The classification results are detailed in Sec. 3.5.5.

1.4.4 Online Laser Extrinsic and Temporal Calibration

To our best knowledge this thesis presents the first direct online extrinsic and temporal calibration between a 2D laser scanner and an inertial sensor that considers the rolling shutter effect of the laser scanner. By modeling the extrinsic calibration in our EKF estimator, we are able to obtain a variance reduction in both the rotational and translation components of the extrinsic calibration. Furthermore, by modeling the timestamping bias and rolling shutter aspects of the laser scanner, we are able to account for both intrascan distortion and latency in the data collection process.

Previous works on laser and inertial sensor calibration has been presented in a few contexts. Bosse et al. [5] presented a novel data collection system that placed a 2D laser scanner at the end of a passive spring. By exploiting the temporal offset between the IMU data and the motion of the spring, Bosse was able to calibrate the timestamping bias between the laser and IMU sensors. Rehder et al. [54] presented an optimization based framework for calibrating a 2D laser scanner against a VIO system. While this work achieves promising results, it makes the assumption that a visual inertial odometry system is available and ignores the rolling shutter nature of the laser scanner.

The closest method to our online extrinsic and temporal calibration algorithm was presented by Li et al. [52] but in the context of a VIO system. The authors incorporated a model for the timestamp bias and rolling shutter nature of a low-cost camera sensor into their EKF estimator. In contrast to [52], we instead apply the rolling shutter model to a 2D laser scanner and incorporate it into our online calibration procedure. We briefly describe the model used for modeling a rolling shutter laser scanner in Sec. 2.3, the calibration algorithm in Sec. 4.6, and present the results of the calibration procedure in Sec. 4.9.1.

1.4.5 Laser-Aided Inertial Navigation

The main novel contribution of the 3D localization algorithms of Chapter 4 is the laser-aided inertial navigation (L-INS). In this thesis we present a method for fusing IMU and any number of 2D laser scanners using a tightly coupled EKF estimator. Furthermore, the presented algorithm is not only capable of creating a planar representation of the environment with no a priori map but also can operate even when the environment is not Manhattan.

The contributed L-INS algorithm can be seen as extension of the previous works of Hesch et al. [61] and Zhao and Farrell [62, 63]. In [61], the authors presented a L-INS system mounted to a cane to assist the visually impaired. The authors assumed that the operator was traveling through an environment with only axis-aligned planes. Due to this constraint, the operator had to go through a calibration procedure to align the L-INS's internal coordinate system to the dominant directions of the building. Zhao and Farrell [62, 63] built upon this work and developed an outdoor system that utilized city maps and a 2D laser scanner for automotive localization in urban environments. While the authors relaxed the Manhattan assumption by allowing vertical planes of arbitrary orientation, they assumed that an a priori map was available to the L-INS system.

In this thesis we present an L-INS system that is capable of operating in non-Manhattan environments even when no a priori plane map is available. Furthermore, we naturally extend the algorithm to fuse data from any number of 2D laser scanners that are present on the system. In Sec. 4.6 we detail the L-INS data fusion algorithm and the results for the L-INS estimator can be found in Sec. 4.9.2.

1.4.6 Tightly Coupled Multi-Sensor Fusion On a Backpack System

The final contribution presented in this thesis is a modular multi-sensor tightly coupled EKF estimator for sensor fusion. By formulating the SLAM problem as an EKF estimator we are able to fuse sensor data from lasers, cameras, and IMU sensors in an extensible and modular fashion. As the sensor data asynchronously arrives into the system, an individual EKF update algorithm is run for each type of data. In doing so, we are able to handle any number of lasers, a camera, and an IMU seamlessly.

Multi-sensor systems have also been studied in previous works. In [7], Fallon et al. presented a ambulatory mobile mapping system that fused data from a 2D laser scanner, a barometer, and an IMU in a loosely coupled fashion. In [50], Bok et al. presented an outdoor system was used for historical preservation applications. Bok et al. fused camera, 2D lasers, and a GPS sensor using loose coupling and graph optimization.

In contrast to loosely coupled systems, tightly coupled estimators have also been utilized for multi-sensor systems. Wei et al. [64] presented an Information Filter approach to fusing data from laser based ICP, a GPS, and a stereoscopic camera system. The works that comes closest to our multi-sensor system are those of Lynen et al. [65] and Shen et al. [66]. Both Lynen et al. and Shen et al. presented a tightly coupled EKF estimator to fuse inertial, pressure, and monocular visual information for micro-aerial vehicles. Our approach differs from those previously presented because we not only utilize 2D laser data in the estimation process, but maintain a compact planar representation of the environment for applying loop closure within the EKF estimator.

Chapter 2

Preliminaries

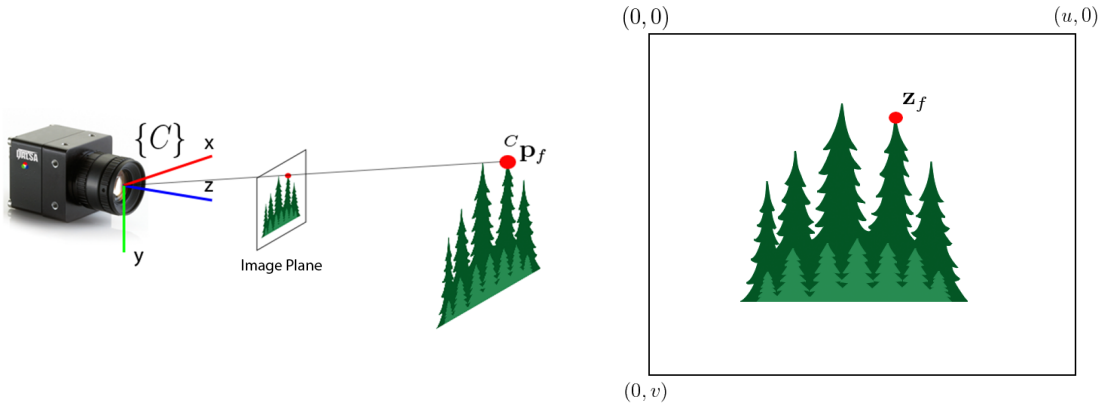
2.1 Notation

The following notation is used throughout this thesis.

Symbol Notation	**Example**
Scalars are denoted in lower case	x
Vectors are denoted in bold lower case	$\mathbf{x}$
Matrices are denoted in bold upper case	$\mathbf{X}$
Cross product of a vector uses $\lfloor \cdot \times \rfloor$	$\lfloor \boldsymbol{\omega} \times \rfloor$
Quaternion product is denoted as $\otimes$	$\mathbf{q}_1 \otimes \mathbf{q}_2$
Time derivative is represented by a dot	$\dot{\mathbf{p}} = \mathbf{v}$
Jacobian of function $f(\cdot)$ by vector $\mathbf{x}$	$\mathbf{J}_\mathbf{x}\big(f(\cdot)\big)$ or $\mathbf{H}_\mathbf{x}$ or $\boldsymbol{\Gamma}_\mathbf{x}$

Estimation Notation	**Example**
An estimated value is denoted by a hat	$\hat{\mathbf{x}}$
A tilde denotes an estimation error	$\tilde{\mathbf{x}}$
Rotation error is denoted using $\tilde{\boldsymbol{\theta}}$ or δ	$\tilde{\boldsymbol{\theta}}^G$ or $\delta\mathbf{q}$

Coordinate Notation	**Example**
Coordinate frames are denoted in upper case	G or $\{G\}$
A rotation from $\{G\}$ to $\{B\}$	${}^B_G\mathbf{R}$ or ${}^B\mathbf{q}_G$
The coordinate frame of a point $\mathbf{p}_B$ in $\{G\}$	${}^G\mathbf{p}_B$



(a) (b)

Figure 2.1: An illustration of a point being projected into a camera image. (a): A camera views a point ${}^C\mathbf{p}_f$ in its coordinate frame $\{C\}$. (b): The resulting location in pixel coordinates $\mathbf{z}_f$ is predicted using the rectilinear projection model.

2.2 Camera Sensor Modeling

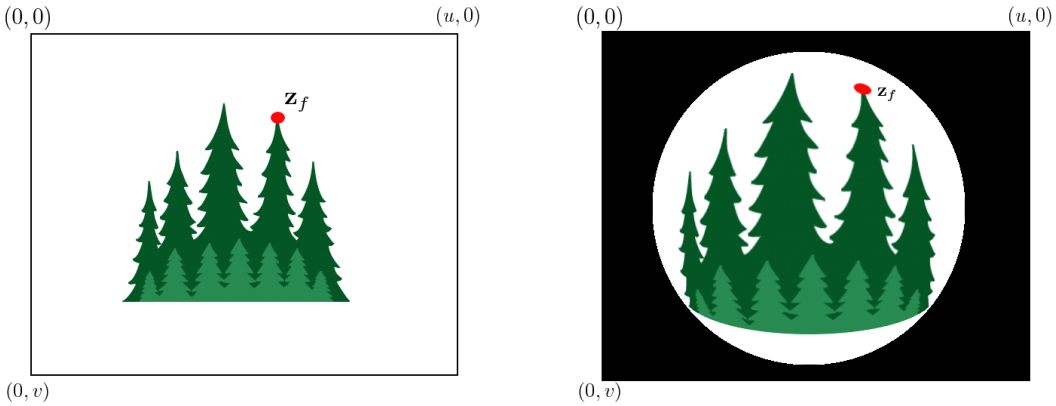
As imaging sensors become cheaper and lighter, the demand for algorithms that take advantage of high-quality low cost imaging sensors has increased. Coupled with the wide availability of open source tools and a mature literature on sensor modeling, applications in the localization and mapping communities that take advantage of imaging sensors have become ubiquitous. In this section, we will detail the camera sensor models used in this work.

2.2.1 Camera Sensor Models

$$\mathbf{z}_f = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{{}^C Z} \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} {}^C\mathbf{p}_f \tag{2.1}$$

Numerous camera models have been proposed to describe the relationship between 3D points ${}^C\mathbf{p}_f = [{}^C X, {}^C Y, {}^C Z]^T$ viewed by a camera and their pixel locations in the resulting image $\mathbf{z}_f$. Depending on the type of lens system used by the camera, either a rectilinear or fisheye model is appropriate. For small field of view lens systems, the rectilinear camera model has been shown to accurately model image formation [48]. In Eq. 2.1 the quantities $\mathbf{f} = [f_x, f_y]^T$ and $\mathbf{o} = [o_x, o_y]^T$ represent the focal length and center pixel of the camera. Figure 2.1 shows an illustration of the camera projection process. Shown in Fig. 2.1 a point ${}^C\mathbf{p}_f$ is viewed by a rectilinear camera. The location of the 3D point in pixel coordinates $\mathbf{z}_f$ is predicted using the rectilinear projection model of Eq. 2.1.

Unfortunately, most camera lens systems are built from both rectilinear and aspherical elements in order to increase the field of view and reduce color scattering as light travels



(a) (b)

Figure 2.2: An illustration demonstrating omnidirectional camera image distortion applied to the tree scene from Figure 2.1. (a): The scene under rectilinear projection. (b): The scene under omnidirectional projective distortion.

though the lens. These types of lens systems cause distortions in the resulting image and must be modeled appropriately. To this end, the distortion of the lens is typically modeled to account for such effects using both by a radial component $\mathbf{k} = [k_1, k_2, k_3]^T$ and tangential component $\mathbf{t} = [t_1, t_2]^T$ [48, 67].

$$\begin{aligned} \mathbf{z}_f &= \mathbf{h}({}^C\mathbf{p}_f) \\ &= \begin{bmatrix} o_x \\ o_y \end{bmatrix} + \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \left((1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} 2uvt_1 + (r^2 + 2u^2)t_2 \\ (r^2 + 2v^2)t_1 + 2uvt_2 \end{bmatrix} \right) \end{aligned} \tag{2.2}$$

Here the 3D feature point is expressed as ${}^C\mathbf{p}_f$ and the intermediary values u, v, and r are defined as $u = X/Z$, $v = Y/Z$, and $r^2 = u^2 + v^2$. The expanded model of 2.2 is generally calibrated offline using a collection of images of a planar checkerboard pattern and any one of many open source software packages [68, 69].

In contrast to rectilinear lens systems, some cameras utilize omnidirectional lenses or ball mirrors to increase the field of view. By expanding the field of view of the imaging sensor, more distortion is introduced into the resulting image. Under this type of projection, straight lines viewed by the camera get warped to be circular in the resulting image. Figure 2.2 shows what the forest scene from Fig. 2.1(a) under rectilinear and omnidirectional projection models. Notice how in the omnidirectional image of Fig. 2.1(b) lines and edges have become distorted.

Numerous calibration methods have been proposed to model these lens systems [70, 71]. For this work we model the fisheye omnidirectional lenses on the ambulatory backpack system using the model described by Scaramuzza et al. in [71].

(a) (b)

Figure 2.3: An example of applying image undistortion to a fisheye image. (a): The input distorted fisheye image. (b): An undistorted version of a subsection of the input fisheye image.

$$\mathbf{z}_f = \begin{bmatrix} o_x \\ o_y \end{bmatrix} + \frac{\rho(\theta)}{R} \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{2.3a}$$

$$R = \sqrt{{}^{C}X^2 + {}^{C}Y^2} \tag{2.3b}$$

$$\theta = \text{atan}({}^{C}Z/R) \tag{2.3c}$$

$$\rho(\theta) = k_0 + k_1\theta + k_2\theta^2 + k_3\theta^3 \tag{2.3d}$$

In the previous equations we can see that the distortion is now a function of the distance to the camera via Eq. 2.3b and the radius away from the optical axis via Eq. 2.3c. The severity of the distortion is modeled by the polynomial $\rho(\theta)$ and distortion coefficients $\mathbf{k} = [k_1, k_2, k_3, k_4]^T$. This matches the intuition that omnidirectional lenses cause spherical shaped distortion around the edges of the image. The omnidirectional projection model of Eq. 2.3 is typically calibrated offline using open source software [72].

2.2.2 Image Undistortion

The backpack systems described in Section 2.4 contain camera sensors equipped with fisheye omnidirectional lenses. Fisheye lenses offer a good balance between wide field of view and spherical distortion. For some applications, such as feature tracking or 3D reconstruction, a wide field of view is beneficial. Unfortunately, many computer vision, graphics, and rendering algorithms assume that the input images are subjected to rectilinear distortion. To overcome this limitation, we preprocess the fisheye images to undistort them and remove any spherical distortion caused by the omnidirectional lens. Given a calibrated omnidirectional camera model and a target rectilinear camera model, the undistortion process warps an input image from one projection model to the other [73].

Figure 2.3 shows an example of the undistortion process. Fig. 2.3(a) shows a fisheye image captured from one of the backpack systems. A $70^\circ \times 70^\circ$ section of the center of the image

was selected for undistortion. The resulting rectilinear image is shown in Fig. 2.3(b). Notice how some vertical and horizontal building elements in the fisheye image have been curved due to omnidirectional lens distortion. The same building elements have been straightened out in the resulting rectilinear image.

2.3 Laser Sensor Modeling

Laser scanning technology has quickly begun to dominate the fields of historical preservation, construction monitoring, and architectural modeling due to their portability and high accuracy. As such, numerous companies have produced high-quality laser scanning equipment based on LiDAR technology. In this section we briefly describe how we model these types of sensors.

2.3.1 Laser Sensor Model

LiDAR sensors generally operate by emitting light pulses and then timing how long it takes for a return to be reflected back at the sensor. Since the speed of light is a known constant the reflection time can be used to compute the distance from the laser scanner to an object in the environment. By combining many range readings, a laser scanner provides a metrically accurate representation of the surrounding environment.

Laser scanners come in both 2D and 3D varieties. Termed "profilers", 2D laser scanners typically have a single laser and a single mirror that rotates around a central axis. By spinning the central mirror relative to the laser scanner, a 2D profile of the surrounding objects is created. 3D laser scanners are either built from an array of profilers or by rotating the entire 2D laser assembly.

The two ambulatory backpack systems shown in Section 2.4 contain orthogonally mounted Hokuyo UTM-30LX 2D laser scanners [74]. The UTM-30LX is a mid-range infrared LiDAR scanner intended for mobile robotics applications. Like other profilers, this laser scanner consists of a single laser beam and rotating central mirror system. Each laser scan from the UTM-30LX consists of only a list of range measurements. Assuming a bearing for each range measurement, the 2D profile is reconstructed.

$$ {}^{L}\mathbf{p}_i = r_i \begin{bmatrix} \cos(\theta_i) \\ \sin(\theta_i) \\ 0 \end{bmatrix} \tag{2.4} $$

The range measured by the scanner r_i is converted into a 3D point in the laser frame of reference using the assumed bearing angle of the measurement θ_i. The entire collection of range readings are assigned a single timestamp t. Although the noise in the range measurement has been shown to be both a function of the color, reflectivity, and distance to target [75,76], it is typically modeled as being corrupted by white Gaussian noise with standard deviation around 2 cm.

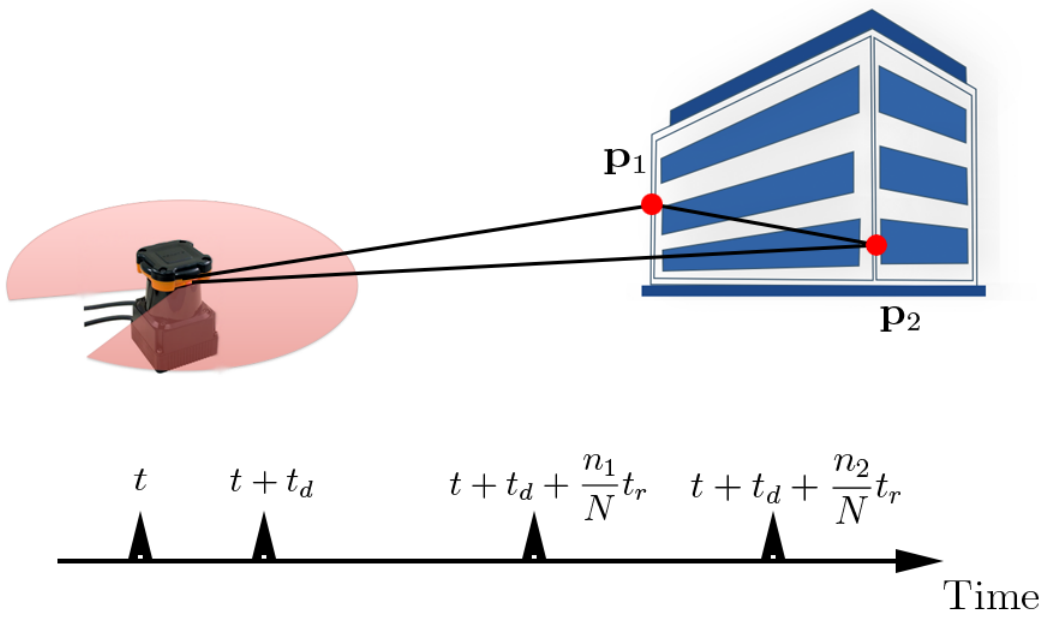


Figure 2.4: An illustration depicting the rolling shutter nature of a 2D profiling laser scanner. The individual points $\mathbf{p}_1$ and $\mathbf{p}_2$ are captured at different times from a non-level scanner. Each point's timestamp is offset by some fixed delay t_d plus a delay that increases as the index of the point increases kt_r/N.

2.3.2 Rolling Shutter Model

While the simple laser model from the previous section works for low-speed applications, the laser readings become distorted when the laser scanner is subjected to high dynamic motion. The 2D profiling scanners do not measure the entire profile simultaneously. As the central mirror spins around the LiDAR sensor samples the points one at a time in a linear fashion. If the platform's dynamics are significantly faster than the frame rate of the laser scanner, the resulting data will become warped.

We model this phenomena using a rolling shutter model for the laser scanner. Although the UTM-30LX scanner assigns a single timestamp to the entire laser scan, we model the timestamp of each individual point.

$$t_n = t + t_d + \frac{n}{N} t_r \tag{2.5}$$

In the above expression, the true capture time of the nth point t_n consists of the base timestamp t , a constant bias t_d that models any system latency, and a fraction of the sensors readout time t_r. If the position of the laser scanner is also known, the range data can be undistorted when it is transformed into a common coordinate system.

Figure 2.4 illustrates the rolling shutter nature of a 2D profiling laser scanner. As the non-level laser scanner scans the building it takes individual points in a sweeping fashion in

a 2D scan plane. The timestamp of each laser point linearly increases as the scan moves from first point to the Nth point in the scan. As shown, a point $\mathbf{p}_i$ with index n_i has a timestamp that is calculated as $t + t_d + t_r n_i / N$.

2.4 Hardware Systems

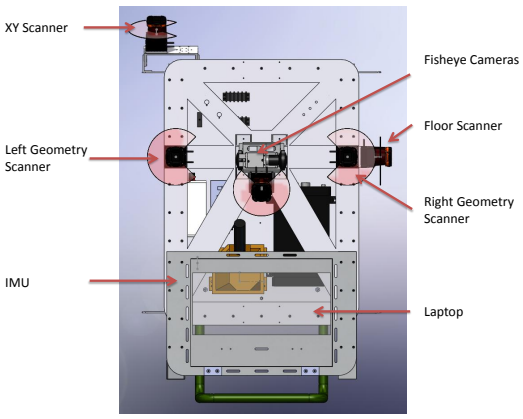
The Video and Image Processing Lab at University of California has developed a custom ambulatory backpack system for indoor mapping, navigation, and geometric modeling. In contrast to static scanning and wheeled robotic platforms, backpack-like systems strike an important balance between speed of acquisition and system mobility.

Figure 2.5 shows the two revisions of the backpack system used for testing the algorithms of this thesis. Figure 2.5(a) shows a CAD model of the first generation prototype system used for testing the 2.5D algorithms presented in Chapter 3. Figures 2.5(b) and 2.5(c) contain a CAD drawing and a photo of the weight-reduced second generation backpack system used for testing the multi-story particle filtering algorithm of Section 3.2.3 and the 3D localization algorithms presented in 4.

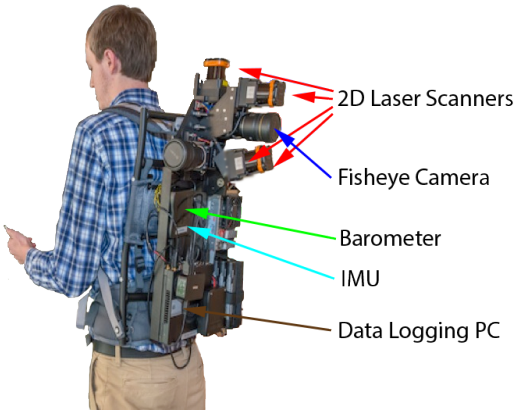
Both backpack systems are equipped with a number of 2D Hokuyo UTM-30LX laser scanners. These laser scanners are capable of measuring the distances up to 30 meters in a 270° arc around the scanner and operate at a rate of up to 40Hz. The laser scanners are configured so that they scan orthogonal scanning planes. A horizontally mounted laser scanner is primarily responsible for measuring the velocity in the global xy plane and heading. Another 2D scanner is mounted so that it scans the floors and ceilings behind the operator in order to measure the distance to floor and pitch angles. Lastly, the remaining scanners are positioned so that they measure the distance perpendicular to the operators direction of motion and are responsible for collecting dense geometric information.

In addition to laser scanners, both systems contain fisheye cameras for capturing imagery as the operator traverses the environment. The first generation prototype shown in Fig 2.5(a) contains dual Grasshopper 5MP cameras with Nikon 180° fisheye lenses facing opposite directions to provide a full sphere view of the environment. The updated second generation system contains a backward facing and two sideways facing Dalsa Teledyne Genie cameras equipped with Nikkor fisheye lenses that provide a 140° diagonal field of view. The backwards facing Genie camera captures 12MP images using a global shutter at a rate of 7Hz and is used for data fusion via visual inertial odometry. The other two cameras capture 12MP images at 1Hz using a global shutter and are used for colorizing the generated point clouds.

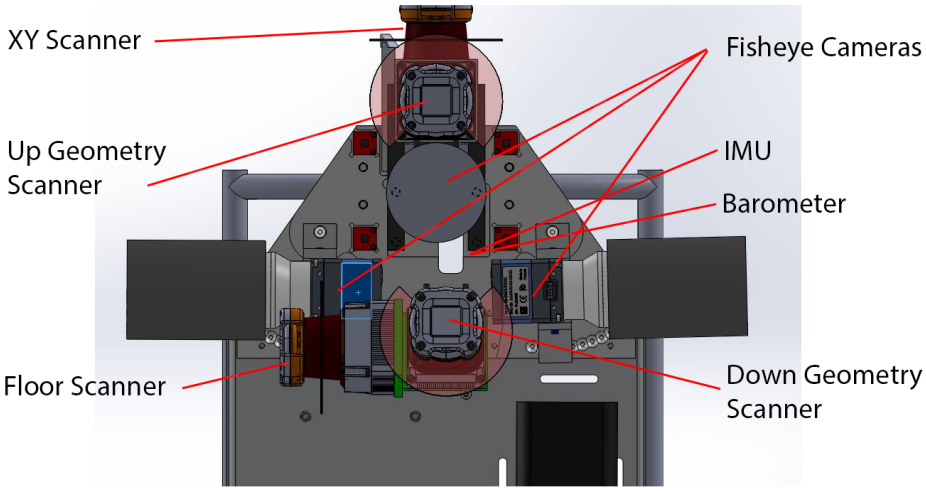
InterSense IntertiaCube IMUs are also present on both system to provide high rate orientation, gyroscope, and accelerometer data at 200Hz. This data is utilized to provide accurate estimation of the gravity direction for the system. Furthermore, the second generation system has been outfitted with a Bosch SensorTech BMP085 MEMS barometer to record temperature and pressure data at a rate of 8Hz for detecting floor transitions in the data.



(a) Generation 1 System - CAD Drawing



(b) Generation 2 System - Picture



(c) Generation 2 System - CAD Drawing

Figure 2.5: Images of the hardware systems used for testing the algorithms contained in this thesis. Two versions of the ambulatory backpack system have been constructed. (a): A CAD drawing of the first generation prototype backpack system. (b): A labeled photo of the second generation backpack system. (c): A CAD drawing of the second generation, weight-reduced backpack system.

Chapter 3

2.5D Localization Algorithms

3.1 Algorithm Overview

In order to address the shortcomings of existing work on 2.5D mobile mapping [77–79], we present an algorithmic shift from our previous efforts. Previously we relied on appearance based loop closure detection algorithms to reduce accumulated errors in the dead reckoning trajectory. Instead, we first use a Rao-Blackwellized particle filter to obtain a coarse grid map of the environment so that we can robustly detect loop closures. The detected loop closures are then fused with the dead reckoning trajectory to alleviate the spatial and temporal quantization effects inherent to grid mapping.

First, rather than relying on loop closure constraints detected from optical imagery, we derive them from an occupancy grid map created via Rao-Blackwellized particle filtering. Previous methods for loop closure detection, such as those relying on keypoints [80], features learned from a machine learning framework [81], or correlative map matching [47] all aim to discover loop closures constraints between arbitrary locations in the environment. Although these methods have successfully been applied to indoor environments, they often result in erroneous detections when repeated patterns appear in the environment. In indoor environments repeated structures exist in both the optical imagery, such as a repeated wallpaper pattern, or in the laser data. Furthermore, because such methods do not incorporate prior geometric information, they are unable to prune false detections using line-of-sight or other geometric constraints. To address these issues, we propose to utilize the information contained in the occupancy grid map as a geometric prior and only detect loop closures in locations that are suitable for scan matching. By detecting loop closures based on a grid map representation of the environment, we leverage the information contained in the occupancy grid map and ensure that constraints are detected only in locations that are well conditioned for scan matching.

Next, we extend the 2D occupancy grid map based Rao-Blackwellized particle filtering algorithm across multiple floors of a building. By using a single 2D occupancy grid map, the Rao-Blackwellized particle filter inherently makes the assumption that the world is well represented by a single 2D projection of the environment. Unfortunately, many indoor environments, such as multi-story buildings, violate the single 2D projection assumption and thus the particle filter must be extended to work in these environments. We overcome

this limitation by extending the particle filter to use a collection of 2D maps to represent the individual floors of a building. To this end, we present a novel barometer based floor segmentation technique that automatically detects the number of floors and segments the path accordingly using a variant on the Chinese Restaurant Process [56, 57].

Our third contribution is our proposed outlier-resistant, genetic scan matching algorithm that accurately matches scans despite a poor initial condition. Previous genetic scan matching algorithms, such as the Hybrid Genetic Scan Matcher [59], use a genetic search using a correlation based metric followed by a round of Iterated Closest Point (ICP) [82] to refine the solution. We propose two extensions to this algorithm. First, as our system's lasers often scan the ceiling, a large number of outliers can be present in the scan data and therefore a correlation based metric is not appropriate. To that end, we use the fractional root mean square distance metric [60] to provide robustness against outlier points. Secondly, we apply scan matching to force each chromosome into a local minima of the objective function. Under this formulation, the genetic scan matching algorithm efficiently searches only the local minima of the objective function.

Lastly, we present two metrics based on the amount and complexity of overlapping geometry in order to vet the estimated loop closure constraints. Indoor environments contain many locations, such as long narrow hallways, where scan matching is ill-conditioned. By examining the quantity and complexity of overlapping geometry we automatically prevent erroneous loop closures from degrading the accuracy of the reconstructed trajectory.

Figure 3.1 shows a block diagram of our proposed off-line algorithmic pipeline for 2.5D localization and mapping. Unlike wheeled systems, an ambulatory system is unable to utilize wheel encoders for dead reckoning and thus we first use the XY scanner and IMU to generate odometry measurements. Assuming that the environment is composed of vertically oriented planes, we are able to undistort scan distortion that arises from the pitch and roll motion of the operators natural gait by using the 2.5D assumption. Furthermore, due to the ambulatory nature of the system, outlier points are often detected when the laser scans the ceiling or ground planes. We overcome this problem by applying a scan matching algorithm which explicitly models the presence of outliers. By segmenting the scan into a set of inlier points, we automatically detect points which violate the vertical wall assumption without a priori knowledge of the outlier distribution. The scan matching odometry results are then concatenated to form a dead reckoning trajectory.

Integrating odometry for dead reckoning results in accumulated error in the reconstructed trajectory known as drift. In order to correct for accumulated drift, we fuse the odometry readings and laser data into a single geometrically consistent representation using a particle filtering approach. Particle filtering is performed using two passes of a Rao-Blackwellized particle filter (RBPF). We first use the dead reckoning trajectory to aggregate temporally adjacent laser readings into local submaps by running a small particle filter around small subsets of the trajectory. Since the XY scanner is subjected to pitch and roll motion, it contains many points, such as ceiling or ground strikes, which do not fit the vertical wall assumption. These points must be eliminated before any 2D SLAM approach can be effectively utilized. Then we fuse the submaps into a single topographically correct, occupancy grid map by applying another round of RBPF using the created submaps.

In contrast to [79], we detect loop closure constraints using the occupancy grid map representation of the environment. Unlike our previous approaches [77–79], we utilize a

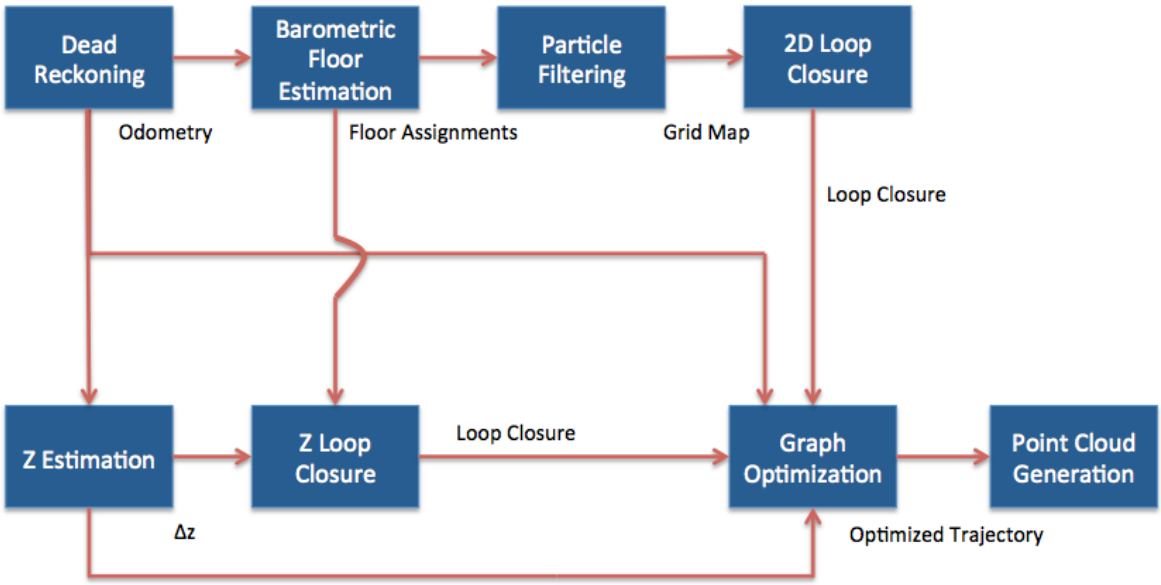


Figure 3.1: Block diagram of the algorithms used for localizing an ambulatory backpack system.

geometrically consistent occupancy grid map to construct loop closures only in regions that are inherently well conditioned for scan matching. Since the particle filter also computes a rough estimation of the position for each submap, we are able to derive an initial condition for the loop closure constraints and compute the metric transformation between locations by applying a genetic scan matching algorithm. We then vet the detected loop closure constraints using a combination of metrics that quantify the amount and complexity of overlapping geometry. This allows us to both detect and vet loop closure constraints without any tedious manual intervention.

The 2D mapping results are extended to a full 6DOF pose by combining the pitch and roll from the IMU with a height estimate at each location. We use the adaptive height estimator of Kua et al. [79] directly to obtain an estimate of the height difference between adjacent poses Δz. We concatenate the 2D location and height deltas Δz to obtain the 3D position and combine pitch, roll, and heading estimates to recover the orientation. Once we have an initial estimate of the 3D trajectory of the system, we find additional loop closures to correct for any accumulated drift in the height estimates. We split the 2D trajectory according to the recovered floor partitions and place loop closure constraints anywhere the 2D trajectory intersects itself and is detected to be on flat ground.

Finally, we fuse the 2D dead reckoning trajectory, height information, pitch and roll data, and verified loop closure constraints via graph optimization. Specifically, we form an edge directed graph using the odometry to create pairwise connections between temporally adjacent nodes. We then insert the vetted loop closure constraints between the detected loop closure locations. We apply a graph optimization procedure such as TORO [36], g2o [34], or SAM [35] to generate a single 2D optimized trajectory. The optimized 6DOF path is then combined with the cameras' and geometric scanners' data to produce a dense, colorized

3D point cloud.

The rest of the chapter is organized as follows: First, Section 3.2 describes algorithms used in computing the 3D trajectory of the system. Then, Section 3.3 presents the proposed methodology for detecting and validating loop closure constraints from 2D occupancy grid maps. Next, Section 3.4 describes how the optimized 2D trajectory is extended to a full 6DOF trajectory. Finally, experimental results for the proposed algorithms are included in Section 3.5.

3.2 2.5D Particle Filtering Localization

3.2.1 2D Dead Reckoning

This section provides a detailed description of the 2.5D off-line algorithmic pipeline shown in Figure 3.1 which is used for localizing the human-mounted, ambulatory backpack system. Unlike wheeled systems where wheel encoders provide dead reckoning, human-mounted mobile mapping systems have to derive odometry from other sources such as scan matching or IMU measurements [7, 61, 77, 83]. We utilize scan matching to align temporally adjacent sensor readings from the XY scanner to estimate incremental motion. The incremental motion is then concatenated to produce a dead reckoning trajectory. Since the system is carried by a human operator, the XY scanner is not always perfectly level. Specifically, pitch and roll introduced by the operator's gait causes significant warping of the sensor readings and direct scan matching leads to large errors in the reconstructed trajectory.

Assuming that the walls are perfectly vertical, we project the scans along the direction of gravity to undo the warping in the XY scanner's data. By projecting the scan points along the direction of the gravity vector, we correct for the warping introduced by non-zero pitch and roll. Figure 3.2 shows an an example of this procedure. Figure 3.2(a) shows a simulated scan inside a box-like environment. The raw sensor readings, shown in Figure 3.2(b), contain warping caused by the non-zero pitch and roll resulting in the angle between lines to be less than 90°. Figure 3.2(c) shows the result of scan projection which results in the lines' angles of intersection to be 90°.

Assuming that the environment remains static between scans, we use successive readings from the lasers to estimate the incremental motion between scans. Many approaches have been suggested to solve the scan matching problem including global approaches [84] and iterative approaches [47, 85, 86]. Of particular interest is a class of iterative algorithms known as the Iterative Closest Point algorithms, or ICP [82].

The classical ICP problem is framed in the following manner. Given two dimensional points sets, P and Q, ICP iteratively attempts to find transformation $T(\cdot, \mu)$ that minimizes the following objective function:

$$e = \sum_{i=0}^{|Q|-1} \|\mathbf{p}_i - T(\mathbf{q}_i, \mu)\|^2 \tag{3.1}$$

where $\mathbf{p}_i \in P$ and $\mathbf{q}_i \in Q$ are matched elements, $|Q|$ is the number of elements in set Q, and $T(\cdot, \mu)$ is the transformation operator that rotates and translates point $\mathbf{q}_i$ into the reference

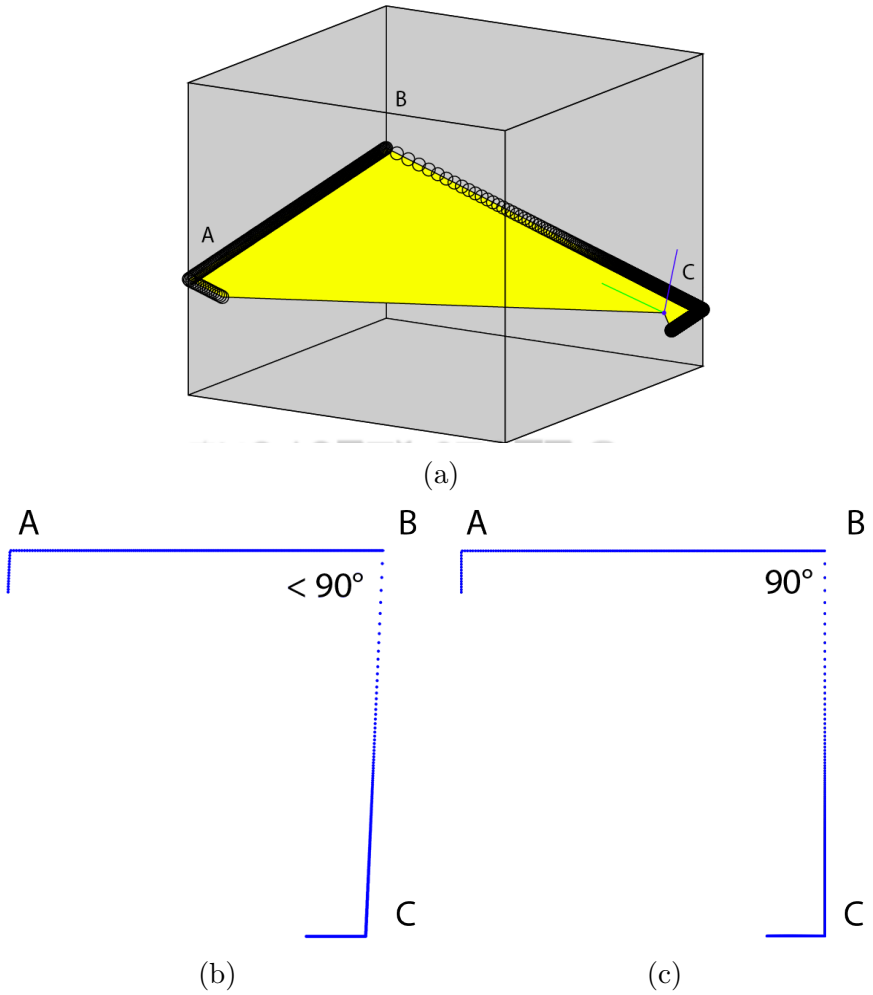


Figure 3.2: An example of scan projection. (a): An off-axis scanner in a cubic environment. (b): The raw readings of the laser scanner. Note that the angle ABC is less than 90° due to scan warping. (c): After projection the angle ABC has been corrected to 90°.

frame of P using transformation parameters μ. Since our system is equipped with 2D lasers, we are concerned only with the specific case of 2D point matching.

Variants of the metric of Eq. 3.1 have been suggested for improving accuracy and robustness to outliers. The point-to-line metric of [85] uses point-to-surface matching to reduce inaccuracies caused by the lasers sampling the surfaces at different locations. Additionally, because an ambulatory backpack system undergoes significant roll and pitch motion due to the operators natural gait, a large number of outliers may be present from dynamic objects or ceiling strikes. Proposed originally for arbitrary point cloud matching, the fractional iterative closest point (FICP) algorithm introduces a metric known as the fractional root mean square distance (FRMSD). The optimal transform parameters $T(\cdot, \mu)$ and inlier set $D_f \subseteq Q$ are obtained by iterating the following steps [60]:

- Given an initial transform $T(\cdot, \mu_0)$, points in Q are matched to their nearest neighbors in set P.

- Assuming a given $T(\cdot, \mu_0)$, an optimal set of inlier points D_f is identified.
- Using inlier set D_f, the transform parameters μ are recovered using a first-order Taylor expansion and solving for the optimal linear estimate of μ [47].

Although the vertical wall assumption corrects for the natural gait of the human operator, scan points that originate from the ceiling, floor, or dynamic objects in the environment are not well modeled by a vertical plane and must be handled separately. To this end, we use the point-to-line and fractional metrics in the following objective function:

$$e = \frac{1}{f^{\lambda}} \sqrt{\frac{1}{|D_f|} \sum_{\mathbf{q}_i \in D_f} \|\mathbf{n}_i^T (\mathbf{p}_i - T(\mathbf{q}_i, \mu))\|^2} \tag{3.2}$$

where f is the fraction of points that are considered inliers, D_f is the set of inlier points, $\mathbf{n}_i$ is the normal vector of the surface at point $\mathbf{p}_i$, and λ is a free parameter that controls how aggressively points are labeled inliers. The objective function is then minimized using the FICP framework [60].

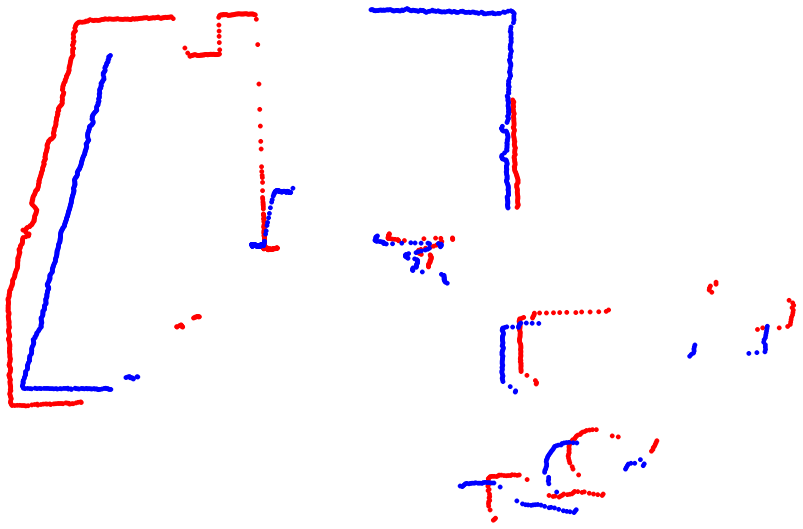
By solving for both the set of inliers and the optimal transformation parameters, the metric in Equation (3.2) identifies outliers in the data without any prior knowledge of their distribution. The process of iteratively segmenting geometry and recovering the transformation allows for accurate recovery of incremental motion even in the presence of a large number of outlier points.

Fig. 3.3 shows an example of the above scan matching algorithm. Fig. 3.3(a) depicts an example pair of LiDAR scans, shown in red and blue, aligned using an initial estimate of the transformation based upon a priori information. Fig. 3.3(b) depicts the alignment of scan pair from Fig. 3.3(a) after the FICP algorithm has been performed. The portion of the geometry that has been found to be part of the inlier set D_f is shown in green. By segmenting the inlier points, we detect and ignore points that do not meet the vertical wall assumption.

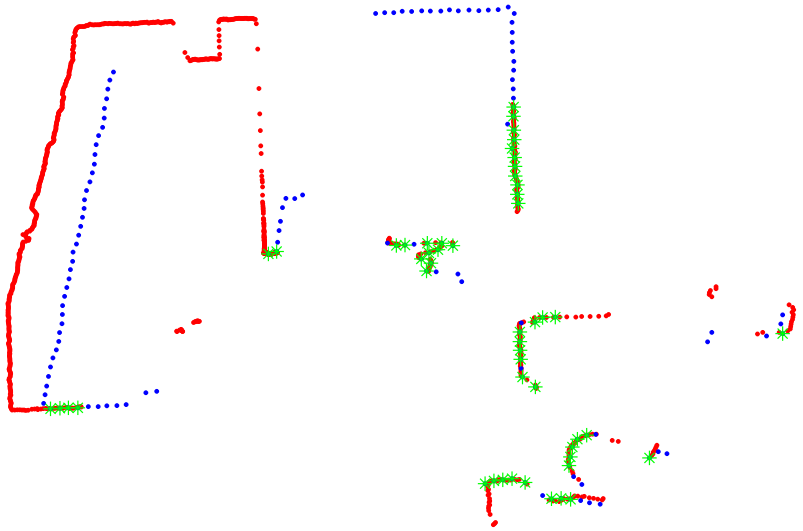
We take the incremental changes in position and integrate them to recover the dead reckoning trajectory. Since the path is built recursively, any errors in the transformations are compounded and lead to drift in the reconstructed trajectory. Inaccuracies in the recovered path lead to a geometrically inconsistent map and thus are not suitable for most mapping applications. In order to overcome the accumulated drift in the dead reckoning trajectory, we reformulate the problem to obtain a solution that optimizes both the path and the environment simultaneously. Classical solutions such as those involving Kalman filters, particle filters, and graph based approaches are discussed in [87].

3.2.2 Submap Generation

Since our system is mounted on a human operator, the sensor readings can contain a large number of outlier points due to clutter, ceiling strikes, or dynamic objects in the environment. In order to apply classical 2D SLAM algorithms, we must first eliminate the outlier points that result from the roll and pitch motion of the system. We apply a two-pass particle filtering algorithm in order to first aggregate temporally adjacent scanner readings into local submaps to eliminate outlier points and enhance the sensor's field of view before generating a



(a)



(b)

Figure 3.3: An example of performing the FICP algorithm. (a): The individual scans, depicted in red and blue, aligned using the initial estimate of the transformation. (b) The scan pairs after the FICP algorithm has been performed. The portion of geometry declared as the inlier points are denoted in green.

single geometrically consistent occupancy grid map representation of the environment. This section discusses the submap generation procedure.

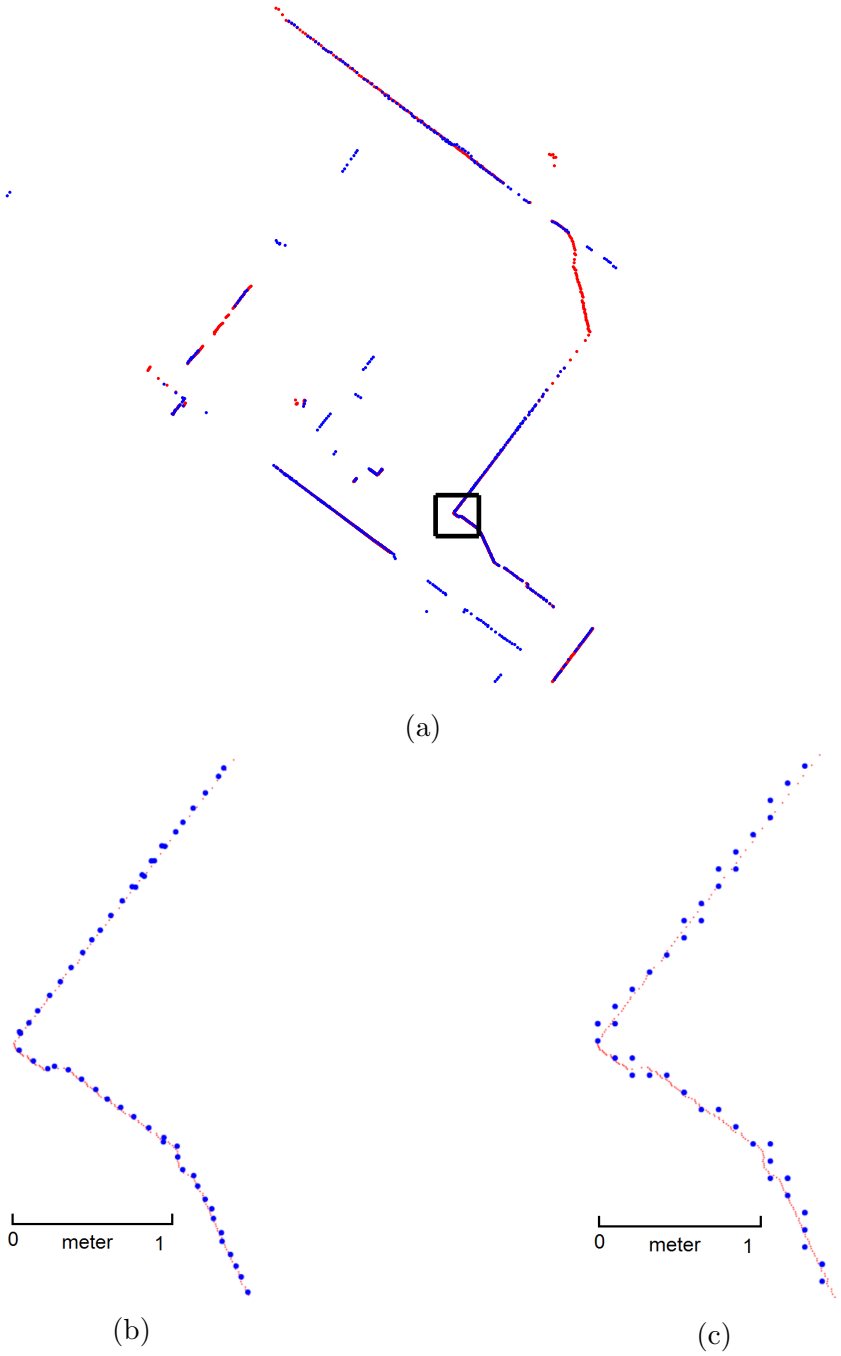
In order to fuse sequential scans into a geometrically consistent local submap, the first pass of particle filtering only merges scans from a small, temporally close, region of the environment. Although typically formalized for large scale mapping, we utilize the RBPF to generate accurate maps of subsections of the environment [88]. The approach we take here follows the classical grid mapping framework with three distinctions. First, we use the dead reckoning results of Section 3.2.1 as the odometry and consider only the inlier points of the scan matching result when merging points into the map or computing the weighting factor. By considering only the inlier points, we ignore any outlier points that originated from dynamic objects in the environment. Secondly, rather than using a strict discretization to create the grid map, we compute the average position of all points that lie in a grid cell. By using the average position as the representative sample for each grid, quantization errors can be mitigated. Lastly, we only consider the temporally closest observations when creating each submap as we are only interested in creating a map for a small subsection of the environment.

Figure 3.4 show typical results of RBPF based submapping. The red points represent the sensor's original readings while the blue points represent geometry that is added by the submapping procedure. By aggregating temporally adjacent scans, the amount of visible geometry for scan matching has been substantially increased and ceiling points have been removed. Additionally, as seen in Fig. 3.4(b), tracking the average point of each grid cell mitigates quantization effects that result from the grid mapping approach. The original corner, denoted by the black square in Fig. 3.4(a), matches very closely to the corner built during submap construction. For comparison, the same corner generated with a strict discretization is shown in Fig. 3.4(c).

The number of temporally adjacent scans used in submap construction impacts both the construction time and the amount of geometry in the reconstructed RBPF based submapping algorithm. To limit computation time, we choose local submap sizes based on the following heuristic. Given a location of interest, we collect scans from neighboring locations until either the estimated cumulative translation has exceeded N meters or the estimated rotation has exceeded θ degrees. We have empirically determined values of $N = 2$ and $\theta = 30^\circ$ to work well for our experiments.

3.2.3 Automatic Floor Partitioning

Before we can apply occupancy grid map based particle filtering, we must segment the data based on which floor of the building it originated from. Previous works, such as [8, 10, 32], have introduced mobile mapping systems that produce a high-quality 2D and 3D maps of an environment using either an ambulatory or unmanned aerial vehicle (UAV). Each of these systems track the position of the system using a combination of cameras and 2D laser scanners. Using a single 2D laser, these systems create a 2D map of the environment. However, to enable the system to map more than a single floor of the interior of a building, the laser must be augmented with additional information to detect transitions between floors. The method of [10] uses a small mirror to redirect part of the laser's field of view downward to make a direct measurement of the system's height above ground. This can then be tracked



(a)

(b)

(c)

Figure 3.4: A typical result of RBPF based submapping. The original sensor's readings are shown in red while the resulting submap is shown in blue. (a): The amount of visible geometry has been expanded beyond the original sensor's readings and ceiling points have been removed. (b): A close up of the section of the submap denoted by the black square shown in (a) shows how spatial averaging mitigates some quantization errors. (c): The same corner using a strict discretization.

as a function of time to detect floor transitions in the environment. The system presented in [8] uses an additional 2D laser pointed downward to track the vertical velocity of the system which is then integrated to recover the height.

Barometric pressure data has also been used for floor partitioning on mobile mapping platforms. Low-grade barometers such as the Bosch BMP085 are capable of providing pressure and altitude readings to approximately 1 meter RMS accuracy. Fallon et al. presented in [7] an algorithm to group pressure readings by floor from an on-board digital barometer. Using a Gaussian Mixture Model (GMM) [89], Fallon et al. were able to partition the barometric readings by building floor using the Expectation Maximization [90] (EM) algorithm.

The contributions of this section are most closely related to the work of Fallon et al. [7] with one important distinction. In traditional GMM modeling the number of clusters must be known a priori, and thus either the number of floors in the building must be known or must be deduced from the data. In contrast to Fallon et al. who solve this problem using a set of threshold based heuristics to detect when the operator travels to a new floor, we employ a Bayesian nonparametric approach to automatically determine the number of floors in the building while simultaneously assigning the pressure readings to a particular floor. Furthermore, we demonstrate that the proposed nonparametric is extremely insensitive to choice of the user defined parameter.

The barometer data is modeled most naturally while the barometer is constrained to a single floor c by a Gaussian distribution with mean pressure ϕ_c set by the altitude above sea level and variance σ_b^2 set by the sensor's noise characteristics. As the barometer changes floor, the mean of the Gaussian distribution changes but retains a constant variance. Due to this, the variance is assumed to be a known constant derived from the sensor noise characteristics. This can be obtained through usage training data. If the number of floors was known a priori, a GMM would be sufficient for clustering the data. However, since this information is not available, we instead apply a Chinese Restaurant Process Mixture Model [57] (CRPMM) to allow for an unbounded number of floors to be used during the clustering process.

The CRPMM models the generation of the data according to the following assumptions. The clustering of N pressure readings into floor partitioning $\pi_{[N]}$ is distributed according to the Chinese Restaurant Process with concentration parameter α. Furthermore, given a floor partitioning $\pi_{[N]}$, the mean pressure readings ϕ_c of the floor partitions are distributed according to the known prior Gaussian distribution G_0 with mean set to the mean pressure reading and the variance set to the variance of all data samples. Lastly, we assume that given a floor partitioning and prior distribution on each floors mean pressure ϕ_c, the pressure readings x_i for floor c are themselves distributed independently according to the known Gaussian distribution $F(\phi_c)$ with mean ϕ_c and variance set by the noise level of the system's barometer. We chose these initial distributions to make the following computations tractable.

$$\begin{aligned} \pi_{[N]} &\sim \mathrm{CRP}(\alpha, N) \\ \phi_c|\pi_{[N]} &\overset{iid}{\sim} G_0 \qquad\qquad \text{for } c \in \pi_{[N]} \\ x_i|\phi_c, \pi_{[N]} &\overset{ind}{\sim} F(\phi_c) \qquad \text{for } c \in \pi_{[N]}, i \in c \end{aligned} \tag{3.3}$$

Following the derivation presented in [91], we begin by marginalizing out the ϕ parameters to define the joint probability of the pressure readings and floor partitioning $p(\pi_{[N]}, x)$.

$$p(\pi_{[N]}, x) = \frac{\alpha^K}{\alpha^{(N)}} \prod_{c \in \pi_{[N]}} (|c-1|)! f(x_c) \tag{3.4}$$

The probability measure $f(x_c)$ in the above equation is defined as the marginal probability of the pressure readings associated with floor c.

$$f(x_c) = \int \left(\prod_{i \in c} f(x_i|\phi_c) \right) g_0(\phi_c) d\phi_c \tag{3.5}$$

Here $g_0(\phi_c)$ is defined as the prior probability, drawn from G_0, of floor c having mean pressure ϕ_c and $f(x_i|\phi_c)$ is the conditional probability of pressure reading x_i given its assignment to floor c. Using this definition, we define a Gibbs sampling algorithm [92] to efficiently draw samples of $\pi_{[N]}$.

$$p(\pi|\pi_{-i}, x_i) \propto \begin{cases} \dfrac{|c|}{\alpha + N - 1} f(x_i|x_c) & \text{for } \pi = \pi_{-i} - c + (c \cup x_i), c \in \pi_{-i} \\ \dfrac{\alpha}{\alpha + N - 1} f(x_i) & \text{for } \pi = \pi_{-i} + i \end{cases} \tag{3.6}$$

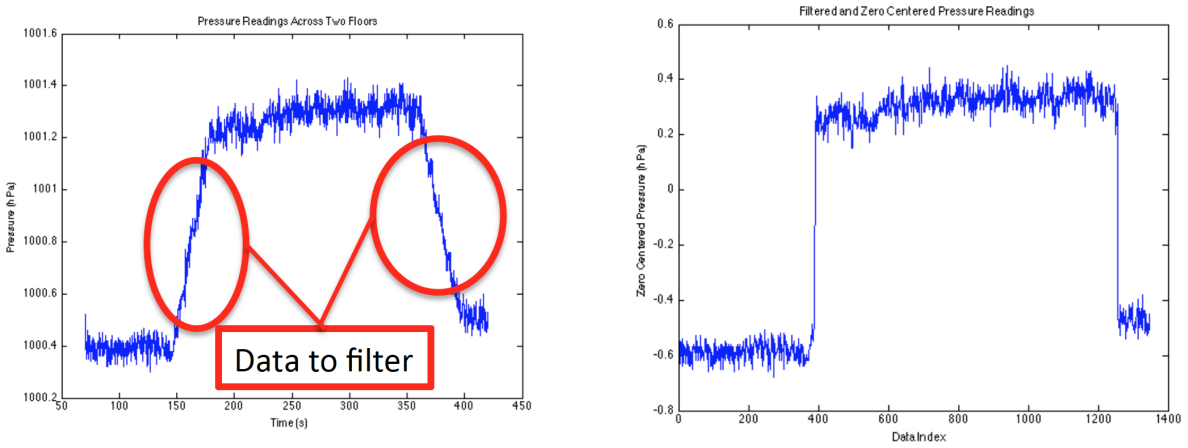
where π_{-i} is the partitioning with sensor reading x_i removed, $-c$ denotes the set subtraction of readings belonging to floor c, and $+(c \cup x_i)$ is set addition of assigning element x_i to floor c. In the above expression the conditional probability of data reading x_i given all data from floor c is given by

$$f(x_i|x_c) = f(x_c \cup i)/f(x_c) \tag{3.7}$$

Equation 3.6 therefore recursively defines the probability of partitioning π in terms of the partitioning with data element x_i removed π_{-i}. The top case corresponds to the probability of assigning data element x_i to cluster c and the bottom case is the probability of creating a new cluster. The above model is sampled using a Monte Carlo Markov Chain (MCMC) method, such as a Gibbs sampler [93], to generate samples from the target distribution. MCMC methods create samples from a target distribution by formulating the sampling process as a Markov Chain and simulating it until it reaches a steady state distribution.

Since the barometric pressure at a given altitude is constant and the sensor is assumed to be corrupted with Gaussian white noise, the most natural choice for the conditional probability of a pressure reading given the floor's mean pressure ϕ_c and sensor noise σ_b^2 is a Gaussian distribution.

$$f(x_i|\phi_c) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\Big(-\frac{1}{2\sigma_b^2}(x_i - \phi_c)^2 \Big) \tag{3.8}$$



(a) (b)

Figure 3.5: An example result of filtering transition sections from the barometric pressure readings. (a): The raw pressure readings. The transition sections have been highlighted in red. (b): The filtered pressure readings with transition sections successfully removed.

To make the integral in Eq. 3.5 tractable, we choose the prior distribution of mean pressures to be Gaussian with mean ϕ_0 and variance σ_b^2/k_0. This allows the us to rewrite Eq. 3.5 in closed form. In practice, the pressure data is mean-centered before processing so $\phi_0 = 0$ and k_0 is chosen to be the variance of the entire set of pressure readings.

$$f(x_c) = \frac{1}{(2\pi\sigma_b^2)^{|c|/2}} \frac{1}{(2\pi\frac{\sigma_b^2}{k_0})^{1/2}} \int \exp\left(\frac{-1}{2\sigma_b^2}\left(\sum_{i\in c}(x_i-\phi_c)^2 - k_0(\phi_0-\phi_c)^2\right)\right) d\phi_c \tag{3.9}$$

After algebraic manipulation and omission of any constants of proportionality we are left with an expression that is only a function of the first and second moments of the pressure readings associated with the floor assignment.

$$f(x_c) = \frac{1}{(2\pi\sigma_b^2)^{|c|/2}} \frac{1}{(2\pi\frac{\sigma_b^2}{k_0+|c|})^{1/2}} \exp\left(\frac{-1}{2\sigma_b^2}\left(\sum_{i\in c} x_i^2 - \frac{(k_0\phi_0 - \sum_{i\in c} x_i)^2}{k_0+|c|}\right)\right) \tag{3.10}$$

The above expression is efficiently evaluated by a Gibbs sampler since it only contains the first and second moments of the floor partition's pressure readings, $\sum_{i\in c} x_i$ and $\sum_{i\in c} x_i^2$ respectively. The Gibbs sampling process results in a discrete set of samples from probability distribution of the partitions. Since the number of partitions is combinatorially large, it is unreasonable to assume that a dense sampling of the distribution can be obtained for arbitrarily large datasets. To overcome this, the log-likelihood of all sample partitions S is computed and the partition s which maximizes the log-likelihood was chosen as the final clustering.

Algorithm 1 The CRPMM floor segmentation algorithm.

```
S ← ∅
x = filter_data(x)
while i < num_samples do
    s ← make_gibbs_sample(x)
    S ← S ∪ {s}
    i ← i + 1
end while
π_[N] ← maximize_loglikelihood(S)
return π_[N]
```

$$\max_{s \in S} \sum_{c \in \pi_{[N]}} \log(f(x_c)) \tag{3.11}$$

It is important to note that before the CRPMM can be used to fit the data, sections where the operator is transitioning between floors must be removed so the data fits the Gaussian Mixture Model assumption. Although filtering transitioning sections reveals if the operator is traversing upward or downward through the building, it does not solve the problem of detecting the number of floors without additional heuristics. When an operator revisits a floor, data must be clustered across time to group the individual floor's segments. The CRPMM is able to simultaneously detect the number of floor while clustering the data without additional heuristics.

Algorithm 1 summarizes the CRPMM based floor segmentation algorithm. The algorithm begins by filtering the raw pressure readings to zero-center the data and remove any sections where the operator is transitioning floors. Then, a Gibbs sampler is used to generate independent samples from the distribution of possible floor partitions using Eq. 3.6 in conjunction with 3.10. Finally, the best partition is selected by iterating over the generated samples and selecting the one that maximizes the log-likelihood via Eq. 3.11.

Elimination of transitional sections is performed by first computing the variance of the pressure readings using a sliding window across the data. Then, if the variance in any segment is larger than σ_b^2 then it is safe to assume that the section corresponds to a transition and thus can be discarded. For all experiments presented here a window size of 10 was deemed sufficient. Figure 3.5 shows an example result of pressure filtering to remove transitions sections. Shown in Fig. 3.5(a) is an example dataset consisting of an operator traversing two stories connected by a stairwell. Figure 3.5 shows the results of filtering out transition sections. Note that the mean pressure has been removed so that the data is centered around zero for numeric stability.

Lastly, some important numerical considerations need to be observed when computing this function to avoid numerical under and over flow problems. In particular, the normalizers involve exponentiating $2\pi\sigma_b^2$ to the power of $|c|/2$. When $|c|$ is large this can easily cause overflow problems. Luckily, these terms are only large when considering $f(x_i|x_c)$ from Eq 3.7 and thus we can perform cancellation of this term in the normalizer to avoid this

issue. Secondly, barometric pressure at sea-level is 101325 Pascals and thus squaring pressure readings and taking the negative exponential can lead to numerical underflow issues. To compensate for this problem, all data is mean-shifted to have mean zero before being fit with the CRPMM. Experimental results are shown in Section 3.5.1.

3.2.4 Rao-Blackwellized Particle Filtering

In this section we discuss the process of fusing the local submaps generated in Section 4.8.1 into a series of geometrically consistent occupancy grid maps.

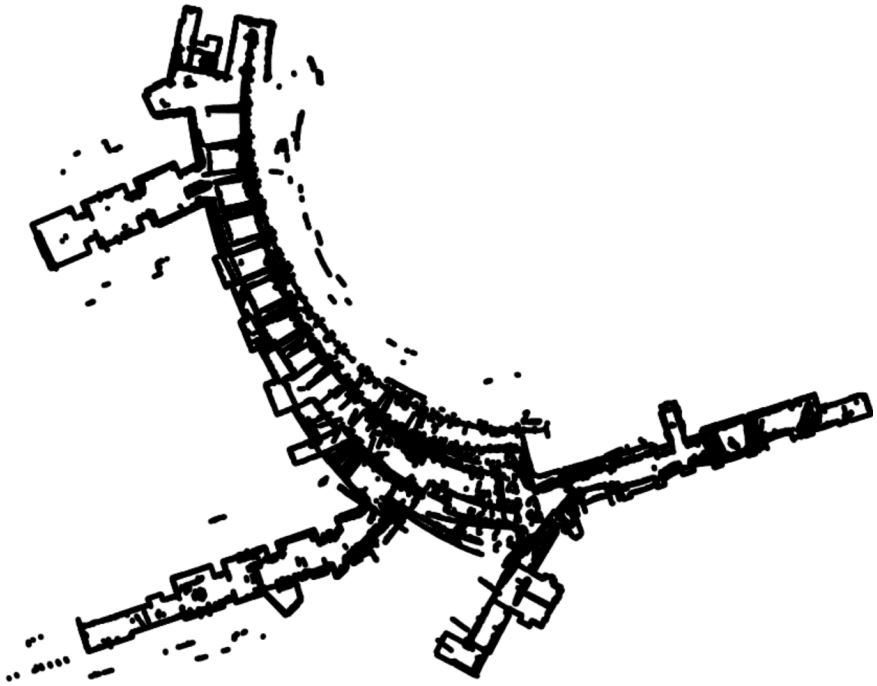
Once the submaps have been created, we combine them into a series of geometrically consistent occupancy grid maps using a second pass of Rao-Blackwellized particle filtering. This process begins by matching sequential submaps via FICP scan matching to obtain new odometry estimates between the submap locations. We then fuse the submaps and the newly generated odometry estimates using the Sampling Importance (SIR) filter and adaptive proposal distribution [32] with two main distinctions. First, our submaps are not directly generated from a laser range finder and thus this model is not appropriate. Rather, we use a simple approach that approximates the importance weight of each particle by using the spatial correlation of the new sensor readings with the particle's current map:

$$w_i \propto \frac{|S_i \cup M_{i-1}|}{\min(|S_i|, |M_i|)} \tag{3.12}$$

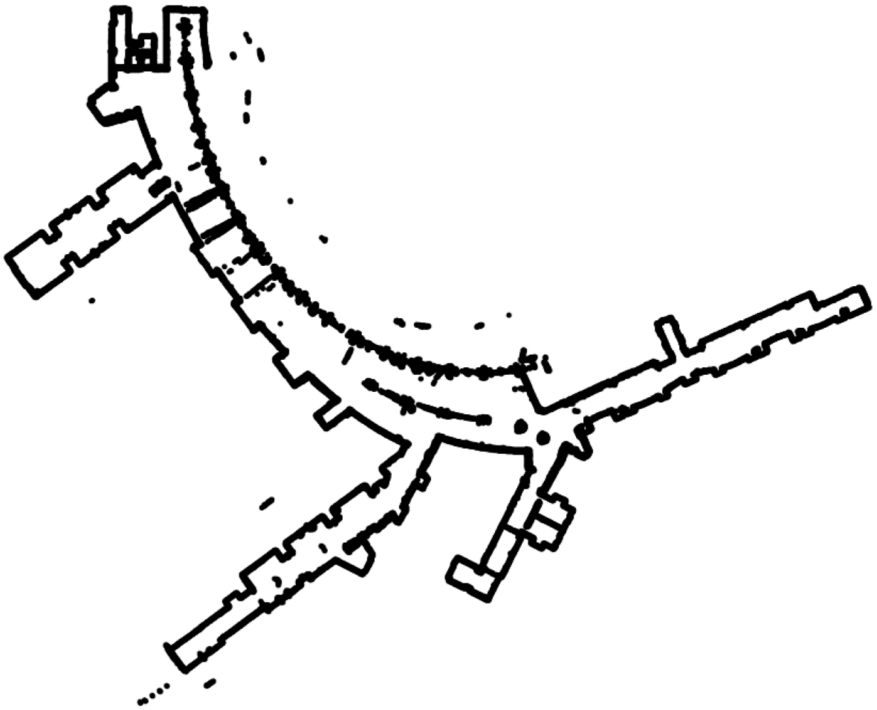
where S_i is the set of points in the i^{th} submap, M_{i-1} is the map at time $i-1$, and w_i is the approximated importance weighting. Secondly, we incorporate the floor assignments derived from Section 3.2.3 by creating a collection of 2D occupancy grid maps. Since each cluster detected by the CRPMM corresponds to a single floor of the building, each particle creates a occupancy grid map for each barometric pressure cluster. In this way, we extend the standard 2D particle filtering algorithm to multi-story buildings in a natural fashion without making assumptions about the layout or vertical stacking of the building.

Figure 3.6 shows an example of the RBPF algorithm. A trajectory of approximately 740 m was traversed over a period of 15 min in a hotel. The map formed by using dead reckoning alone is shown in Fig. 3.6(a) while the map created by the RBPF is shown in Fig. 3.6(b). The occupancy grid map generated by the RBPF is geometrically consistent even when traversing previously visited locations.

While the resulting grid maps are geometrically consistent, the accuracy is still fundamentally limited by the size of the grid cells used. Despite averaging the contributions to each grid cell as its representative point, quantization errors still exist. Furthermore, because temporally adjacent scans are first aggregated into local submaps, the system trajectory resulting from the RBPF contains poses for only the locations of the submaps, not the original sensor reading locations. In order to compute poses for all sensor readings we must interpolate between the locations of the submaps. The interpolation is carried out by formulating a graph optimization problem where the poses are the nodes of the graph, the odometry readings serve as the edges between temporally adjacent poses, and loop closures extracted from the occupancy grid map provide global constraints on the graph. By fusing the full-



(a)



(b)

Figure 3.6: An example result of applying the Rao-Blackwellized particle filtering algorithm to the generated submaps. (a): The map generated using only odometry. (b): The map resulting from the RBPF using 100 particles at a resolution of 10 cm.

rate odometry and RBPF localization results into a single full-rate trajectory we obtain a geometrically consistent trajectory with no temporal or spatial quantization.

3.3 Loop Closure

3.3.1 Loop Closure Extraction

This section describes the proposed methodology for extracting loop closure constraints from the occupancy grid maps created in Section 3.2.4. To detect loop closures we first convert the map into a representation that defines a measure of similarity between poses in the trajectory. Using both the grid map and accompanying trajectory, we explicitly recover which poses observe which occupancy grid cells from the map. We define the correlation $C(i, j)$ between each pose i and j using the following function:

$$C(i,j) = \frac{|Z_i \bigcap Z_j|}{\min(|Z_i|, |Z_j|)} \tag{3.13}$$

where Z_i and Z_j are the set of occupancy grid cells observed by pose i and j respectively and $|Z|$ is the number of elements in set Z. We repeat this for all pairs of poses and collect the resulting coefficients into a correlation matrix C. Fig. 3.7(a) shows the correlation matrix formed from the grid map of Fig. 3.6(b). Note that regions of high correlation that are located off diagonal correspond to temporally distant pose pairs that contain a large number of overlapping grid cells.

Next we apply hierarchical clustering to group the poses into smaller sets that contain similar geometry in order to reduce the search space of possible loop closure locations. The clustering algorithm begins with each pose of the trajectory as its own cluster, and iteratively joins the most similar clusters until either the required number of clusters is obtained or all remaining clusters have a similarity metric of 0 [94]. The chosen definition of similarity can dramatically affect the clustering results. We define the similarity between two clusters X and Y using the arithmetic mean between cluster elements [94]:

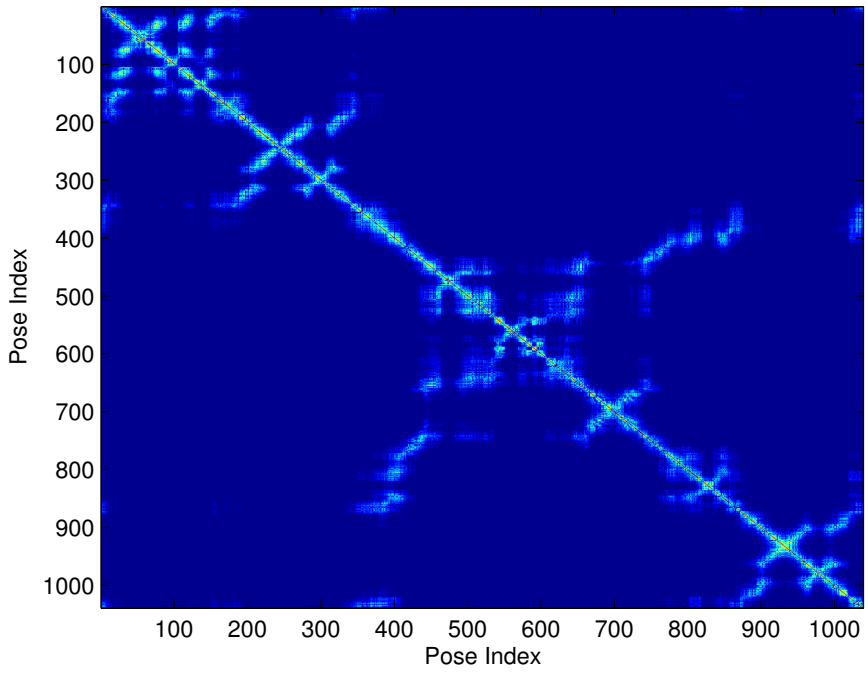
$$s = \frac{1}{|X||Y|} \sum_{i \in X} \sum_{j \in Y} C(i,j) \tag{3.14}$$

Additionally, for a hierarchical clustering algorithm it is necessary to first determine the number of desired clusters. We compute this by inspecting the eigenvalues of the correlation matrix C. Given the eigenvalues $\lambda_1^C < \lambda_2^C < \ldots, < \lambda_{\max}^C$ we choose the number of clusters by finding the first eigenvalue $\lambda_{n_c}^C$ where

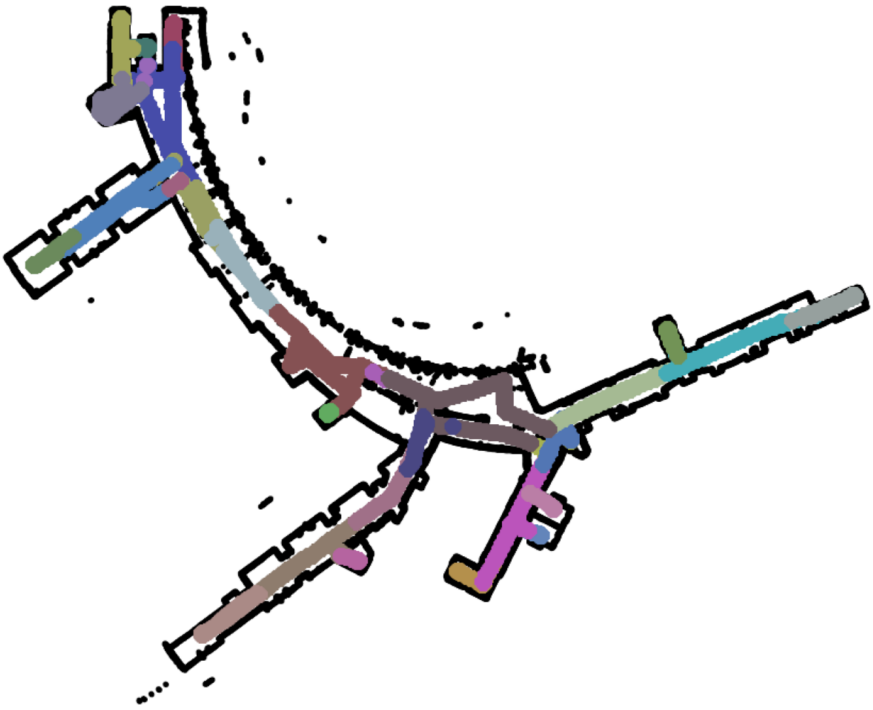
$$\frac{\lambda_{n_c}^C}{\lambda_{\max}^C} \geq r^C \tag{3.15}$$

and denote the number of clusters by n_c. We have empirically found that $r^C = 0.3$ provides a sufficient number of clusters for our experiments.

Figure 3.7(b) shows the results of the hierarchical clustering algorithm on the correlation

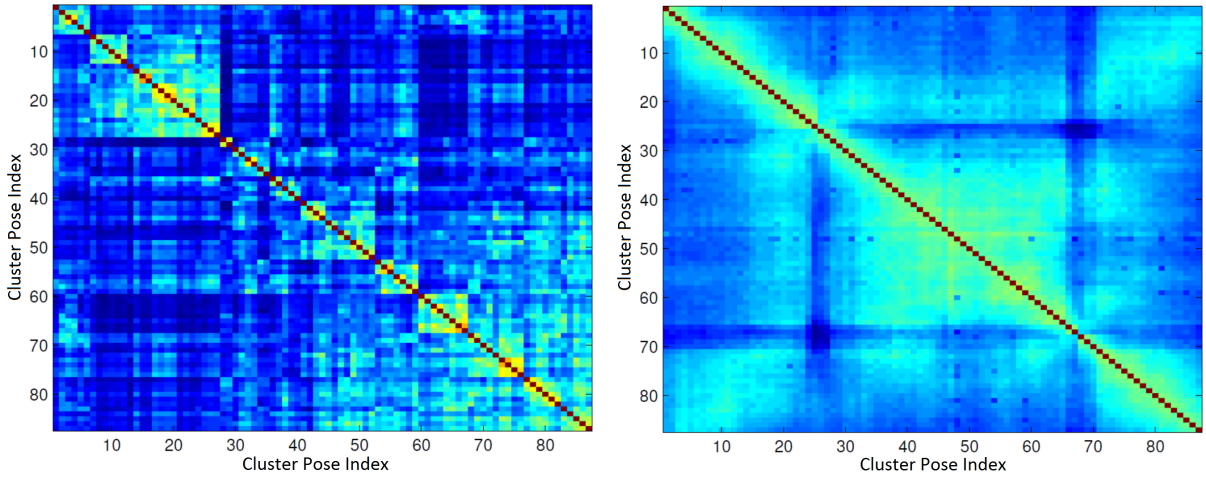


(a)



(b)

Figure 3.7: Correlation Matrix Based Clustering. (a): Correlation matrix formed from the occupancy grid map of Figure 3.6. (b): The results of clustering the correlation matrix from (a).



(a) (b)

Figure 3.8: Optimization of the correlation matrix used in clustering. (a): The correlation matrix of a cluster before optimization of pairwise constraints. (b): The correlation matrix after FICP has been run on the pairwise constraints.

matrix of Fig. 3.7(a). Each cluster's poses are shown using a different color overlain on the occupancy grid map. The clustering algorithm correctly groups the poses spatially even when the poses are temporally distant. This means we only must search for loop closure indices inside a single cluster which greatly reduces the search space and speeds up the subsequent cluster correlation matrix optimization.

Once the clusters have been identified, we extract the portion of the correlation matrix that corresponds to each cluster. Fig. 3.8(a) shows an example of a cluster's correlation matrix. However, because the occupancy grid map may still have small errors in the poses, we optimize the alignment between each pair of poses within a cluster to further refine each cluster's correlation matrix. To do so, we run a round of FICP on each pair of scans to optimize alignment before computing the correlation using Eq. 3.13. Fig. 3.8(b) shows the result of applying the optimization procedure to the similarity matrix from Fig. 3.8(a). The resulting matrix contains smoother gradients between regions which adds local consistency. Additionally, a greater number of strong correlation peaks are present which reduces the rate of missed loop closure candidates. We then select candidate loop closures using the local maxima of the clusters' correlation matrices. The intuition for this is that local maxima located away from the matrix diagonal correspond to revisited locations with a large amount of overlapping geometry. To avoid grouping loop closures candidates too closely, we smooth the clusters' correlation matrices using a 3-by-3 averaging filter before selecting candidate pairs and enforce the restriction that a pose can only be part of a single loop closure candidate.

Figure 3.9(a) shows the result of detecting correlation maxima on the cluster correlation matrix of Fig. 3.8(b). The maxima from the lower triangular section of the matrix are shown using black dots. Fig. 3.9(b) shows a close up of the upper portion of the trajectory. The candidate pairs corresponding to the detected maxima are connected via green lines and the red dots indicate poses in the cluster. The result of maxima detection for all clusters

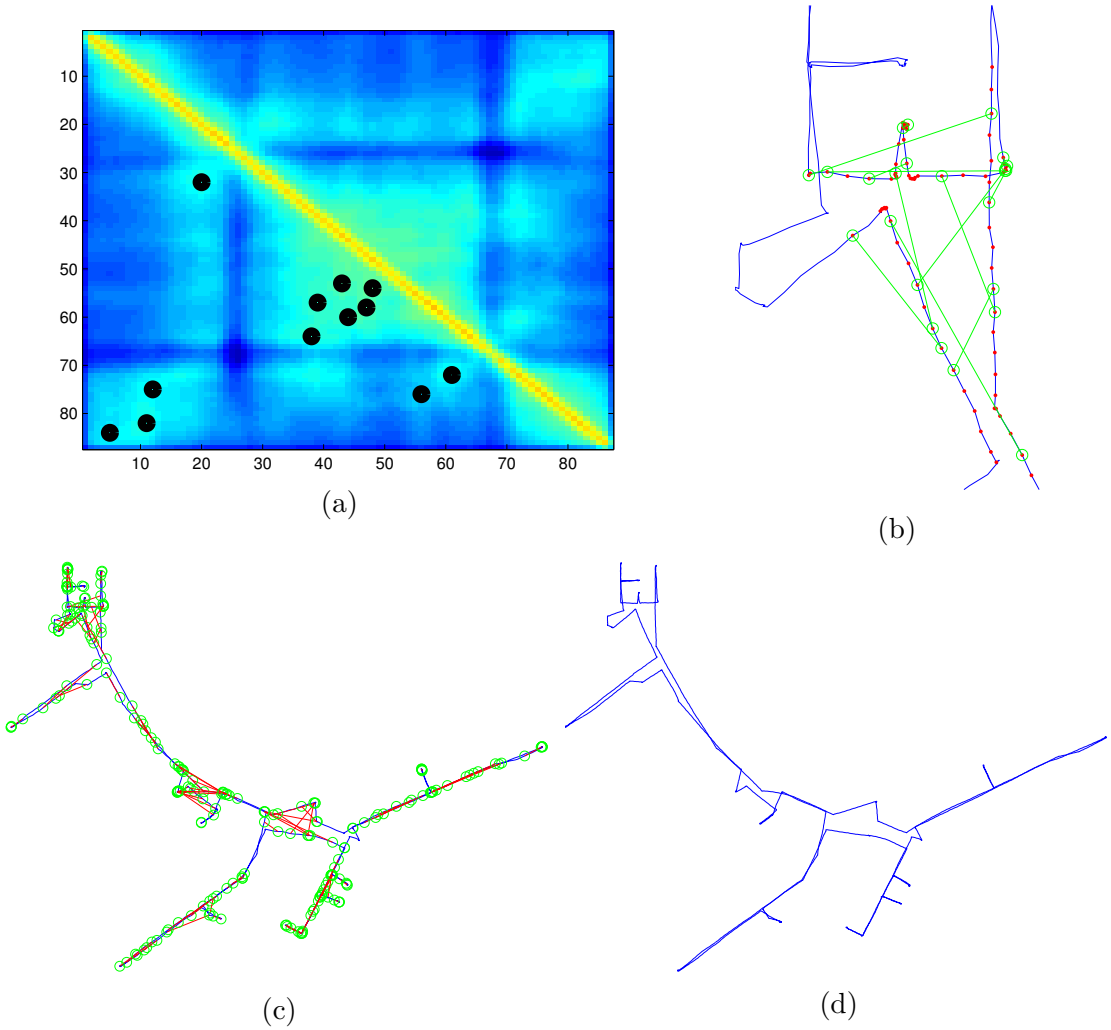


Figure 3.9: Loop Closure Selection From Cluster Correlation. (a): The detected maxima of the cluster correlation matrix of Figure 3.8(b). The maxima from the lower triangular section of the matrix are shown using black dots. (b): The loop closure pairs corresponding to the detected maxima from (a). (c): The result of detecting maxima from all clusters is shown by green circles connected by red lines. (d): The fully optimized trajectory.

is shown in Fig. 3.9(c). As shown, the algorithm correctly extracts loop closure constraints at revisited locations and between poses that contain a significant portion of overlapping geometry.

3.3.2 Loop Closure Transform Estimation

Once the candidate pairs have been extracted, the transformation and covariance matrices must be computed before they can be used in a graph optimizer. Since the submaps from Section 3.2.2 are spatially quantized, we take the original pair of scans corresponding to a given loop closure, apply the scan projection algorithm of Section 3.2.1, and match them via a genetic scan matching algorithm. The result is a set of metric transformations between

loop closure pose indices.

A widely known limitation of non-linear optimization techniques is that they can fall into local minima of the objective function. To overcome this limitation, the algorithm is typically supplied with an initial condition $T(\cdot, \mu_0)$. The resulting solution is often highly dependent on choice of $T(\cdot, \mu_0)$ and a poor choice of initial condition can lead to an undesirable solution. A class of stochastic optimization techniques, known as genetic algorithms, can overcome this problem by providing a derivative-free approach for exploring the solution space. Each solution, or chromosome, is evaluated according to a cost function and assigned a fitness value. Thus, genetic searches are able to better cope with the presence of local minima or poor initial conditions [95].

Genetic search (GS) algorithms operate by first considering a randomly distributed set of solution chromosomes. Using an appropriate fitness metric, the population is evaluated and each chromosome is assigned a fitness value. The most fit chromosomes are retained and the rest are discarded. The population is then replenished by creating new chromosomes whose parameters are chosen randomly from the surviving chromosomes. The generated solutions are mutated by adding random variation to the chromosomes' parameters and the process is iterated until the population reaches a stable configuration. In this manner, GS algorithms avoid local minima while exploring solutions that are not in the original sampling of the solution space.

Similar to the previous works of Martinez et al. [96] and Lenac *et al.* [59], we parameterize the solution space using a three element chromosome. Specifically, each individual in the population consists of two translational and one rotational parameter.

$$\mu = \begin{bmatrix} dx, & dy, & d\theta \end{bmatrix}^T \tag{3.16}$$

In this parametrization dx and dy represent incremental translations in the x and y directions respectively while $d\theta$ represents a counter-clockwise rotation.

In contrast to the existing approaches of [59, 96], we propose a new algorithm, Fractional Genetic Scan Matching (FGSM), which introduces a transformation function into the lifetime of each chromosome. Specifically, FGSM starts by considering a random set of chromosomes, S_0, sampled uniformly from a rough initial condition, μ_0. The initial condition allows for a priori information, such as odometry, to be incorporated into the FGSM algorithm. Denoting each individual chromosome $\mathbf{s}_i$, we transform the initial population by applying the ICP optimization procedure to obtain the optimal transformation $\mathbf{r}_i$. The fitness value for each chromosome $\mathbf{s}_i$ is set to the residual of the objective function from Eq. 3.2 evaluated at $\mathbf{r}_i$. The chromosomes with the best fitness scores are replaced by their optimal transform and the rest are discarded. Because each chromosome is replaced by its optimal transformation $\mathbf{r}_i$ the FGSM algorithm can be interpreted as a combination of a stochastic and gradient-based search over a subset of the local minima of the objective function.

To simulate the mutation between generations, random i.i.d. Gaussian noise is added to the newly generated chromosomes. The variance of the added Gaussian noise is determined by considering the variance of the remaining chromosomes' parameters. Specifically, if the population's n^{th} parameter has variance σ_n^2, then a random sample drawn from $\mathcal{N}(0, \sigma_n^2)$ is added to each of the generated chromosome's n-th parameter. The process of transformation, fitness evaluation, selection, and mutation is iterated until the population converges to a

Algorithm 2 The FGSM Algorithm

1: $\mu_0 \leftarrow$ initial estimate
2: $S_0 \leftarrow random\_chromosomes(\mu_0)$
3: **while** $S_{n+1} \neq S_n$ **do**
4: $\quad R_n \leftarrow \emptyset$
5: $\quad$ **for** $\mathbf{s}_i \in S_n$ **do**
6: $\quad\quad \mathbf{r}_i \leftarrow ICP(\mathbf{s}_i)$
7: $\quad\quad R_n \leftarrow R_n \cup \{\mathbf{r}_i\}$
8: $\quad$ **end for**
9: $\quad R_n \leftarrow best\_subset\_of(R_n)$
10: $\quad S_{n+1} \leftarrow R_n \cup make\_children(R_n)$
11: **end while**
12: **return** μ and D_f from best chromosome.

single dominant trait. This procedure is summarized in Algorithm 2.

An example of the FGSM algorithm is shown in Fig. 3.10. Figure 3.10(a) shows the initial chromosome population sampled uniformly around the supplied initial condition μ_0. Shown in Fig. 3.10(b) is the distribution of chromosomes after a few iterations. Notice that the chromosomes only inhabit a few locations corresponding to the local minima of the objective function Eq. 3.2. After enough iterations have passed, the population, shown in Fig. 3.10(c), has converged to a single dominant chromosome representing the correct transformation parameters.

As seen in line 6 of Algorithm 2, each chromosome is used as the initial condition to the ICP procedure and is replaced by the resulting locally optimal transform. While this step forces each chromosome into a local minima of the objective function, it is also computationally expensive. Furthermore, as the population nears convergence many chromosomes reside in approximately the same location resulting in redundant computation that should be avoided.

Examining Eq. 3.2, we notice that if both the point matches and the inlier set are given, then the ICP problem reduces to minimizing the following objective function:

$$e = \sum_{\mathbf{q}_i \in D_f} \|\mathbf{n}_i^T(\mathbf{p}_i - T(\mathbf{q}_i, \mu))\|^2 \tag{3.17}$$

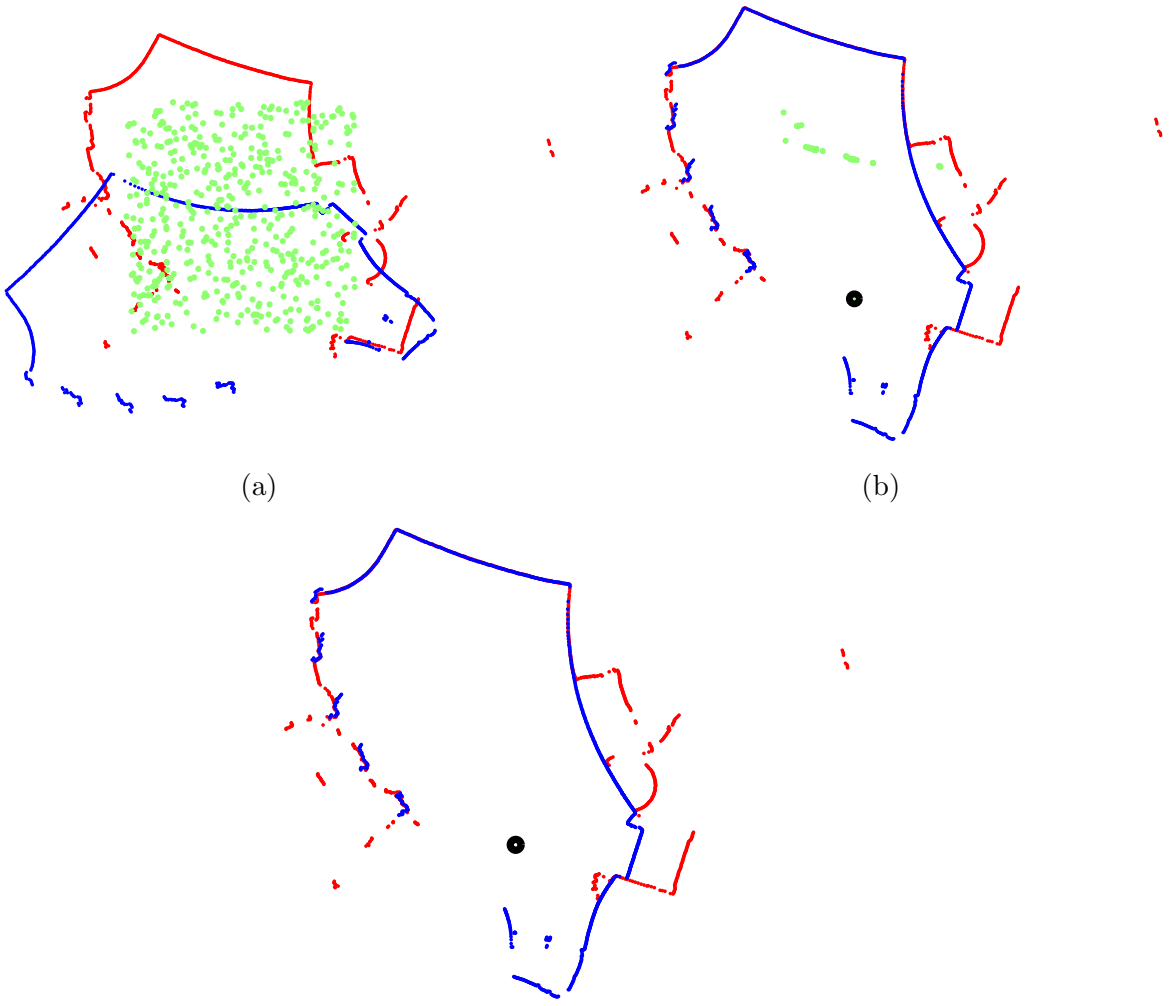
Following [47], Equation (3.17) can be rewritten as:

$$e = \mathbf{e}(\mu,\ D_f)^T \mathbf{e}(\mu,\ D_f) \tag{3.18}$$

where i-th element of vector $\mathbf{e}(\cdot)$ corresponds the error contributed from the i-th element of inlier set D_f. Using the Taylor expansion with respect to the transformation parameters, a linearized version of $\mathbf{e}(\cdot)$ can be written as:

$$\mathbf{e}(\mu_0,\ D_f) \approx \mathbf{e}(\mu_0,\ D_f) + \mathbf{H}\mu_0 \tag{3.19}$$

where μ_0 is the linearization point and $\mathbf{H} \in \mathbb{R}^{|D_f| \times 3}$ is the Jacobian of $\mathbf{e}(\cdot)$ with respect to



(a) (b) (c)

Figure 3.10: An example of the FGSM process. The scans are shown in red and blue while the chromosomes are shown in green. A black circle denotes the location of the best chromosome in the population. (a): The initial chromosome sampling. (b): After a few generations the chromosomes have converged to a subset of the local minima. (c): After enough generations have passed the population has converged to a single dominant location.

μ. The i-th row of $\mathbf{H}$ is computed via

$$\mathbf{H}_i = \begin{bmatrix} -\mathbf{n}_i \\ \mathbf{n}_i^T \begin{bmatrix} \sin(d\theta) & \cos(d\theta) \\ -\cos(d\theta) & \sin(d\theta) \end{bmatrix} \mathbf{p}_i \end{bmatrix}^T \tag{3.20}$$

and the optimal transform can be found using

$$\mu = -(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{e}(\mu_0,\ D_f) \tag{3.21}$$

Equations 3.20 and 3.21 reveal that when the Jacobian is evaluated around small angles, $d\theta \approx 0$, small deviations $d\mu$ from the linearization point result in insignificant changes to the resulting transformation. This analysis relies on the fact that the objective function is continuous, which in this case is not true. However, Eq. 3.17 is locally continuous provided the deviations $d\mu$ are small enough to not alter the nearest neighbor point matches.

In order to take advantage of the near-linearity of the Jacobian, we take the following strategy. We first discretize the solution space into small grid cells and bin chromosomes according to their parameters. Each time a chromosome is used as the initial condition for ICP, a record is kept of the resulting transformation. In this way, redundant computation can be eliminated by building a lookup table between initial conditions and the resulting transforms.

The amount of discretization in the solution space represents a trade-off between speed and adherence to the underlying assumptions. The discretization must represent small enough changes so as not to violate the small-angle approximation or the continuity of the objective function, yet be large enough to effectively speed up the genetic search. We have empirically found a spacing of 10 centimeters and 1 degree to provide the best trade-off between computation time and accuracy for our experimental setup. Section 3.5.4 provides empirical justification for these values.

3.3.3 Loop Closure Transformation Verification

Even though the FGSM algorithm is a robust way of matching loop closure scans with unknown initial conditions, care must still be taken to eliminate erroneous transformations. Poor scan matches can arise when a loop closure has been falsely identified or when geometry is ill-conditioned. In these situations it is best to recognize that the algorithm has failed and avoid using the loop closure constraint. In this section we now propose two metrics, independent of the detection and estimation tasks, to robustly reject erroneous loop closure transformations while maintaining a high true positive rate. The proposed metrics are designed to encompass two important factors for characterizing a loop closure transformation: the amount and complexity of the shared geometry.

The first metric characterizes the amount of shared geometry between matched scans. Using the proposed transformation from Section 3.3.2, the scans are rotated and translated into alignment. Next, a pair of normalized two-dimensional histograms are built by binning the scans' points into equally sized spatial bins. Then the histograms are correlated using

the intersection kernel [97]:

$$c = \sum_{i,j} \min(h_1(i,j), h_2(i,j)) \tag{3.22}$$

where $h_1(i,j)$ and $h_2(i,j)$ corresponds to the i,j-th bin of the respective histograms. This measure evaluates to $c = 1$ if the histograms are identical and $c < 1$ if they differ.

Our second metric measures the complexity of the overlapping geometry. Using the proposed transform from Section 3.3.2, Eq. 3.2 can be minimized to obtain an inlier set D_f. Considering only the points in D_f, the portion of geometry that is shared between grid maps is identified. Next, the normal vectors of the inlier points, $\mathbf{n}_i$ are collected into the matrix

$$\mathbf{N} = \left[\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_n\right]^T \tag{3.23}$$

with which the correlation matrix:

$$\mathbf{R} = \mathbf{N}^T\mathbf{N} \tag{3.24}$$

is formed. The eigenvalues of $\mathbf{R}$, denoted by λ_1^R and λ_2^R, are used to form the ratio

$$r^R = \frac{\lambda_1^R}{\lambda_2^R} \tag{3.25}$$

assuming $\lambda_1^R \leq \lambda_2^R$. When geometry is ill-defined for scan matching, such as in a long featureless hallway, the normal vectors of the points lie almost entirely along the line normal to the walls. This observation implies that λ_1^R, and by extension r^R, is close to zero. On the other hand, in situations where geometry is well defined, such as in a complex environment with many corners, $\lambda_1^R \approx \lambda_2^R$ and thus r^R is close to 1. While the normal vector for each point is not detected using a range sensor, in practice techniques such as SVD analysis of the point's local neighborhood can provide a reasonable estimate.

3.4 3D Point Cloud Generation

3.4.1 Height Estimation

After the optimized 2D path has been obtained, the height of the system must be estimated at each pose before a full 6DOF path can be recovered. We recover height using the adaptive method presented in [79] whereby the floor scanner's, as shown in the CAD drawings of Figure 2.5, data is classified as either planar or non-planar. If the floor is detected planar, then a direct estimate of the system's height to the floor can be made. However, if the system is not on flat terrain, sequential scan data from the floor scanner can be matched to recover the incremental change in height [77]. The incremental and absolute measurements are combined to estimate the height of the system in the global frame of reference.

3.4.2 IMU and Height Data Fusion

The height information along with the IMU readings and the optimized 2D path are combined to form a full six degree of freedom trajectory. The six degree of freedom pose at time t, $\mathbf{x}_t$, is comprised of

$$\mathbf{x}_t = [x_t, y_t, z_t, \phi_t, \psi_t, \theta_t]^T \tag{3.26}$$

where x_t, y_t, and z_t are the x, y, and z positions and ϕ_t, ψ_t and θ_t are the roll, pitch and yaw orientations. The methods of this chapter use the 3-2-1 Euler angle convention to describe orientations in 3D space. Although quaternions are generally preferred for navigation and optimization applications, Euler angles are used in this chapter because the orientation components are not directly used in numerical optimization. In the 3-2-1 Euler convention roll, corresponds to a rotation about the x-axis, pitch corresponds to a rotation about the y-axis, and yaw corresponds to a rotation about the z-axis. Specifically, if $R_a(b)$ is a rotation about axis a by angle b, then the rotation matrix that transforms from the local coordinate frame $\{B\}$ to the global coordinate frame $\{G\}$ is given by:

$$\overset{G}{_B}\mathbf{R} = \mathbf{R}_z(\theta)\mathbf{R}_y(\psi)\mathbf{R}_x(\phi) \tag{3.27}$$

Since the 2D SLAM and height estimation algorithms provide values in the global coordinate system, the position component of $\mathbf{x}_t$ is built simply by concatenating the previously obtained values. The rotational components however must be formed more carefully. The 2D path reconstruction recovers the heading angles α_t which represent the projection of the operator's forward direction into the global xy-plane. Since the operator has non-zero pitch and roll, the heading and yaw angle are not identical in the general case. However, we can derive the correct yaw angle by combining the pitch, roll, and heading into the correct rotation matrix via:

$$\overset{G}{_B}\mathbf{R} = \mathbf{R}_z(\alpha_t - \beta_t)\mathbf{R}_y(\psi_t)\mathbf{R}_x(\phi_t) \tag{3.28}$$

The correction factor β_t is the calculated from the rotation matrix if only the roll and pitch components of the orientation were applied.

$$\overset{G}{_B}\mathbf{R}' = \mathbf{R}_z(0)\mathbf{R}_y(\psi_t)\mathbf{R}_x(\phi_t) \tag{3.29}$$

Using $\overset{G}{_B}\mathbf{R}'$, the heading correction factor β_t is then computed by projecting the systems look vector into the global xy-plane and finding the angle it makes with the global x-axis. By accounting for the heading correction factor β_t we ensure that even after the pitch and roll is applied the heading angle remains α_t.

The rotation matrix in Equation 3.28 is then decomposed into the correct roll, pitch, and yaw values using the Given's rotations [98]. Concatenating the position and orientation values yields the full six degree of freedom pose. Figure 3.9d shows an example of an optimized path using the example data shown in Figure 3.9(a)–(c).

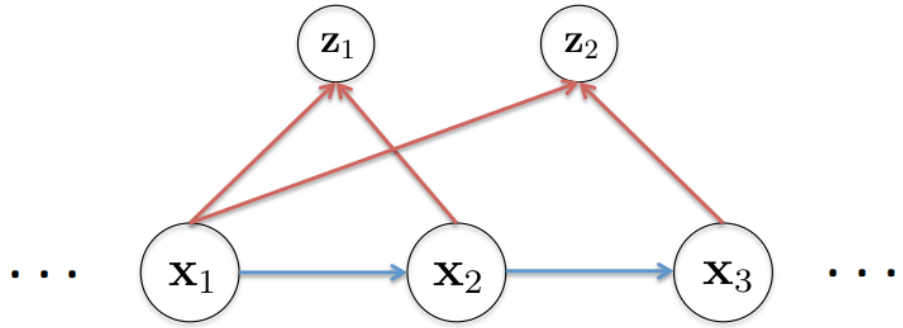


Figure 3.11: An example depicting how control points can be utilized in the graph optimization framework. Poses $\mathbf{x}_1$ through $\mathbf{x}_3$ observe landmarks $\mathbf{z}_1$ and $\mathbf{z}_2$. The blue arrows depict the pairwise constraints on the poses from the odometry measurements while the red arrows depict the range and bearing constraints from the control point observations.

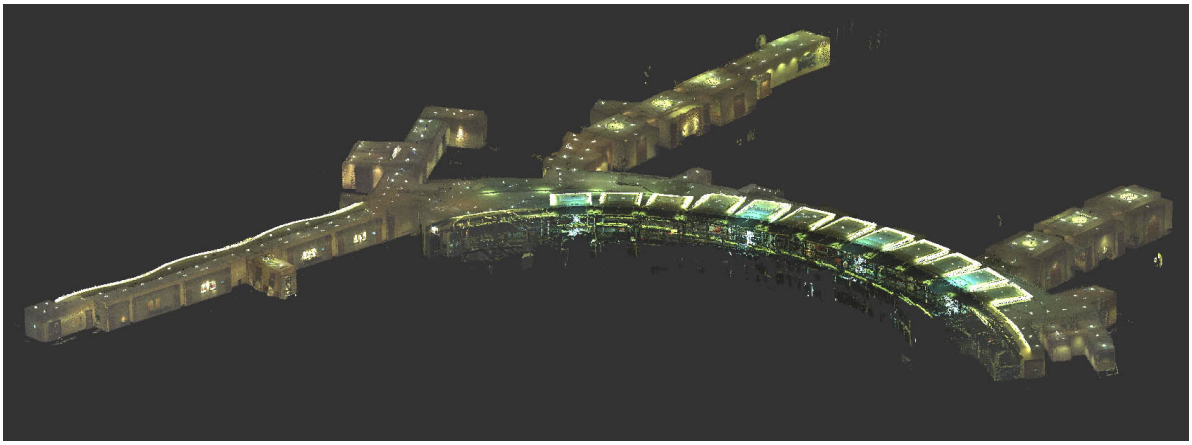
3.4.3 Incorporating Control Points

Control point targeting is a common practice in architectural modeling and historical preservation applications. Static scanning systems, such as the Leica Scanstation [99], must have target points in order to stitch scans taken from different locations into a single point cloud. To automatically detect and utilize these points, static scanning systems often require the user to hang checkerboard like targets throughout the environment. Once registered, the (x, y, z) location of the control points can be estimated accurately to less than a millimeter.

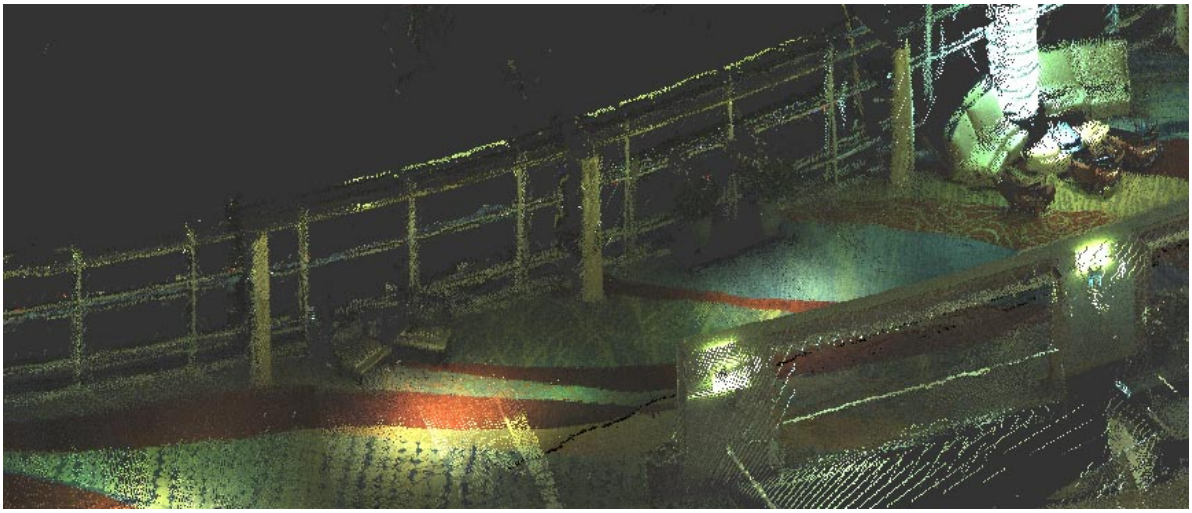
If control points are available, our ambulatory backpack system can utilize them to improve the accuracy of the reconstructed trajectory. Similarly to static scanning stations, our backpack system does not require the control points to be pre-surveyed in order to use them to improve the accuracy of the reconstructed trajectory. Figure 3.11 conceptually shows how control points can be naturally utilized by a factor graph optimization framework such as GTSAM [35]. Poses $\mathbf{x}_1$ through $\mathbf{x}_3$ contain the initial estimate of the trajectory of the backpack and are related via the odometry measurements depicted as blue arrows. The control points $\mathbf{z}_1$ and $\mathbf{z}_2$ are observed by the system as it travels through the environment. The control points observations, shown as red arrows, define a range and bearing measurement from the pose $\mathbf{x}_i$ to landmark $\mathbf{z}_k$. This defines additional constraints on the graph which improve the overall trajectory of the system. See Section 3.5.8 for the quantitative effects of adding control points to improve the accuracy of the reconstructed trajectory.

3.4.4 Point Cloud Generation

Using the data collected by the left and right geometry scanners and the recovered trajectory we are able to generate a dense, 3D point cloud by placing the sensor readings at the correct pose location. Since the system is also equipped with 2 fish-eye cameras, the sensor readings can be colorized by projecting them into the temporally closest image. Although the hardware systems used contain global shutter cameras, the 2.5D localization methods of this Chapter do not consider any timing delays in the laser data. If these effects were com-



(a)



(b)

Figure 3.12: An example point cloud generated from the 2.5D algorithm. (a): The exterior view of a dense, colorized, 3D point cloud generated by the backpack system. (b): A close up view of an interior section illustrates the point clouds high level of detail.

pensated for, the generated point clouds could be generated and colorized more accurately. Figure 3.12(a) shows an example point clouded generated using the trajectory from Fig. 3.9. A close up shot of some detailed furniture is shown in Fig. 3.12(b). The point cloud contains over 106 million points each with a time stamp, position, color, and unique identifier. When represented in a space delimited ASCII file, the point cloud takes up approximately 4 GB on disk.

3.5 Results

In this section we present results for the proposed 2.5D localization algorithms. Section 3.5.1 shows the experimental verification of the automatic floor segmentation algorithm. Section 3.5.2 shows example results of the multi-story particle filtering algorithm. Section 3.5.3

presents performance results of the FGSM algorithm using a variety of initial conditions. Section 3.5.4 provides empirical justification for the FGSM's discretization parameters used throughout. A thorough analysis of the vetting criteria is presented in Section 3.5.5 using both manual and automatically selected loop closure candidates. Finally, end-to-end system performance and examples are presented in Section 3.5.6.

3.5.1 Automatic Floor Segmentation Results

We tested the CRPMM based automatic floor segmentation algorithm in a variety of scenarios. First, we ran the proposed segmentation algorithm on the filtered pressure data readings from Fig. 3.5(b). We varied the choice of concentration parameter α and compared the results to determine the sensitivity of the CRPMM segmentation algorithm. Figure 3.13 shows the results of this experiment. When using $\alpha = \{0.5, 5, 100\}$ the CRPMM segmentation performed flawlessly. In order to test the limits of the algorithm, we pushed α to one million before the segmentation algorithm finally broke. This shows that the CRPMM is robust to choice of user defined concentration parameter α across many orders of magnitude.

Next, the CRPMM was tested against both supervised and unsupervised Gaussian Mixture algorithms to compare performance. The method of [7] uses heuristics to deduce the number of floors and then performs data clustering using the EM algorithm to fit a GMM to the data. This can be viewed as a two step process that first identifies the number of floors and then clusters the pressure readings. In order to perform the fairest comparison possible, we provide the number of floor clusters directly to the method of [7] and seed the EM algorithm using K-Means [100]. The second method we compare against is the unsupervised clustering algorithm presented by Figueiredo and Jain [58]. Figueiredo and Jain use the Minimum Message Length [101] (MML) criterion and the EM algorithm to simultaneously detect the number of components and cluster the data. We apply the method directly as presented for comparison purposes.

Figure 3.14 shows the results of the floor segmentation comparison tests. The dataset used for comparison was taken across 5 floors of an academic building. The filtered and zero centered pressure readings are shown in Fig. 3.14(a). Figure 3.14(b) shows the results of our CRPMM based segmentation algorithm. Notice that the correct number of floors were detected and the pressure data is segmented correctly. Figure 3.14(c) shows the results of applying the EM algorithm to fit a GMM model with the number of components known a priori. The EM algorithm incorrectly declares the bottom two clusters as a single floor and erroneously splits the middle segment. Lastly, Fig. 3.14(d) shows the results of the unsupervised clustering method of Figueiredo and Jain. For this example, the unsupervised method has over-segmented the data into 6 components instead of the optimal number of 5.

The final experiment using our automatic, CRPMM-based floor segmentation algorithm was to test its performance on data that contains systematic bias. Figure 3.15(a) shows the filtered pressure readings from a building spanning 7 stories. Notice that some of the pressure reading segments appear to have a linear ramp bias. Figure 3.15(b) shows the segmentation results. Since the data does not fit the underlying assumptions of the CRPMM, a few data points around the cluster boundaries have been misclassified. Example misclassifications are circled in red.

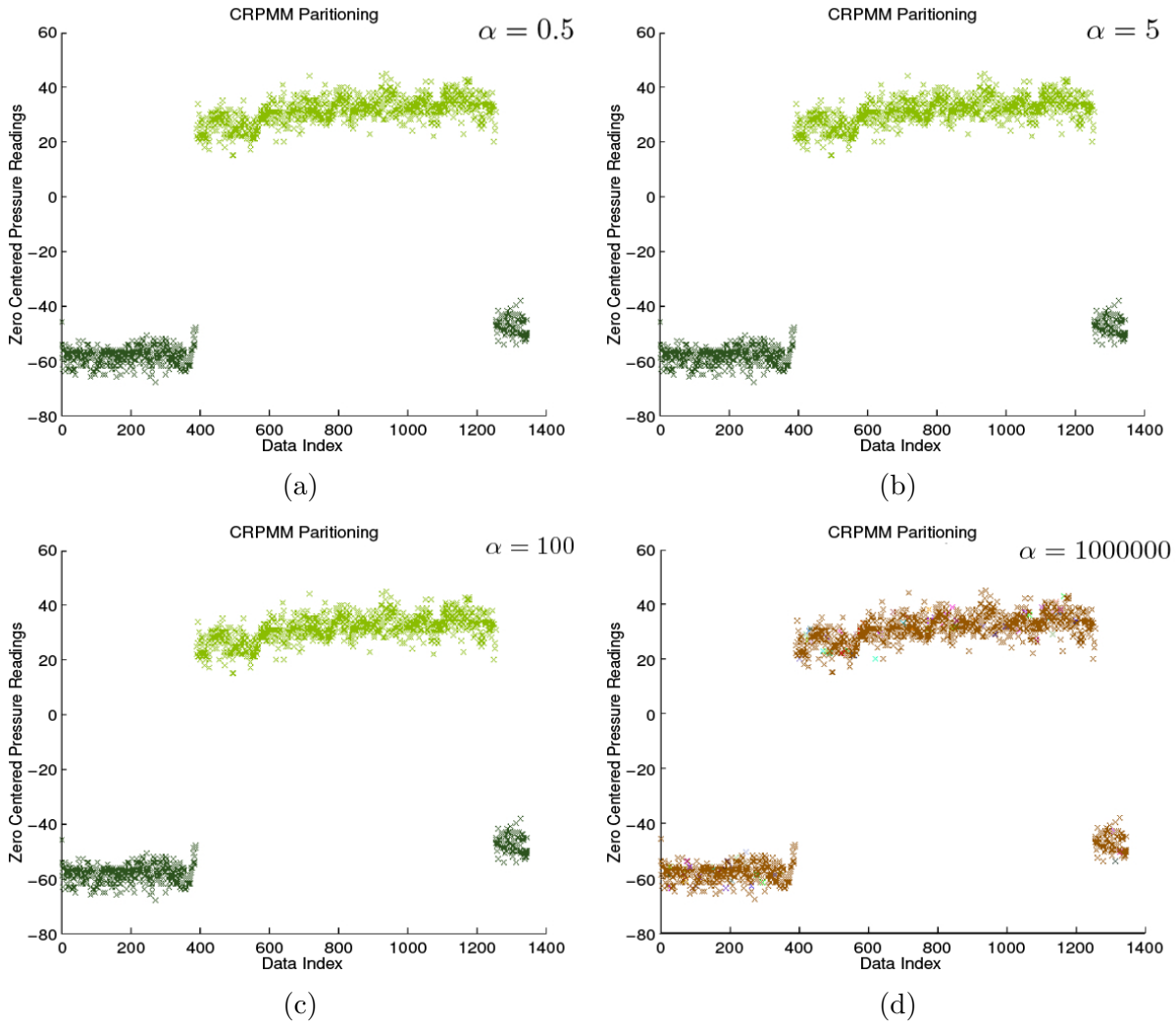


Figure 3.13: Sensitivity results for the Chinese Restaurant Process Mixture Model segmentation results. The concentration parameter α of the model was varied to test model sensitivity. Note that the segmentation results are stable across many orders of magnitude of α. (a): $\alpha = 0.5$ (b): $\alpha = 5$ (c): $\alpha = 100$ (d): $\alpha = 1000000$

3.5.2 Multi-Story Particle Filtering Results

In this section we present an example result for the multi-story extension to the standard Rao-Blackwellized particle filtering algorithm. A dataset was taken across two floors of an academic building connected by a single staircase. Pressure readings, IMU data, and laser data from the horizontally mounted scanner were collected. We then ran the CRPMM-based floor segmentation algorithm to partition the data based on what floor of the building it originated. The multi-story particle filter was then run to create a collection of geometrically consistent occupancy grid maps.

Figure 3.16 shows the results of applying the multi-story particle filtering algorithm using barometric floor segmentation. Figures 3.16(a) and 3.16(b) show the reconstructed occupancy grid maps for lower and upper levels of the building respectively. The red dots

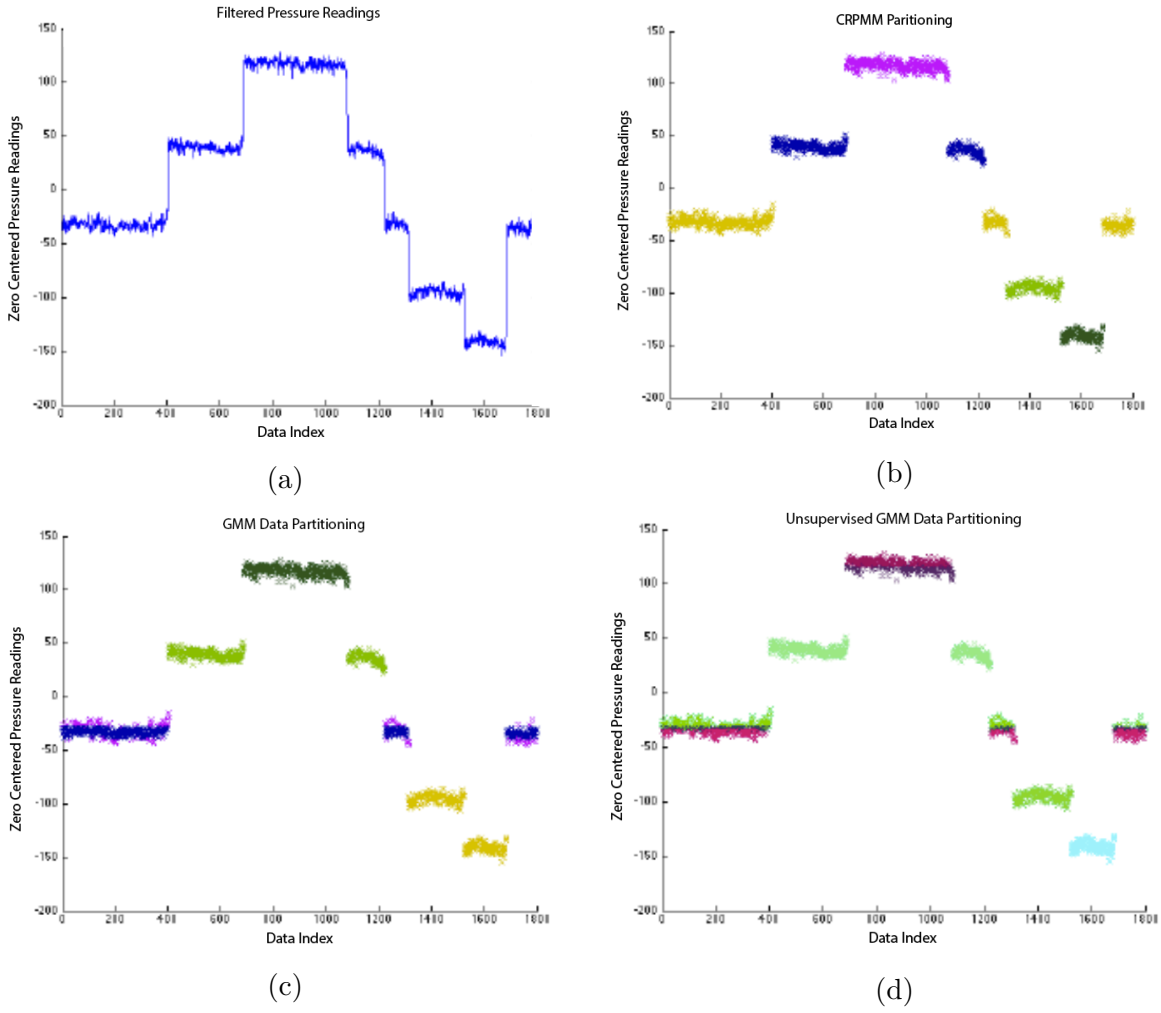


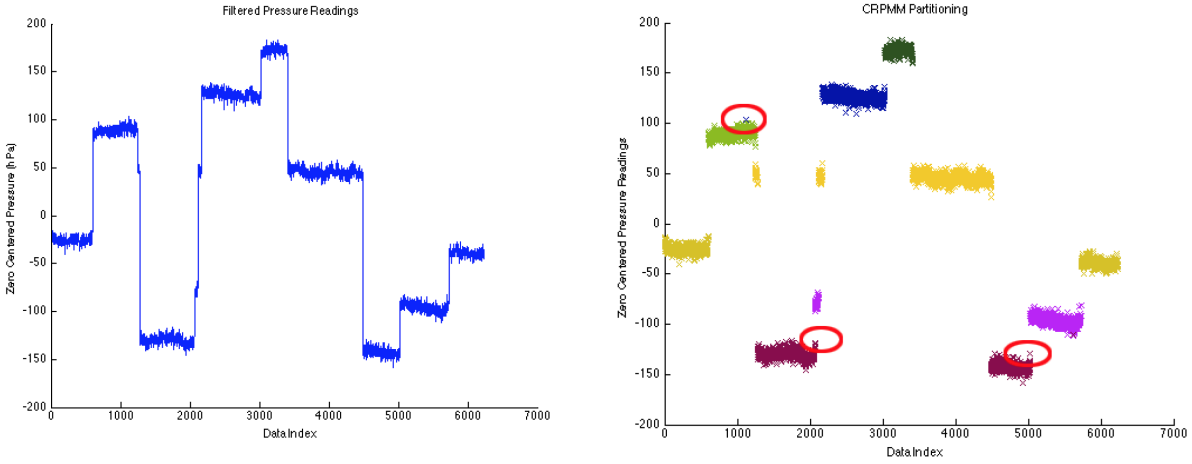
Figure 3.14: Barometric Floor Segmentation Comparison Results. (a): The filtered pressure readings from a dataset spanning 5 floors of a building. (b): The results when applying our CRPMM based segmentation algorithm. (c): The results when applying a GMM based model with the number of clusters known a priori. (d): The results of the unsupervised method of Figueiredo and Jain [58].

indicate the locations where the floor transition was correctly detected.

3.5.3 FGSM Performance Evaluation

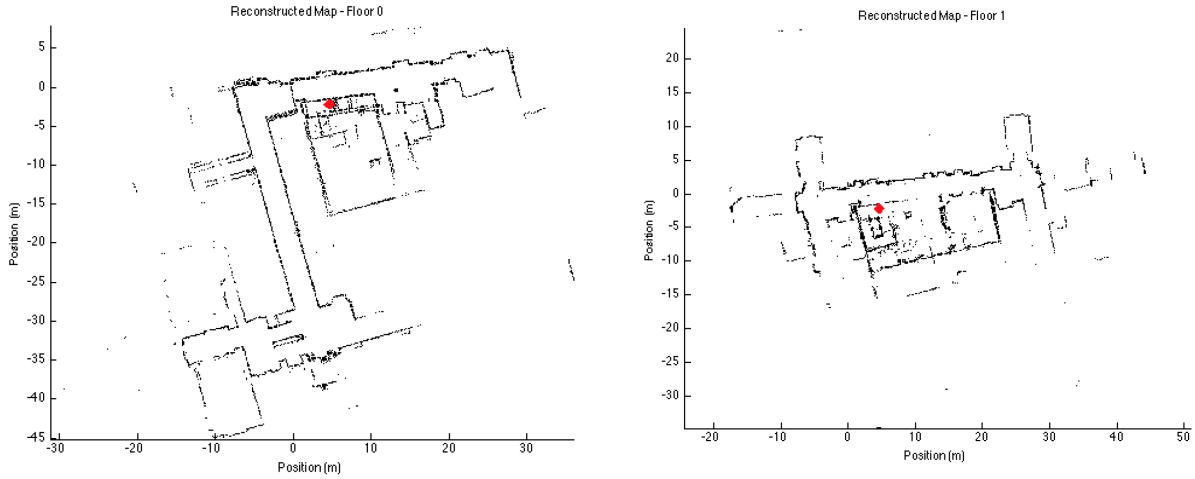
In this section we compare the proposed FGSM algorithm and a few state of the art alternatives. First, we compare against the open source libpointmatcher library [102] because it contains several implementations of state of the art ICP variants. In particular, we use the library's FICP implementation as it provided the best performance on our data. Secondly, we compare our proposed FGSM algorithm with the original hybrid genetic algorithm from which it was extended [59].

We construct the following experiment to quantify the performance of each of the three



(a) (b)

Figure 3.15: An example of the CRPMM segmentation algorithm on a dataset with systematic bias. (a): The filtered pressure readings. (b): The segmentation results. Notice that a few of the clusters contain a linear ramp bias. The resulting segmentation is prone to misclassification around the cluster boundaries. A few misclassified data points are circled in red.



(a) (b)

Figure 3.16: The results of the multi-story particle filtering algorithm using barometric floor segmentation. (a): The grid map reconstructed from the lower level. (b): The grid map reconstructed from the upper level. The red dot indicates the location where the floors transition was correctly detected.

chosen algorithms. We first hand-construct a set of 177 ground truth scan matching results taken from 10 different environments. Then, random initial conditions are created by selecting values from a collection of i.i.d. zero-mean Gaussian random variables with increasing

σ_T (m)	Libpointmatcher [102]	Hybrid-GSM [59]	FGSM
0.25	93%	90%	94%
0.5	89%	91%	95%
1.0	82%	89%	94%
2.0	68%	73%	91%
3.0	58%	59%	85%
5.0	45%	42%	75%
8.0	30%	36%	66%
10.0	25%	30%	62%

Table 3.1: A comparison of the proposed FGSM algorithm to previous methods in the case of translation only error. The percentage of trials that succeeded for various standard deviations in the translational component of the initial condition.

standard deviation. Ten unique trials are conducted for each scan pair resulting in 1,770 attempts for each standard deviation. Each algorithm is run and the resulting transforms are recorded. We consider any transform within 5 cm or 1 degree to have converged to the correct solution.

The first experiment assesses the algorithms' performance in the presence of translation only error. Using increasing levels of translation offset, ranging from 0.25 m to 10 m in standard deviation, we ran the experimental setup and computed the percentage of successful scan matches. Table 3.1 summarizes the results of this experiment. In situations where the error is small, the algorithms perform nearly identical. However, as the level of errors is increased, the accuracy rates of libpointmatcher and the hybrid genetic scam matching (hybrid-GSM) algorithms decay at a much faster rate than the proposed method.

In the second experiment we repeat the same procedure, but add only rotational offset using standard deviations ranging from 18° to 180°. Table 3.2 presents the results of this experiment. In contrast to the translation only scenario, the proposed algorithm outperforms both competitors by a wide margin. Surprisingly, the hybrid-GSM algorithm lags behind libpointmatcher by a considerable amount in this test. We suspect that this is because the hybrid-GSM algorithm does not model the presence of outliers data points.

In practical scenarios the error in initial condition is typically a mix of both rotation and translation. To characterize the effects of both error sources simultaneously, we repeat the experiment but this time add variations to both the rotational and translational components of the initial condition. As seen in Table 3.3, the algorithms perform similarly to the case of rotation only error. This result is not surprising because only the rotational variable appears as a non-linear in the object functions Jacobian. The results of these experiments show that the FGSM algorithm clearly outperforms comparable algorithms for data typical of our system.

3.5.4 Effect of Discretization on the FGSM Algorithm

As stated in Section 3.3.2, the amount of solution space discretization in the FGSM algorithm represents a trade-off between computational efficiency and accuracy. In order to determine

σ_θ (degrees)	Libpointmatcher [102]	Hybrid-GSM [59]	FGSM
18	74%	60%	92%
30	59%	44%	84%
45	45%	31%	73%
60	34%	24%	63%
90	25%	17%	49%
180	17%	11%	37%

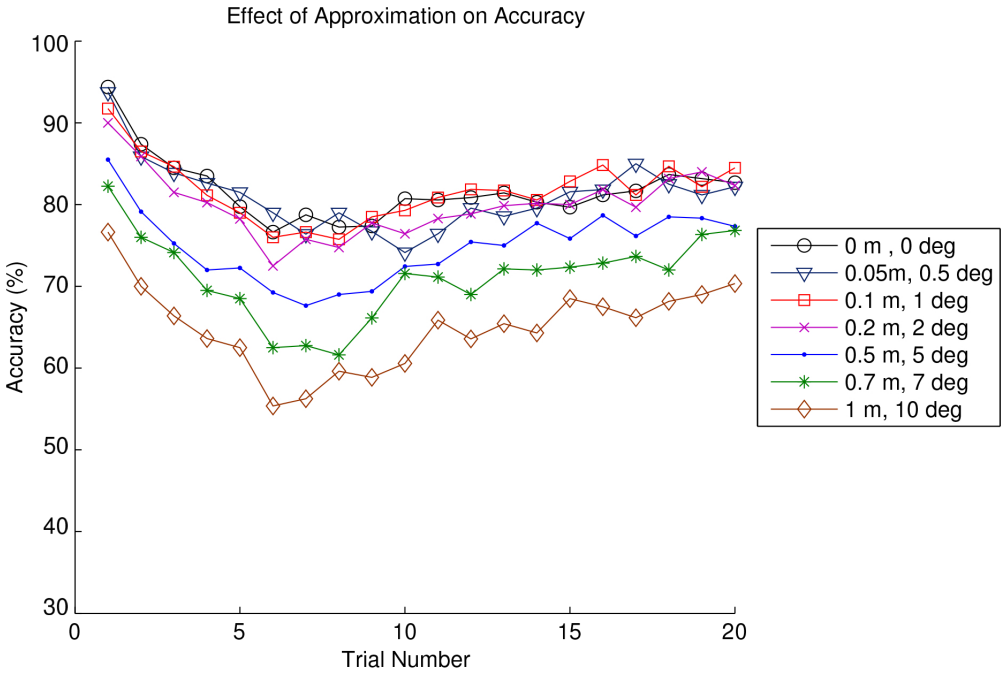
Table 3.2: A comparison of the proposed FGSM algorithm to previous methods in the case of rotational only error. The percentage of trials that succeeded for various standard deviations in the rotational component of the initial condition.

$(\sigma_T, \sigma_\theta)$ (m,°)	Libpointmatcher [102]	Hybrid-GSM [59]	FGSM
(0.25, 18)	74%	60%	91%
(0.5, 30)	55%	43%	84%
(1.0, 45)	42%	33%	74%
(2.0, 60)	28%	21%	61%
(3.0, 90)	18%	11%	45%
(5.0, 180)	9%	6%	33%

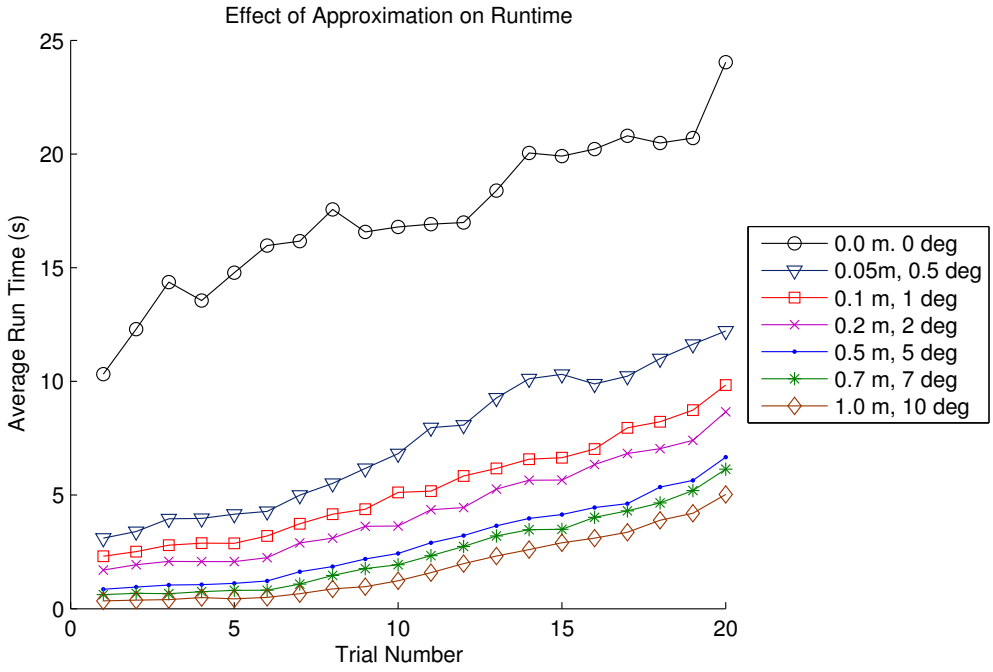
Table 3.3: A comparison of the proposed FGSM algorithm to previous methods in the case of both rotational and translational error. The percentage of trials that succeeded for various standard deviations in the rotational and translational components of the initial condition.

an appropriate level of discretization, we conducted the following test. We selected seven levels of discretization, ranging from no discretization to 1 m and 10 degrees, and evaluated the performance of the FGSM at increasing amounts of error in the algorithm's initial condition. For each chosen level of discretization and error in initial condition, we conducted a trial of one thousand scan matches drawn from a database of one hundred scan pairs. For each scan match we used two hundred initial chromosomes and computed the accuracy and running time by comparing the FGSM results to the manually defined ground truth alignment.

Figure 3.17 shows the results of this experiment. Fig. 3.17(a) shows the effect of discretization on accuracy while Fig. 3.17(b) shows the corresponding effect on run time. The baseline, shown in black, corresponds to no approximation. As the solution space discretization is increased, both the average runtime and accuracy decreases. For specific discretization parameters of 0.1 meters and 1 degree, the runtime is decreased by a factor of 3.25 without sacrificing a significant amount of accuracy compared to the baseline algorithm. Furthermore, Fig. 3.17(b) shows that the FGSM algorithm's running time is correlated with the amount of error in initial condition. This result is expected because as the error in initial condition increases, the algorithm must explore a larger solution space and thus it converges at a slower rate.



(a)



(b)

Figure 3.17: The effect of solution space discretization on the FGSM algorithm for increasing levels of error in initial condition. (a): Accuracy computed for different amounts of solution space discretization. (b): The average runtime of the FGSM algorithm for different levels of discretization.

3.5.5 Loop Closure Verification

In order to characterize the effectiveness of the proposed metrics for loop closure verification, a database of 15 datasets has been collected from 10 unique environments. The classification metrics are evaluated in the following two scenarios. First, between 50–100 true loop closures are manually defined for each dataset resulting in 590 positive examples. To simulate the presence of falsely identified loop closures, 10 loop closures are randomly defined for each dataset yielding 150 potentially erroneous examples. The FGSM algorithm is then run on each loop closure candidate and the resulting transformations were then manually inspected and labeled.

Figure 3.18(a) shows a scatter plot of the results. The candidates for which the FGSM algorithm correctly identifies the transformation is shown using a green $\circ$, while the incorrect transforms are shown via a red $\times$. The two regions form a near disjoint partition of the graph suggesting the use of a threshold based rule. Specifically, transformations are defined to be correct if the ratio of eigenvalues exceeds threshold r^R and the correlation metric exceeds threshold c.

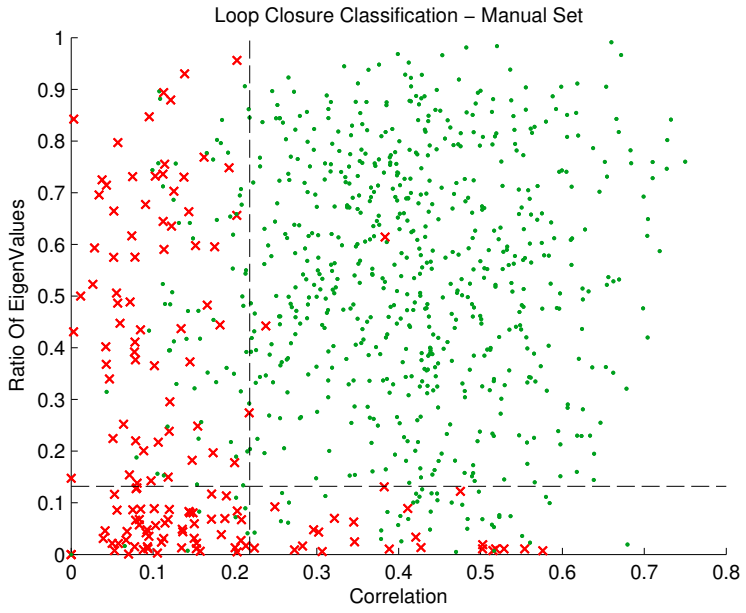
Statistics for various thresholds are computed and the resulting receiver operating characteristic (ROC) curve is shown in Fig. 3.19. When the false positive rate is required to be $\leq$1%, the corresponding true positive rate is 84.7% for thresholds of $r^R > 0.132$ and $c > 0.207$. The computed area under the ROC curve was found to be 94.2%.

Since the existing work on loop closure verification [47] only report statistics for detection, estimation, and validation combined, a one-to-one comparison is not possible. While a direct comparison cannot be made, the authors rely on a correlation-like technique to validate loop closure transformations. To demonstrate the performance of a correlation only scheme in indoor environments, we perform the loop closure vetting again using only the correlation metric c of Eq. 3.22. Figure 3.19 shows the resulting ROC curves. When the false alarm rate is required to be $\leq$ 1%, we obtain true positive rates of 16.9% and 13.5% for the manual and automatic loop closure sets respectively. This empirically shows that a correlation metric is insufficient for robust operation in indoor environments.

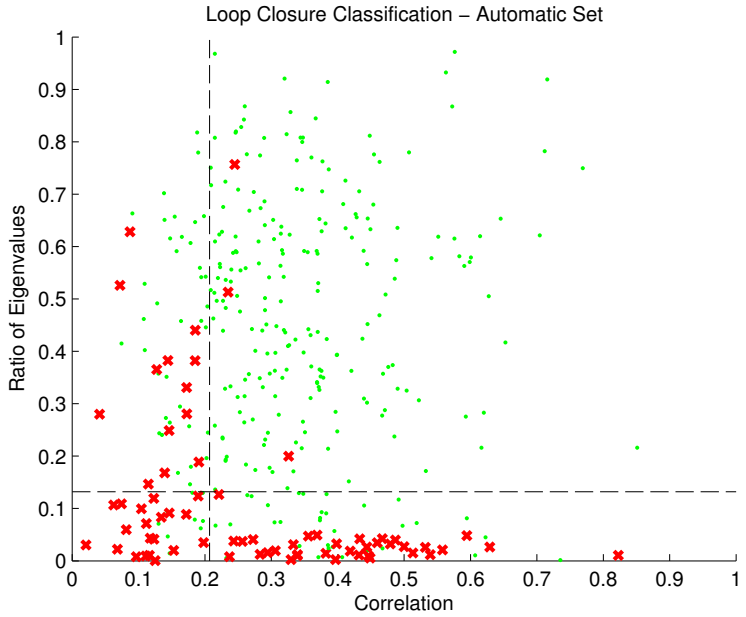
The same test is repeated, but this time using automatically detected loop closures via the method of Section 3.3.1. Using 5 new datasets, 361 examples are automatically detected from grid map data. The FGSM algorithm is run and the results are manually inspected and labeled. Figure 3.18(b) presents the results of using the proposed classifier on the automatically detected loop closures. The distribution of points in the plot supports the use of a threshold based rule. Using the thresholds derived from the manually defined set of loop closures, a true positive rate of 72.3% and a false positive rate of 4.3% is obtained. This analysis empirically shows that the false alarm rate and optimal thresholds are only loosely dependent on the source of loop closure candidates.

3.5.6 End-To-End System Results

We evaluated the end-to-end system performance using the following experiment. We collected data along a 350 meter trajectory from an office building with approximately 1,500 square meters of floor space. The office environment contained 100 pre-surveyed control points each denoted using a paper target containing a checkerboard-like pattern of known



(a)



(b)

Figure 3.18: A scatter plot showing the distribution of loop closure candidates using the proposed metrics. A red × corresponds to a failed loop closure transformation, while a green ∘ represents a successful candidate pair. The chosen thresholds are denoted using the dashed black lines. (a): The results for the manually defined loop closure candidates. (b): The results for the automatically detect set of loop closures.

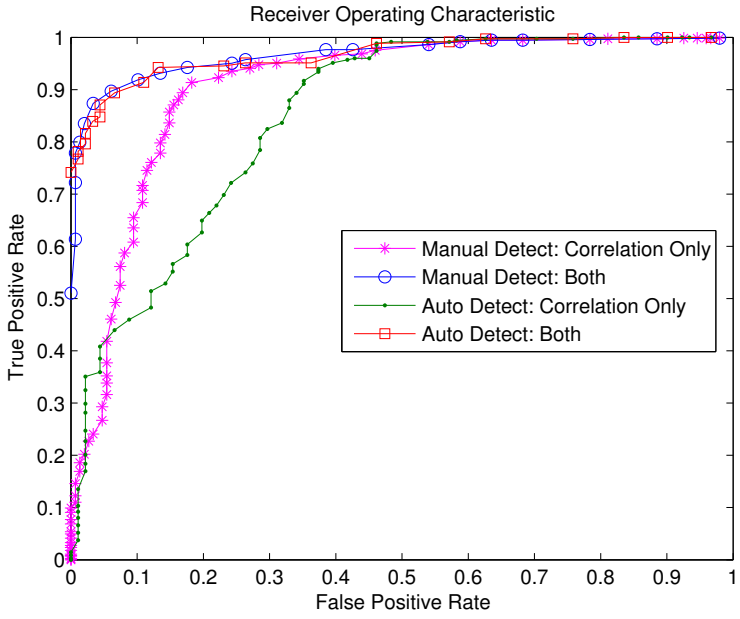


Figure 3.19: Comparison of receiver operating characteristic curves when using both metrics *versus* only correlation.

size. Using a Leica HDS6000 and C10 Scanstation, 17 high-density static scans were automatically stitched together to provide millimeter level accuracy for each survey point. We then localized the system using both the proposed algorithm and our previous approach [79]. We estimated the location of each control point by projecting the temporally closest laser readings into the optical imagery and then manually selecting all laser points that landed on the paper target. We then fit a plane to these points using Principal Component Analysis [103] to determine the precise (x, y, z) location of the control point. The estimated locations were then compared to the ground truth locations in order to characterize the accuracy of each method.

Figure 3.20 shows the results of the control point experiment. Figure 3.20(a) shows the locations of the control points, shown in red, overlain on the reconstructed geometry. A well trained human engineer tediously spent approximately 10 h manually selecting, estimating, and vetting 150 loop closure constraints in order to apply the method of [79]. Figure 3.20(b) shows the resulting histogram of errors for the previous method with a maximum reported error of 27.66 cm a mean error of 9.91 cm, and standard deviation of 5.30 cm amongst all control points. The histogram of errors for the proposed method is shown in Figure 3.20(c). The proposed method utilizes only 63 loop closures, yet still attains a maximum error of 27.73 cm, a mean error of 10.66 cm, and a standard deviation of 5.49 cm. Figure 3.20(d) and 3.20(e) show the locations of the loop closure constraints used for both the previous and proposed methods. Despite automatically generating fewer loop closure constraints, the new method is able to match the performance of the previous method which required manual intervention.

To further demonstrate the end-to-end performance of the backpack system, we have applied the proposed algorithms to three more datasets of increasing complexity. Fig-

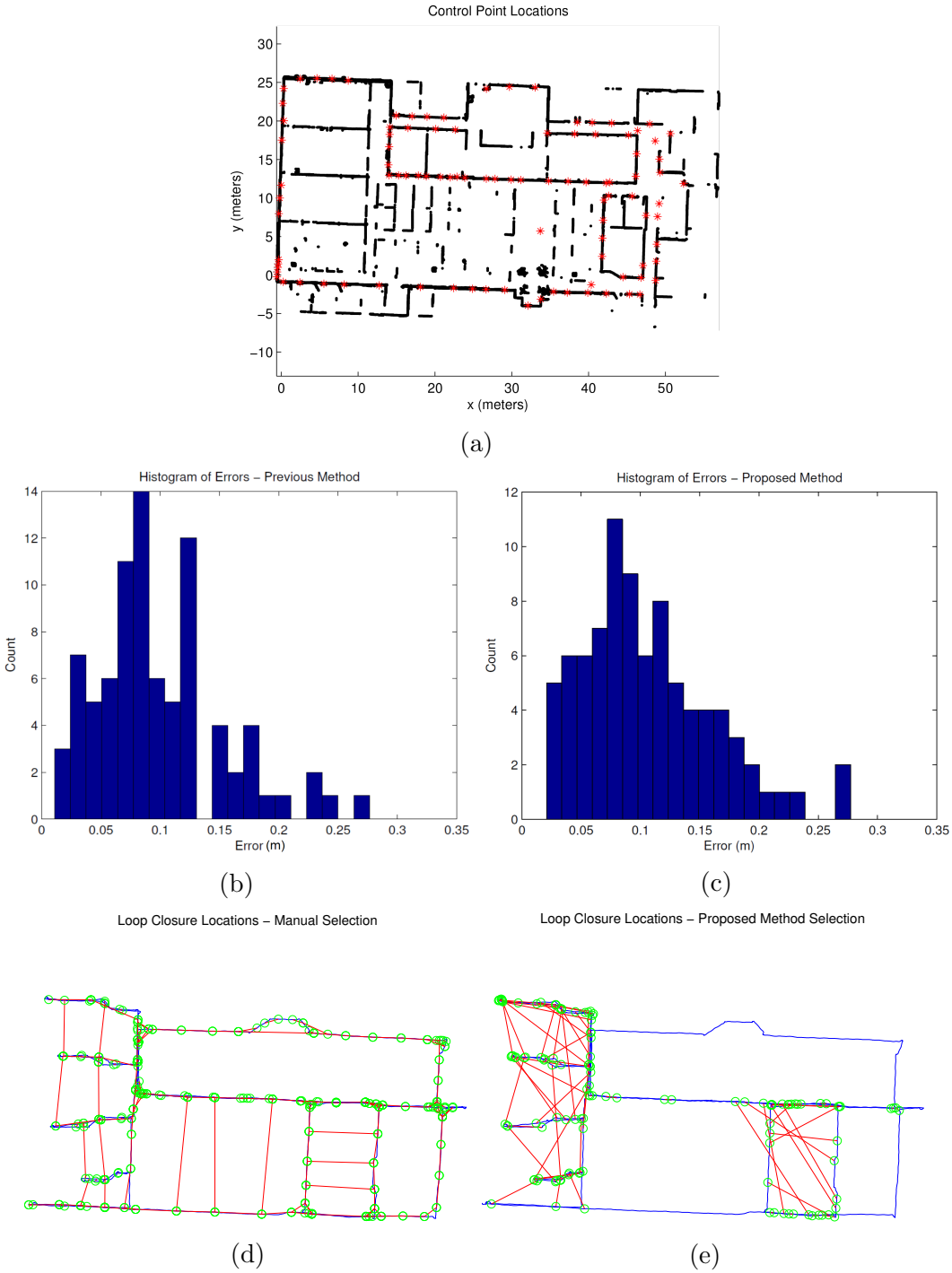


Figure 3.20: Results from the control point experiment. (a): The locations of the control points, shown in red, overlain on the reconstructed map. (b): Histogram of the errors in estimating the control points for the previous method. (c): Histogram of the errors in estimating the control points for proposed method. (d): The locations of the manually selected loop closure constraints, shown by green circles connected by red lines, used when applying the previous method. (e): The loop closure locations automatically selected via the proposed method.

Subtask	Total Time (min)	Per Minute Time (sec)
Dead Reckoning	3	4.5
Floor Segmentation	1	1.5
Submap Generation	50	75.0
RBPF Grid Mapping	60	90.0
Loop Closure Detection	10	15.0
Transform Estimation	2	3.0
Verification	1	1.5
Graph Optimization	2	3.0
Height Estimation	10	15
3D Path Generation	1	1.5
Total	139	208.5

Table 3.4: A rough estimate of the running time for the various stages of the presented algorithm for a 40 min data collection.

ure 3.21(a)–(c) show a simple dataset taken from a 250 m single loop on one floor of an academic building. The resulting occupancy grid map is shown in Figure 3.21(a). Global constraints have been extracted from the grid map and the FGSM algorithm is run to compute the transformation between loop closure locations. The trajectory after optimization is shown in Figure 3.21(c).

The second set, shown in Figure 3.21(d)–(f) is a longer dataset taken from an entire floor of an academic building containing a few inner loops. Totaling 22 min, the operator walked for 750 m through the environment. Figure 3.21(e) shows the dead reckoning trajectory with the 80 accepted loop closures overlain. The optimized, geometrically consistent trajectory is shown in Figure 3.21(f).

The final dataset is the most challenging because the trajectory contained many interior loops and a large portion of the surrounding geometry did not fit the vertical wall assumption made in Section 3.2.1. Taken from a warehouse-sized retail shopping center, the scanned area exceeded 5,820 square meters. The operator traversed the environment for 31 min and covered a total distance of 1,500 m. Despite the large amount of accumulated bias present in Figure 3.21(h), the 59 accepted loop closure constraints extracted from the occupancy grid map enforce geometric consistency in the final optimized trajectory. This result shows that our proposed algorithms are able to scale to buildings of considerable size.

3.5.7 Timing Results

Although the proposed algorithms were designed to be run off-line, it is important to highlight the approximate running time for the various stages of the algorithms. Table 3.4 lists the approximate running time for the various subtasks using unoptimized code on a single core of an 2.4 GHz Intel Core i7 laptop. For the 40 min data collection taken in the warehouse-sized retail shopping center, the approximate running time of all stages was 139 min. Based on the results of Table 3.4, we note that the running time is dominated by the

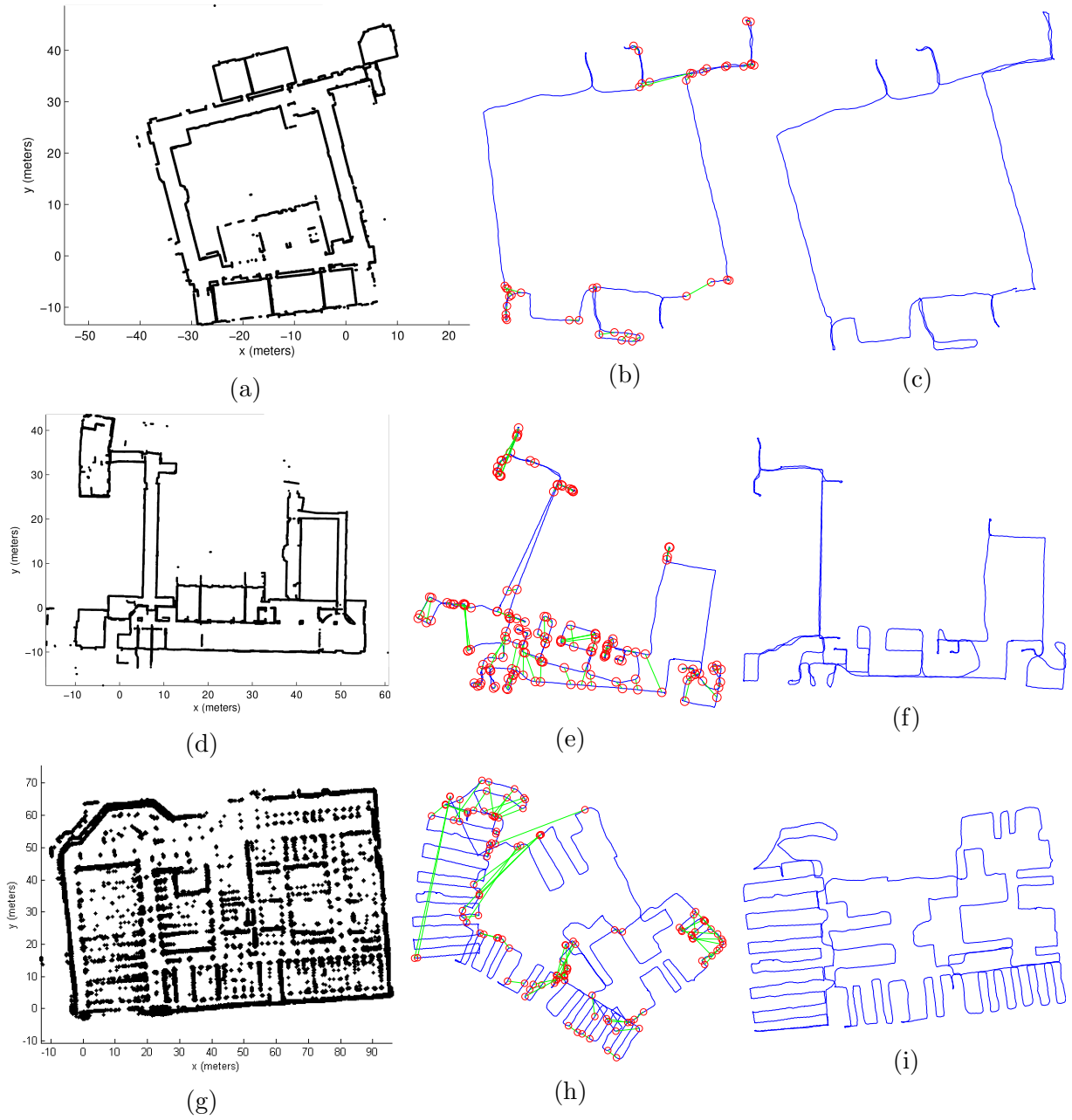


Figure 3.21: Results of applying the end-to-end system. (a),(d),(e): The occupancy grid maps that result from the RBPF algorithm. (b),(e),(h): The dead reckoning trajectories with validated loop closure constraints overlain. (c),(f),(i): The 3D paths viewed from the top down after optimization has been applied.

submap generation and grid mapping step of the algorithm. This result is easily understood due to the fact that both stages require the Monte Carlo based RBPF.

The number of particles used in the RBPF represents a trade-off between accuracy and computation time. Particle filters require sampling a discrete set of samples from a continuous posterior distribution and thus as the number of particles increases the posterior is estimated more accurately. Since each particle must have its contributions computed independently, the running time also increases linearly with the number of particles used by the RBPF. Since the 2.5D localizations algorithms are run in an off-line, we typically use an overabundance of particles in the RBPFs for maximal accuracy. For a large-scale dataset the number of particles utilized is typically on the order of 1000.

In addition to the RBPF based sections of the algorithm, loop closure detection also takes a substantial portion of the runtime. In particular, the optimization of the correlation matrix is the most computationally expensive portion of loop closure detection. For a cluster of N poses, correlation matrix optimization requires on the order of N^2 applications of FICP. For clusters involving approximately 100 poses, the time required to optimize the clusters' correlation matrices is approximately 50 seconds per cluster.

Although the algorithms were designed to be run off-line, several opportunities exist to make the algorithms more suitable for real-time operation. Since the particles in the particle filter are computed independently, the RBPF is trivially parallelization across multiple threads. Furthermore, when performing optimization of the cluster's covariance matrix each application of FICP is completely independent and thus will trivially scale with the number of threads used for computation.

3.5.8 Incorporating Control Points Results

In order to test the effect of the graph-based control point algorithm of Section 3.4.3 on the accuracy of the reconstructed trajectory, we conducted the following experiment. Starting with the dataset and control points of Section 3.5.6, we chose a small subset of the 100 total control points and used them as landmarks during graph optimization. The remaining unused control points were used to characterize the error in the trajectory in a cross-validation framework. Since we had pre-surveyed ground truth estimates of all control points in this dataset, the estimates of the unused control points were compared against the ground truth to produce an error metric for each point. We then varied the number of control points being utilized to quantify the effect of the number of control points on the system's accuracy.

Figure 3.22 shows the results of incorporating control points on the accuracy of the reconstructed trajectory. The mean alignment error of all remaining control points and the 95% confidence interval were estimated by computing the results over 100 trials of a bootstrapped sampling of the data [104]. By inspecting Fig. 3.22, we find that for a small number of control points there is no statistically significant change in the mean control point alignment error. However, as the number of utilized control points increases, the mean alignment error is decreased and the 95% confidence interval shrinks. By utilizing around 90 control points as landmarks, the mean alignment error drops to around 8 cm. This is a 50% decrease over the baseline case of utilizing no control points.

In practice, the landmarks must be specified in the graph optimization problem with some initial covariance to model the uncertainty in their positions. The initial uncertainty

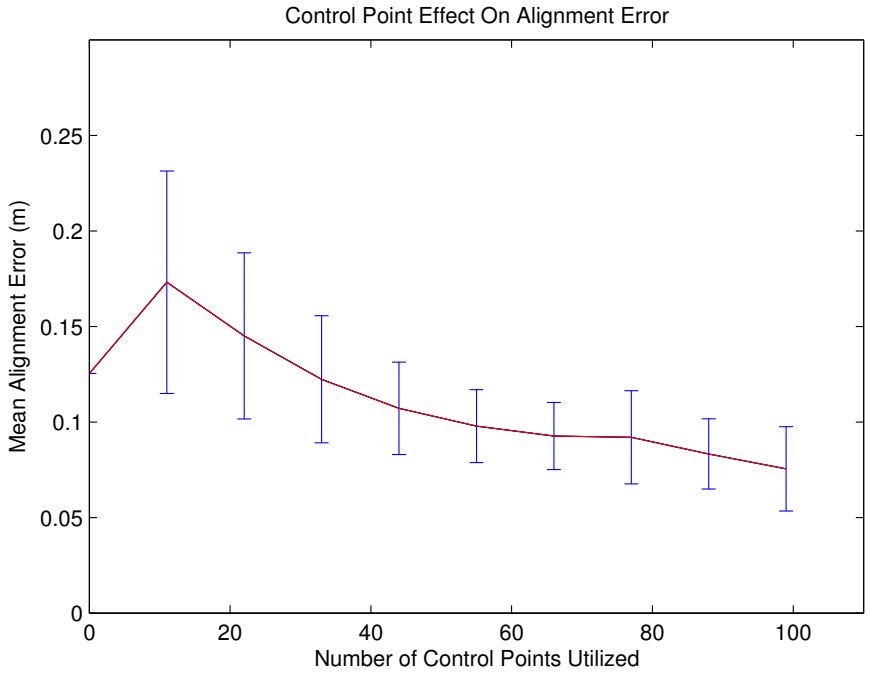


Figure 3.22: The effect of incorporating control points on the accuracy of the reconstructed trajectory. Shown in red is the mean error in control point alignment. The blue bounds indicate the 95% confidence intervals. As control point utilization increases, the mean alignment error decreases.

of the landmarks was chosen to match the noise specifications of the static scanning technology. For the scanning technology utilized for this experiment, the standard deviation of the uncertainty was set to 0.5 mm.

Chapter 4

3D Localization Algorithms

In this Chapter, we present a sensor fusion algorithm for direct 3D localization and mapping of GPS-denied indoor environment in order to overcome a number of major drawbacks of the 2.5D methods presented in Chapter 3. In contrast to the 2.5D localization algorithm that combined data from multiple sensors in an ad hoc fashion, the algorithm presented in this Chapter reformulates the localization and mapping problem using a tightly-coupled Extended Kalman Filter (EKF) estimator. In doing so, it provides a number of key advantages over the 2.5D methods presented in Chapter 3. First, there is only a loose coupling of data between the sensors. Generally, 2.5D methods decouple the localization problem into a 2D localization problem followed by a height estimation problem. In doing so, data from the laser scanners and IMU are combined in an ad hoc method to create a full 6DOF trajectory. Furthermore, data from the IMU is trusted and used without any external feedback or batch optimization. This means that any errors in the IMU data will directly affect the final trajectory and cause degradation in the final model.

Secondly, the data from the camera is completely neglected in the localization process. Camera sensors are capable of providing valuable feedback during the mapping process. By decoupling the localization problem into a 2D localization problem followed by a height estimation problem, the methods in Chapter 3 is unable to easily incorporate camera imagery.

The final drawback with the 2.5D approach from Chapter 3 is that it not capable of calibrating the various intrinsic, extrinsic, and temporal calibration parameters of the system. Since data from individual sensors are captured in their own coordinate systems, accurate calibration and synchronization between sensors is essential for accurate mapping.

We address these shortcomings by proposing a tightly-coupled EKF estimator that fuses data from all available sensors into a single trajectory. By formulating the SLAM problem as an EKF estimator, data from individual sensors are causally fused together to estimate not only the position of the system and a low dimensional planar representation of the surrounding environment, but also an optimized estimate of the system's calibration parameters.

4.1 Algorithm Overview

Similarly to the methods of Chapter 3, a two-pass approach is used. First, data from all sensors is fused into an open-loop estimate of the system's trajectory. Then, a graph opti-

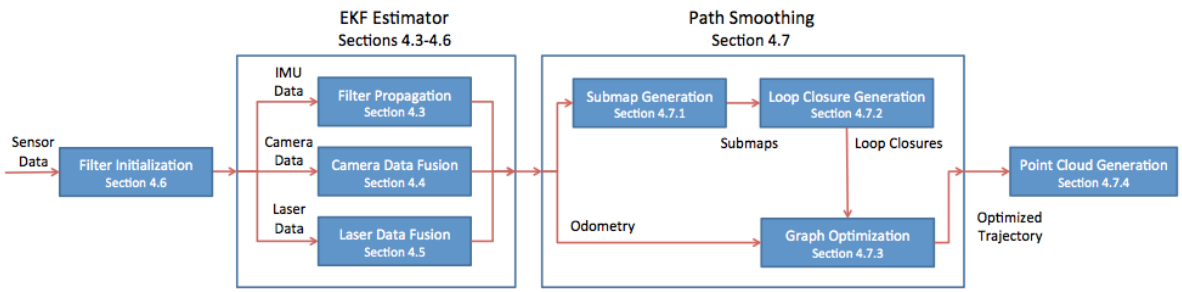


Figure 4.1: The block diagram of the 3D localization and online calibration method presented in this chapter.

mization post-processing step is applied to correct for any accumulated drift and create an optimized system trajectory. The final system trajectory is then used in creating a high-quality 3D point-cloud representation of the environment using the methods presented in Section 4.8.4.

The main contributions of this chapter are threefold. First, we propose a tightly-coupled EKF estimator that fuses data from all available sensors into a single optimized trajectory. By fusing data from both laser and camera sensors, we show that we are able to more accurately estimate the trajectory of the system than if we used only a single sensing modality. Secondly, by formulating the SLAM problem as an EKF estimator, data from individual sensors are causally fused together to estimate not only the position of the system and a low dimensional planar representation of the surrounding environment, but also an optimized estimate of the system's calibration parameters. To our best knowledge this work presents the first direct online extrinsic and temporal calibration between a 2D laser scanner and an inertial sensor that considers the rolling shutter effect of the laser scanner. Lastly, we allow vertical planes of arbitrary orientation. A direct consequence of this is that our algorithm does not require the local coordinate system to be axis aligned as in [61]. Furthermore, this allows the algorithm to be used in a wider variety of indoor environments even when the map is not known a priori.

Data fusion using multiple separate data sources requires a modular approach. We formulate the EKF-SLAM algorithm by queuing the data according to the timestamp reported by the sensor. In doing so, the estimator is agnostic to differences in sampling frequency between the sensors. Furthermore, by processing the data in this fashion, the EKF is capable of flexibly processing data from an arbitrary number of lasers with no modifications to the underlying algorithm. A well-known limitation of EKF-SLAM is that the size of the state vector ultimately limits the number of map features that can be tracked. We address this problem in two ways. First we limit the number of structural elements tracked by the EKF using a set of heuristics. In traditional EKF-SLAM this strategy leads to drift in the estimated trajectory. In order to eliminate any accumulated drift, we also apply an incremental graph optimization as the final step.

Algorithm 3 and Figure 4.1 provide an overview of the algorithms presented in this Chapter. As shown in Figure 4.1, the algorithm begins by first processing the raw sensor data to produce open-loop odometry via an EKF estimator. Each time a new data reading is available to the EKF estimator, the appropriate update function is called. If the data is from

Algorithm 3 Multi-sensor Data Fusion Algorithm Overview

```
1: while has_data_to_process() do
2:     data = read_data()
3:     if data.type == INERTIAL then
4:         propagate_imu_state(data)
5:     else if data.type == LASER then
6:         perform_laser_ekf_update(data)
7:     else if data.type == IMAGERY then
8:         perform_camera_ekf_update(data)
9:     end if
10: end while
11: while has_submaps_to_process() do
12:     add_odometry_to_graph()
13:     if loop_closure_found() then
14:         constraint = compute_transform_using_icp()
15:         update_incremental_graph_optimization(constraint)
16:     end if
17: end while
```

the IMU, then it is used to propagate the state and its covariance forward by integrating the gyroscope and accelerometer measurements. If the data originated from a camera, then the image is used to perform an EKF update according to the MSCKF algorithm [105]. For laser scans, the EKF extracts line features from the raw range data and matches them against planes stored in the EKF's map using our proposed laser processing algorithm. In this way, the EKF estimator processes the data readings in a causal and modular way.

Once the data is processed using the EKF, we perform a path smoothing step to mitigate any accumulated drift. Specifically, we assemble the laser scans into small subsections of the environment. Termed submaps, the small sections are then matched against each other using an ICP variant to create constraints for an incremental graph optimization procedure. The incremental graph optimization results in an optimized trajectory that is then used to assemble all laser data readings into a dense point cloud representation of the environment.

The outline of this section is as follows. Section 4.2 defines the list of symbols used through this chapter. Section 4.3 describes the state space representation of our EKF estimator. Section 4.4 details how IMU data is used in the Kalman filter's propagation step. Next, the processes by which the image and laser data is utilized by the EKF estimator is detailed in Sections 4.5 and 4.6 respectively. Section 4.8 then describes the iterative graph optimization procedure used for mitigating accumulated drift during odometry estimation. Finally, Sections 4.9 and 4.10.1 contain the results and limitations of the proposed algorithms.

4.2 Notation and Table of Symbols

This section defines the symbols used throughout this chapter. The conventions for the notation used for scalars, vectors, and matrices is repeated here from Section 2.1.

4.2.1 General Notation

Symbol Notation	**Example**
Scalars are denoted in lower case	x
Vectors are denoted in bold lower case	$\mathbf{x}$
Matrices are denoted in bold upper case	$\mathbf{X}$
Cross product of a vector uses $\lfloor \cdot \times \rfloor$	$\lfloor \boldsymbol{\omega} \times \rfloor$
Quaternion product is denoted as $\otimes$	$\mathbf{q}_1 \otimes \mathbf{q}_2$
Time derivative is represented by a dot	$\dot{\mathbf{p}} = \mathbf{v}$
Jacobian of function $f(\cdot)$ by vector $\mathbf{x}$	$\mathbf{J}_{\mathbf{x}}\big(f(\cdot)\big)$ or $\mathbf{H}_{\mathbf{x}}$ or $\boldsymbol{\Gamma}_{\mathbf{x}}$

Estimation Notation	**Example**
An estimated value is denoted by a hat	$\hat{\mathbf{x}}$
A tilde denotes an estimation error	$\tilde{\mathbf{x}}$
Rotation error is denoted using $\tilde{\boldsymbol{\theta}}$ or δ	$\tilde{\boldsymbol{\theta}}^G$ or $\delta\mathbf{q}$

Coordinate Notation	**Example**
Coordinate frames are denoted in upper case	G or $\{G\}$
A rotation from $\{G\}$ to $\{B\}$	${}^B_G\mathbf{R}$ or ${}^B\mathbf{q}_G$
The coordinate frame of a point $\mathbf{p}_B$ in $\{G\}$	${}^G\mathbf{p}_B$

4.2.2 Symbol Definitions

α_i Angle coordinate of the i^{th} laser point.

β_θ Plane duplication angle threshold.

β_d Plane duplication offset threshold.

$\mathbf{\Delta x}$ State update vector.

$\mathbf{\Gamma}_l$ The stacked laser Jacobian with respect to the laser feature observations.

$\mathbf{\Gamma}_{\theta j}$ The Jacobian of $\mathbf{y}$ with respect to the j^{th} line feature.

$\mathbf{\Gamma}_{aug}$ The augmentation Jacobian with respect to the laser feature observations.

$\mathbf{\Gamma}_{j\mathbf{g}}$ The Jacobian of Eq 4.107 with respect to the j^{th} line feature.

$\mathbf{\Gamma}_{l_{1j}}$ The Jacobian of $\mathbf{r}_{l_{1j}}$ with respect to the laser line feature $\mathbf{u}_j$.

$\mathbf{\Gamma}_{l_{2j}}$ The Jacobian of $\mathbf{r}_{l_{2j}}$ with respect to the laser line feature $\mathbf{u}_j$.

$\mathbf{\Gamma}_{d_H j}$ The Jacobian of the horizontal offset parameter with respect to the j^{th} line feature $\mathbf{u}_j$.

$\boldsymbol{\omega}$ A rotational velocity vector.

$\boldsymbol{\omega}_m$ The measured rotational velocity vector.

$\mathbf{\Phi}$ State transition matrix.

$\mathbf{\Pi}_i$ The i^{th} plane.

$\boldsymbol{\pi}_i$ The normal vector for the i^{th} plane.

$\mathbf{\Sigma}$ Covariance matrix.

$\mathbf{\Sigma}_{p_i}$ Covariance matrix of the i^{th} plane.

$\boldsymbol{l}_{j\perp}$ The out-of-plane intersection line of the j^{th} laser feature observation.

Δt Timestamp difference.

$\delta\boldsymbol{\theta}$ An error in a quaternion.

$\delta\boldsymbol{\theta}^G$ An error in a global quaternion.

γ Mahalanobis distance.

$\hat{\mathbf{T}}$ Transformation.

$\hat{\mathbf{x}}$ An estimate of quantity $\mathbf{x}$.

$\mathbb{R}^3$ The set of 3D real numbers.

$\mathbf{a}$ An acceleration vector.

$\mathbf{a}_m$ The measured acceleration vector.

$\mathbf{b}_a$ The time-varying bias of the accelerometer.

$\mathbf{b}_g$ The time-varying bias of the gyroscope.

$\mathbf{e}_z$ Elementary basis vector for the z-axis.

$\mathbf{f}$ The focal length parameters of the camera.

$\mathbf{G}_c$ IMU noise transfer function matrix.

$\mathbf{h}(\cdot)$ Camera projection function.

$\mathbf{H}_\mathbf{x}$ General notation for a Jacobian of a function with respect to states $\mathbf{x}$.

$\mathbf{H}_l$ The stacked laser Jacobian with respect to the state vector $\mathbf{x}$.

$\mathbf{H}_{\mathbf{g}B}$ The Jacobian of Eq 4.107 with respect to the IMU body state $\mathbf{x}_B$.

$\mathbf{H}_{\mathbf{g}L}$ The Jacobian of Eq 4.107 with respect to the laser calibration state $\mathbf{x}_L$.

$\mathbf{H}_{\pi_i}$ The Jacobian of the camera observation function with respect to the i^{th} camera pose.

$\mathbf{H}_{\theta B}$ The Jacobian of $\mathbf{y}$ with respect to the IMU body state $\mathbf{x}_B$.

$\mathbf{H}_{\theta L}$ The Jacobian of $\mathbf{y}$ with respect to the laser calibration state $\mathbf{x}_L$.

$\mathbf{H}_{aug}$ The augmentation Jacobian with respect to the state vector $\mathbf{x}$.

$\mathbf{H}_{c_i}$ The Jacobian of the camera observation function with respect to the camera calibration.

$\mathbf{H}_{f_j}$ Stacked Jacobian of the camera observation of the j^{th} point with respect to the j^{th} point.

$\mathbf{H}_{f_{ij}}$ The Jacobian of the camera observation function with respect to the j^{th} feature point.

$\mathbf{H}_{l_{1j}B}$ The Jacobian of $\mathbf{r}_{l_{1j}}$ with respect to the IMU body state $\mathbf{x}_B$.

$\mathbf{H}_{l_{1j}L}$ The Jacobian of $\mathbf{r}_{l_{1j}}$ with respect to the laser calibration state $\mathbf{x}_l$.

$\mathbf{H}_{l_{1j}P_i}$ The Jacobian of $\mathbf{r}_{l_{1j}}$ with respect to the laser calibration state $\mathbf{x}_{p_i}$.

$\mathbf{H}_{l_{1j}}$ The Jacobian of $\mathbf{r}_{l_{1j}}$ with respect to the state vector $\mathbf{x}$.

$\mathbf{H}_{l_{2j}B}$ The Jacobian of $\mathbf{r}_{l_{2j}}$ with respect to the IMU body state $\mathbf{x}_B$.

$\mathbf{H}_{l_{2j}L}$ The Jacobian of $\mathbf{r}_{l_{2j}}$ with respect to the laser calibration state $\mathbf{x}_l$.

$\mathbf{H}_{l_{2j}P_i}$ The Jacobian of $\mathbf{r}_{l_{2j}}$ with respect to the laser calibration state $\mathbf{x}_{p_i}$.

$\mathbf{H}_{l_{2j}}$ The Jacobian of $\mathbf{r}_{l_{2j}}$ with respect to the state vector $\mathbf{x}$.

$\mathbf{H}_{d_H B}$ The Jacobian of the horizontal offset parameter with respect to the IMU body state $\mathbf{x}_B$.

$\mathbf{H}_{d_H L}$ The Jacobian of the horizontal offset parameter with respect to the laser calibration $\mathbf{x}_L$.

$\mathbf{I}$ Identity matrix.

$\mathbf{J_h}$ The Jacobian of the camera projection function.

$\mathbf{J}_{\mathbf{x},\mathbf{u}}(\theta_{V_i})$ The Jacobian of the vertical offset parameter with respect to $\mathbf{x}$ and $\mathbf{u}$.

$\mathbf{K}$ Kalman gain.

$\mathbf{k}$ The radial distortion parameters of the camera.

$\mathbf{n}$ Noise vector.

$\mathbf{N}_j$ The null space of matrix $\mathbf{H}_{f_j}$.

$\mathbf{n}_a$ Accelerometer white noise.

$\mathbf{n}_g$ Gyroscope white noise.

$\mathbf{n}_{wa}$ Accelerometer bias driving white noise.

$\mathbf{n}_{wg}$ Gyroscope bias driving white noise.

$\mathbf{o}$ The image center parameters of the camera.

$\mathbf{Q}_c$ Continuous time IMU noise matrix.

$\mathbf{Q}_n$ Discretized noise matrix.

$\mathbf{q}_{inc}$ An incremental quaternion.

$\mathbf{Q}_{l_j}$ The noise covariance matrix of j^{th} laser feature observation.

$\mathbf{r}_j$ Stacked residual vector for the camera observation of the j^{th} feature point.

$\mathbf{r}_j^{'}$ Marginalized residual vector for the camera observation of the j^{th} feature point.

$\mathbf{R}_{i,j}$ Noise matrix for the observation of the j^{th} point from the i^{th} camera.

$\mathbf{r}_{i,j}$ EKF residual from an observation of the j^{th} feature in the i^{th} camera image.

$\mathbf{r}_{l_{1j}}$ The residual for the angular constraint for the j^{th} line feature.

$\mathbf{r}_{l_{2j}}$ The residual for the distance constraint for the j^{th} line feature.

$\mathbf{t}$ The tangential distortion parameters of the camera.

$\mathbf{x}_B$ The IMU body states.

$\mathbf{x}_C$ The camera calibration states.

$\mathbf{x}_P$ The states for the plane map parameters.

$\mathbf{x}_{\pi_i}$ The i^{th} IMU body pose stored in the EKF's sliding window.

$\mathbf{x}_{aug}$ The augmentation state vector.

$\mathbf{x}_{lm}$ The laser calibration states for the mth laser scanner.

$\mathbf{y}$ The projection of a plane's normal vector into the global xy-plane.

$\mathbf{z}_{i,j}$ An observation of the j^{th} feature in the i^{th} camera image.

$\mathbf{z}_{l_{1j}}$ The angular constraint for the j^{th} line feature.

$\mathbf{z}_{l_{2j}}$ The distance constraint for the j^{th} line feature.

ϕ_j The angle of the j^{th} line observation.

ρ_j The orthogonal distance of the j^{th} line observation.

θ_{V_i} The normal vector angle parameter for the i^{th} vertical plane.

$\tilde{\mathbf{x}}$ The error in estimating quantity $\mathbf{x}$.

$\{B\}$ The IMU body coordinate frame.

$\{C\}$ The camera coordinate frame.

$\{G\}$ The global coordinate frame.

$\{L\}$ The laser scanner coordinate frame.

${}^B\mathbf{q}_G$ The rotation quaternion from global to IMU body coordinates.

${}^B\mathbf{q}_L$ The position of the laser in IMU body coordinates.

${}^B\mathbf{q}_L$ The rotation quaternion from laser to IMU body coordinates.

${}^C\mathbf{p}_{f_j}$ The j^{th} 3D feature point expressed in camera coordinates.

${}^C\mathbf{q}_B$ The position of the IMU body in camera coordinates.

${}^C\mathbf{q}_B$ The rotation quaternion from IMU body to camera coordinates.

${}^G\mathbf{g}$ The gravity vector expressed in global coordinates.

${}^G\mathbf{p}_B$ The position of the IMU in global coordinates.

Table 4.1: Summary of Coordinate Systems

Coordinate System	Symbol
Global Coordinate System	$\{G\}$
IMU Body Coordinate System	$\{B\}$
Camera Coordinate System	$\{C\}$
Laser Coordinate System	$\{L\}$

${}^{G}\mathbf{p}_{f_j}$ The j^{th} 3D feature point expressed in global coordinates.

${}^{G}\mathbf{v}_{B}$ The velocity of the IMU in global coordinates.

d_{H_i} The offset parameter for the i^{th} horizontal plane.

d_{V_i} The offset parameter for the i^{th} vertical plane.

n/N The delay ratio for the nth point.

r_i Radial coordinate of the i^{th} laser point.

t_d The timing delay parameter.

t_r The rolling shutter readout parameter.

W_i Weighting term for the i^{th} laser point.

4.3 State Space Representation

This section details the state space representation used by the Extended Kalman Filter estimator used in this Chapter. Central to IMU based estimators is the notion of coordinate systems. Table 4.1 shows a summary of all coordinate systems used throughout this Chapter. The global coordinate system $\{G\}$ is defined according to the East-North-Up convention so that height corresponds to elevation. The other sensor coordinate frames $\{B\}$, $\{C\}$, and $\{L\}$ are chosen to coincide with the IMU body, camera, and laser coordinate frames respectively.

4.3.1 Full State Definition

The EKF state vector can logically be divided into 5 groups of variables. The first is the time-varying state of the IMU body state $\mathbf{x}_B \in \mathbb{R}^{16\times 1}$ containing the position (3), orientation (4), velocity (3), and sensor biases of the IMU (6). The second group is the extrinsic (7), intrinsic (9), and temporal (2) calibration states of the camera $\mathbf{x}_C \in \mathbb{R}^{17\times 1}$. Similarly, the third group, $\mathbf{x}_{l_1} \cdots \mathbf{x}_{l_M}$, with $\mathbf{x}_{l_m} \in \mathbb{R}^{9\times 1}$, is the extrinsic (7) and temporal (2)calibration parameters for each of the M laser scanners in the system. Then, the plane parameters for both the H horizontal planes $\in \mathbb{R}^{1\times 1}$ and V vertical planes $\in \mathbb{R}^{2\times 1}$ are estimated in the fourth group $\mathbf{x}_P \in \mathbb{R}^{(2V+H)\times 1}$. Finally, a sliding window of the past N body poses are stored for usage by the Visual-Inertial Odometry (VIO) portion of the EKF denoted by $\mathbf{x}_{\pi_1} \cdots \mathbf{x}_{\pi_N}$ with each $\mathbf{x}_{\pi_n} \in \mathbb{R}^{10\times 1}$ containing an orientation (4), a position (3), and a velocity (3).

$$\mathbf{x} = \left[\mathbf{x}_B^T \mid \mathbf{x}_c^T \mid \mathbf{x}_{l_1}^T \cdots \mathbf{x}_{l_M}^T \mid \mathbf{x}_p^T \mid \mathbf{x}_{\pi_1}^T \cdots \mathbf{x}_{\pi_N}^T\right]^T \tag{4.1}$$

The time varying body states $\mathbf{x}_B$ is comprised of the orientation from the global coordinate to local coordinates ${}^B\mathbf{q}_G \in \mathbb{R}^{4\times1}$, the position in global coordinates ${}^G\mathbf{p}_B \in \mathbb{R}^{3\times1}$, the velocity in global coordinates ${}^G\mathbf{v}_B \in \mathbb{R}^{3\times1}$, and bias estimates of the gyroscope and accelerometer sensors, $\mathbf{b}_g \in \mathbb{R}^{3\times1}$ and $\mathbf{b}_a \in \mathbb{R}^{3\times1}$ respectively. The rotation is expressed in unit quaternion form. Thus:

$$\mathbf{x}_B = \begin{bmatrix} {}^B\mathbf{q}_G{}^T & {}^G\mathbf{p}_B{}^T & {}^G\mathbf{v}_B{}^T & \mathbf{b}_g^T & \mathbf{b}_a^T \end{bmatrix}^T \tag{4.2}$$

The camera calibration states $\mathbf{x}_C$ are comprised of the intrinsic, extrinsic, and temporal calibration parameters. The projective intrinsic parameters, as defined in Section 2.2, include the focal length $\mathbf{f} \in \mathbb{R}^{2\times1}$, the image center $\mathbf{o} \in \mathbb{R}^{2\times1}$, the radial distortion parameters $\mathbf{k} \in \mathbb{R}^{3\times1}$, and the tangential distortion parameters $\mathbf{t} \in \mathbb{R}^{2\times1}$. The extrinsic calibration consists of the rotation quaternion, ${}^C\mathbf{q}_B \in \mathbb{R}^{4\times1}$, and translation vector, ${}^C\mathbf{p}_B \in \mathbb{R}^{3\times1}$, from body to camera coordinates. The temporal calibration consists of the time stamping bias t_d between the timestamps reported by the camera and the timestamps reported by the IMU.

$$\mathbf{x}_c = \begin{bmatrix} {}^C\mathbf{q}_B{}^T & {}^C\mathbf{p}_B{}^T & \mathbf{f}^T & \mathbf{o}^T & \mathbf{k}^T & \mathbf{t}^T & t_d \end{bmatrix}^T \tag{4.3}$$

Similarly, each of the M laser state blocks $\mathbf{x}_{l_m}$ contains the rotation quaternion, ${}^B\mathbf{q}_L \in \mathbb{R}^{4\times1}$, and translation vector, ${}^B\mathbf{p}_L \in \mathbb{R}^{3\times1}$, from laser to body coordinates. In addition to the extrinsic calibration, each laser block contains estimates of the time stamping bias, t_d, and readout time t_r as defined in 2.3.2.

$$\mathbf{x}_l = \begin{bmatrix} {}^B\mathbf{q}_L{}^T & {}^B\mathbf{p}_L{}^T & t_d & t_r \end{bmatrix}^T \tag{4.4}$$

The planar map representation contains both horizontal and vertical planes. Horizontal planes are assumed to have their normal vector aligned with the global z-axis and are characterized by their scalar offset from the global origin d_{H_i} along their respective normal vector. Each vertical plane is assumed perfectly vertical and is characterized by its scalar offset from the global origin, d_{V_i}, and the angle, θ_{V_i}, its normal vector forms in the global xy plane. Thus for k horizontal planes and s vertical planes $\mathbf{x}_p$ is written as:

$$\mathbf{x}_p = \begin{bmatrix} d_{H_1} & \cdots & d_{H_k} & \mid & \{d_{V_1} & \theta_{V_1}\} & \cdots & \{d_{V_s} & \theta_{V_s}\} \end{bmatrix}^T \tag{4.5}$$

It is important to note that, although not explicitly represented in the state vector, the delimiting points are tracked for each plane to compute its support. The final block of states contains a sliding window of the N most recent IMU body states. Each pose in the sliding window contains a copy of the position ${}^G\mathbf{p}_B \in \mathbb{R}^{3\times1}$, velocity ${}^G\mathbf{v}_B \in \mathbb{R}^{3\times1}$, and orientation

${}^{B}\mathbf{q}_{G} \in \mathbb{R}^{4\times1}$ from the IMU body state $\mathbf{x}_B$ at the instant the pose was added to the sliding window.

$$\mathbf{x}_\pi = \begin{bmatrix} {}^{B}\mathbf{q}_{G}{}^{T} & {}^{G}\mathbf{p}_{B}{}^{T} & {}^{G}\mathbf{v}_{B}{}^{T} \end{bmatrix}^{T} \tag{4.6}$$

The total size of the state vector $\mathbf{x}$ depends on the number of laser scanners, the size of map, and the total length of the sliding window of poses. If there are M laser scanners, H horizontal planes, V vertical planes, and N poses in the sliding window, then the total state size is $16 + 17 + 9M + H + 2V + 10N$.

4.3.2 Error State Definition

While the quaternion representation of orientation requires 4 parameters, only 3 parameters are required for describing 3D rotations. To obtain a minimal representation of the rotation parameters in the EKF, an Indirect Kalman Filter [29] is used to track the error in the state estimate rather than the state itself. We utilize the standard additive error metric $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$ for all states except for those describing rotation. A tilde over a variable represents an estimation error and a hat over a variable represents a best estimate with the information up to the current time. For all quaternion parameters, the multiplicative error model is used. Specifically, the error between the quaternion estimate $\hat{\mathbf{q}}$ and the true value $\mathbf{q}$ is given by the 3 element error vector $\delta\boldsymbol{\theta}$. This formulation requires only 3 variables to represent rotation uncertainty and thus is minimal.

$$\begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta} \\ 1 \end{bmatrix} = \hat{\mathbf{q}}^{-1} \otimes \mathbf{q} \tag{4.7}$$

The symbol $\otimes$ in the previous equation represents quaternion multiplication.

$$\mathbf{q}_1 \otimes \mathbf{q}_2 = (q_{1w} + q_{1x}\mathbf{i} + q_{1y}\mathbf{j} + q_{1z}\mathbf{k})(q_{2w} + q_{2x}\mathbf{i} + q_{2y}\mathbf{j} + q_{2z}\mathbf{k}) \tag{4.8}$$

The full error state vector then takes the following partitioned form:

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{x}}_B^T \mid \tilde{\mathbf{x}}_c^T \mid \tilde{\mathbf{x}}_{l_1}^T \cdots \tilde{\mathbf{x}}_{l_M}^T \mid \tilde{\mathbf{x}}_p^T \mid \tilde{\mathbf{x}}_{\pi_1}^T \cdots \tilde{\mathbf{x}}_{\pi_N}^T \end{bmatrix}^T \tag{4.9}$$

and has total size $15 + 16 + 8M + H + 2V + 9N$. The key difference between the size of the error state vector and the size of the full state vector in Section 4.3.1 is that all quaternion errors are now parameterized with 3 dimensions instead of 4. This leads to the $2 + M + N$ reduction in state size. Although this reduce is not computationally significant, it allows errors in rotation to be expressed as linear instead of multiplicative constraints.

4.4 Filter Propagation

This section describes the process for propagating the EKF forward each time a new IMU data reading is available.

4.4.1 IMU State Dynamics

The state model chosen to describe the continuous time dynamics of the IMU follows the well developed models of [29] and [106]:

$$\begin{aligned} {}^{B}\dot{\mathbf{q}}_{G}(t) &= \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}(t)){}^{B}\mathbf{q}_{G}(t) \\ {}^{G}\dot{\mathbf{p}}_{B}(t) &= {}^{G}\mathbf{v}_{B}(t) \\ {}^{G}\dot{\mathbf{v}}_{B}(t) &= {}^{G}\mathbf{a}(t) \\ \dot{\mathbf{b}}_{g}(t) &= \mathbf{n}_{wg}(t) \\ \dot{\mathbf{b}}_{a}(t) &= \mathbf{n}_{wa}(t) \end{aligned} \tag{4.10}$$

The input rotational velocity, $\boldsymbol{\omega}(t) = \begin{bmatrix}\omega_x(t) & \omega_y(t) & \omega_z(t)\end{bmatrix}^T$, is expressed in the IMU frame of reference $\{B\}$, while the linear acceleration $\mathbf{a}(t) = \begin{bmatrix}a_x(t) & a_y(t) & a_z(t)\end{bmatrix}^T$ is expressed in the global coordinate frame $\{G\}$. The IMU gyroscope and accelerometer biases are driven by Gaussian white noise processes $\mathbf{n}_{wg}(t)$ and $\mathbf{n}_{wa}(t)$ respectively. Furthermore, the matrix $\boldsymbol{\Omega}(\boldsymbol{\omega})$ is defined as:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\times\rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} \tag{4.11}$$

and the skew-symmetric matrix is defined as:

$$\lfloor\boldsymbol{\omega}\times\rfloor = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{4.12}$$

Unfortunately, the IMU readings are corrupted by both zero mean Gaussian white noise $\mathbf{n}(t)$ and time-varying bias terms $\mathbf{b}(t)$. Therefore the measured readings reported by the IMU sensor are modeled as:

$$\begin{aligned} \boldsymbol{\omega}_m(t) &= \boldsymbol{\omega}(t) + \mathbf{b}_g(t) + \mathbf{n}_g(t) \\ \mathbf{a}_m(t) &= {}^{B}_{G}\mathbf{R}(t)(\mathbf{a}(t) - {}^{G}\mathbf{g}) + \mathbf{b}_a(t) + \mathbf{n}_a(t) \end{aligned} \tag{4.13}$$

where ${}^{G}\mathbf{g}$ is the gravity vector expressed in global coordinates, ${}^{B}_{G}\mathbf{R}(t)$ is the rotation matrix form of quaternion ${}^{B}\mathbf{q}_{G}(t)$, $\mathbf{n}_g(t)$ is the gyroscope sensor noise, and $\mathbf{n}_a(t)$ is the accelerometer white noise. The time varying biases are modeled as a Brownian motion process whose

derivatives are driven by zero mean Gaussian white noise. This implies that the combined noise disturbances are well modeled by a first order Markov process [107]. The input noise vector is then defined as follows:

$$\mathbf{n}(t) = \begin{bmatrix} \mathbf{n}_g(t) \\ \mathbf{n}_{wg}(t) \\ \mathbf{n}_a(t) \\ \mathbf{n}_{wa}(t) \end{bmatrix} \tag{4.14}$$

For simplicity, the covariance of the input noise is modeled as a diagonal matrix whose standard deviations, $\sigma_g, \sigma_{wg}, \sigma_a, \sigma_{wa}$, can be found either from manufacturer specifications or using Allan variance methods [107–110]. The combined IMU noise matrix is then defined as:

$$\mathbf{Q}_c = \mathrm{E}[\mathbf{n}\mathbf{n}^T] = \begin{bmatrix} \sigma_g^2\mathbf{I}_3 & 0 & 0 & 0 \\ 0 & \sigma_{wg}^2\mathbf{I}_3 & 0 & 0 \\ 0 & 0 & \sigma_a^2\mathbf{I}_3 & 0 \\ 0 & 0 & 0 & \sigma_{wa}^2\mathbf{I}_3 \end{bmatrix} \tag{4.15}$$

4.4.2 IMU State Propagation

In order to use the sampled readings from the IMU, the continuous time model of Eq. 4.10 must be converted into a discrete time representation suitable for numerical integration. First, the best estimate for the state propagation model is obtained by linearizing around the current state estimate and taking the expectation of Eq. 4.10.

$$\begin{aligned} {}^{B}\dot{\hat{\mathbf{q}}}_G(t) &= \frac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}(t)){}^{B}\hat{\mathbf{q}}_G(t) \\ {}^{G}\dot{\hat{\mathbf{p}}}_B(t) &= {}^{G}\hat{\mathbf{v}}_B(t) \\ {}^{G}\dot{\hat{\mathbf{v}}}_B(t) &= {}^{B_t}_{G}\hat{\mathbf{R}}^T\hat{\mathbf{a}}(t) + {}^{G}\mathbf{g} \\ \dot{\hat{\mathbf{b}}}_g(t) &= 0 \\ \dot{\hat{\mathbf{b}}}_a(t) &= 0 \end{aligned} \tag{4.16}$$

where the measured IMU values from Eq. 4.13 have been bias subtracted via $\hat{\mathbf{a}}(t) = \mathbf{a}_m(t) - \hat{\mathbf{b}}_a(t)$ and $\hat{\boldsymbol{\omega}}(t) = \boldsymbol{\omega}_m(t) - \hat{\mathbf{b}}_g(t)$ using the best estimate of the biases available. Following the derivation in [111] the discrete orientation quaternion update from time t_n to time t_{n+1} is defined as:

$${}^{B}\hat{\mathbf{q}}_G(t_{n+1}) = \mathbf{q}_{inc} \otimes {}^{B}\hat{\mathbf{q}}_G(t_n) \tag{4.17}$$

where the incremental quaternion $\mathbf{q}_{inc}$ is computed via 4th order Runge-Kutta numerical integration of the quaternion dynamics of Equation 4.16 using the bias subtracted rotational

velocities [112].

Furthermore, the velocity update is found by integrating the acceleration in the global frame.

$$\begin{aligned} {}^{G}\hat{\mathbf{v}}(t_{n+1}) &= {}^{G}\hat{\mathbf{v}}(t_n) + \int_{t_n}^{t_{n+1}} \hat{\mathbf{a}}(\tau)d\tau \\ &= {}^{G}\hat{\mathbf{v}}(t_n) + \int_{t_n}^{t_{n+1}} ({}^{G}_{B_n}\hat{\mathbf{R}}\,{}^{G}\hat{\mathbf{a}}(\tau) + {}^{G}\mathbf{g})d\tau \\ &= {}^{G}\hat{\mathbf{v}}(t_n) + {}^{G}_{B_n}\hat{\mathbf{R}}\hat{\mathbf{s}}_n + {}^{G}\mathbf{g}\Delta t \end{aligned} \tag{4.18}$$

where $\Delta t = t_{n+1} - t_n$ and the integral component is calculated as

$$\hat{\mathbf{s}}_n = \int_{t_n}^{t_{n+1}} {}^{B_{t_n}}_{B_\tau}\hat{\mathbf{R}}\,{}^{B}\hat{\mathbf{a}}(\tau)d\tau \tag{4.19}$$

Similarly, the position update is found by integrating the velocity update from Eq. 4.18.

$$\begin{aligned} {}^{G}\hat{\mathbf{p}}(t_{n+1}) &= {}^{G}\hat{\mathbf{p}}(t_n) + \int_{t_n}^{t_{n+1}} {}^{G}\hat{\mathbf{v}}(\tau)d\tau \\ &= {}^{G}\hat{\mathbf{p}}(t_n) + {}^{G}\hat{\mathbf{v}}(t_n)\Delta t + {}^{G}_{B_{t_n}}\hat{\mathbf{R}}\hat{\mathbf{y}}_n + \frac{1}{2}{}^{G}\mathbf{g}\Delta t^2 \end{aligned} \tag{4.20}$$

with $\hat{\mathbf{y}}_n$ computed as

$$\hat{\mathbf{y}}_n = \int_{t_n}^{t_{n+1}} \int_{t_n}^{t_{n+1}} {}^{B_{t_n}}_{B_\tau}\hat{\mathbf{R}}\,{}^{B}\hat{\mathbf{a}}(\tau)d\tau ds \tag{4.21}$$

In practice, the integrals from Eqs. 4.19 and 4.21 are numerically computed using trapezoidal integration.

4.4.3 IMU Error Propagation

During the prediction phase of the EKF, the covariance estimate of the error states must be propagated to accurately reflect the system dynamics. As noisy IMU readings are used to propagate the system's state forward in time, the covariance of the IMU related states should increase.

The error state covariance matrix is propagated by first linearizing the system around the current state estimates and then propagating it forward using the properties of linear Gaussian systems. Specifically the covariance matrix at time t_{n+1} is related to the covariance matrix at time t_n using

$$\boldsymbol{\Sigma}_{\boldsymbol{n+1|n}} = \begin{bmatrix} \boldsymbol{\Phi}_n & 0 \\ 0 & \mathbf{I} \end{bmatrix} \boldsymbol{\Sigma}_{n|n} \begin{bmatrix} \boldsymbol{\Phi}_n & 0 \\ 0 & \mathbf{I} \end{bmatrix}^T + \begin{bmatrix} \mathbf{Q}_n & 0 \\ 0 & 0 \end{bmatrix} \tag{4.22}$$

where $\mathbf{\Phi}_n$ is the 15×15 error state transition matrix for the IMU body states shown in Eq. 4.23, $\mathbf{I}$ is the identity matrix, and $\mathbf{Q}_n$ is the 15×15 contribution from the IMU noise vector shown in Eq 4.14. Following the derivation presented in [111], the IMU error state transition matrix takes the following block form:

$$\mathbf{\Phi}_n = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{\Phi}_{qb_g} & \mathbf{0}_3 \\ \mathbf{\Phi}_{pq} & \mathbf{I}_3 & \mathbf{I}_3\Delta t & \mathbf{\Phi}_{pb_g} & \mathbf{\Phi}_{pb_a} \\ \mathbf{\Phi}_{vq} & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{\Phi}_{vb_g} & \mathbf{\Phi}_{vb_a} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{4.23}$$

The individual blocks $\mathbf{\Phi}_{\boldsymbol{xy}} \in \mathbb{R}^{3\times3}$ represent the contributions to the state transition matrix from states x and y.

$$\mathbf{\Phi}_{qb_g} = -{}^{B_t}_{G}\hat{\mathbf{R}} \int_t^{t+1} {}^{B_t}_{B_\tau}\hat{\mathbf{R}} d\tau \tag{4.24}$$

$$\mathbf{\Phi}_{pq} = -\lfloor ({}^{G}\hat{\mathbf{p}}_B(t+1) - {}^{G}\hat{\mathbf{p}}_B(t) - {}^{G}\mathbf{v}\Delta t - \frac{1}{2}{}^{G}\mathbf{g}\Delta t^2) \times \rfloor \tag{4.25}$$

$$\mathbf{\Phi}_{pb_g} = \int_t^{t+1} \int_t^{w} \lfloor ({}^{G}\mathbf{a}(\tau) - {}^{G}\mathbf{g}) \times \rfloor {}^{B_t}_{G}\hat{\mathbf{R}} \int_t^{\tau} {}^{B_t}_{B_s}\hat{\mathbf{R}} ds d\tau dw \tag{4.26}$$

$$\mathbf{\Phi}_{pb_a} = -{}^{B_t}_{G}\hat{\mathbf{R}} \int_t^{t+1} \int_t^{\tau} {}^{B_t}_{B_s}\hat{\mathbf{R}} ds d\tau \tag{4.27}$$

$$\mathbf{\Phi}_{vq} = -\lfloor ({}^{G}\hat{\mathbf{v}}_B(t+1) - {}^{G}\hat{\mathbf{v}}_B(t) - {}^{G}\mathbf{g}\Delta t) \times \rfloor \tag{4.28}$$

$$\mathbf{\Phi}_{vb_g} = \int_t^{t+1} \lfloor ({}^{G}\mathbf{a}(\tau) - {}^{G}\mathbf{g}) \times \rfloor {}^{B_t}_{G}\hat{\mathbf{R}} \int_t^{\tau} {}^{B_t}_{B_s}\hat{\mathbf{R}} ds d\tau \tag{4.29}$$

$$\mathbf{\Phi}_{vb_a} = -{}^{B_t}_{G}\hat{\mathbf{R}} \int_t^{t+1} {}^{B_t}_{B_\tau}\hat{\mathbf{R}} d\tau \tag{4.30}$$

In practice, the integrals from the previous expressions are computed using trapezoidal numerical integration. For a full derivation of the individual blocks, see [111, 113, 114]. The IMU noise vector contribution $\mathbf{Q}_n$ in Eq. 4.22 is computed by utilizing the continuous time IMU noise matrix $\mathbf{Q}_c$ from Eq. 4.15 and integrating it through the continuous time dynamics. In particular we define the Jacobian of the error state $\tilde{\mathbf{x}}_B$ with respect to the IMU noise vector $\mathbf{n}$ as

$$\mathbf{G}_c = \begin{bmatrix} -\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & -{}^{B_{t_n}}_{G}\hat{\mathbf{R}}^T & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{4.31}$$

The discrete time noise matrix $\mathbf{Q}_n$ is then computed using trapezoidal integration on the following integral.

$$\mathbf{Q}_n = \int_{t_n}^{t_{n+1}} \mathbf{\Phi}_n \mathbf{G}_c \mathbf{Q}_c \mathbf{G}_c^T \mathbf{\Phi}_n^T d\tau \tag{4.32}$$

Rather than using Eq. 4.22 directly, the structure of the covariance matrix can be exploited to reduce computation. The full state covariance matrix $\mathbf{\Sigma}$ can be partitioned into the 15×15 block $\mathbf{\Sigma}_{BB}$ corresponding to the time-varying IMU states $\mathbf{x}_B$, and the blocks for the remainder of the states $\mathbf{\Sigma}_{RR}$

$$\mathbf{\Sigma}_{\boldsymbol{n+1|n}} = \begin{bmatrix} \mathbf{\Phi}_n & 0 \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_{BB} & \mathbf{\Sigma}_{BR} \\ \mathbf{\Sigma}_{RB} & \mathbf{\Sigma}_{RR} \end{bmatrix} \begin{bmatrix} \mathbf{\Phi}_n & 0 \\ 0 & \mathbf{I} \end{bmatrix}^T + \begin{bmatrix} \mathbf{Q}_n & 0 \\ 0 & 0 \end{bmatrix} \tag{4.33}$$

which reduces to

$$\mathbf{\Sigma}_{\boldsymbol{n+1|n}} = \begin{bmatrix} \mathbf{\Phi}_n \mathbf{\Sigma}_{\boldsymbol{BB}} \mathbf{\Phi}_n^T + \mathbf{Q}_n & \mathbf{\Phi}_n \mathbf{\Sigma}_{\boldsymbol{BR}} \\ (\mathbf{\Phi}_n \mathbf{\Sigma}_{\boldsymbol{BR}})^T & \mathbf{\Sigma}_{\boldsymbol{RR}} \end{bmatrix} \tag{4.34}$$

The above form is preferred because it eliminates unnecessary multiplications and helps ensure that the resulting covariance matrix $\mathbf{\Sigma}_{\boldsymbol{n+1|n}}$ stays symmetric.

4.5 Camera Data Fusion

Camera data fusion is carried out using the Multi-State Constraint Kalman Filtering (MSCKF) framework first presented by Mourikis and Roumeliotis [115]. The MSCKF algorithm is a EKF based Visual-Inertial Odometry estimator. This means that it fuses both IMU and camera data to obtain an open loop estimate of the camera trajectory. Since its initial publication, Li and Mourikis have presented many variants and improvements to the MSCKF algorithm. First, in [105], the filter was extended to incorporate the extrinsic calibration parameters. In [116], the authors proposed to keep track of the location of some landmarks in the state vector to reduce computation. Then, in [52], the MSCKF expanded to calibrate the camera sensor's readout and timestamp bias. Finally, in [40], the calibration model was expanded to recover the calibration parameters for both the IMU and the camera's intrinsic calibration parameters. The variant of the MSCKF used in this work only accounts for the

Figure 4.2: The flowchart for the camera data fusion algorithm.

camera intrinsic, extrinsic, and timestamp bias calibration parameters. In doing so, we only model the calibration parameters that are appropriate for our hardware setup.

Figure 4.2 shows a flowchart of the MSCKF algorithm used for camera data fusion. The key insight behind the MSCKF algorithm is to keep a sliding window of the past N poses where an image was recorded. The algorithm begins by tracking feature points sequentially though the image data. After a feature point is no longer tracked, its metrically accurate 3D position is triangulated using the poses from the sliding window. The tracked feature points and their metric 3D locations are then used to formulate an EKF update by first marginalizing out the component of the residual and Jacobians that is not tracked in the state vector before correcting the EKF's state estimate. In doing this, the filter is able to accurately estimate the velocity and gravity components of the IMU's attitude. However, it is worth noting that the MSCKF is only capable of providing accurate odometry estimates. The position and heading components of the state vector are unobservable and thus have unbounded error characteristics [105].

The details of the MSCKF algorithm are summarized in this section for completeness of presentation.

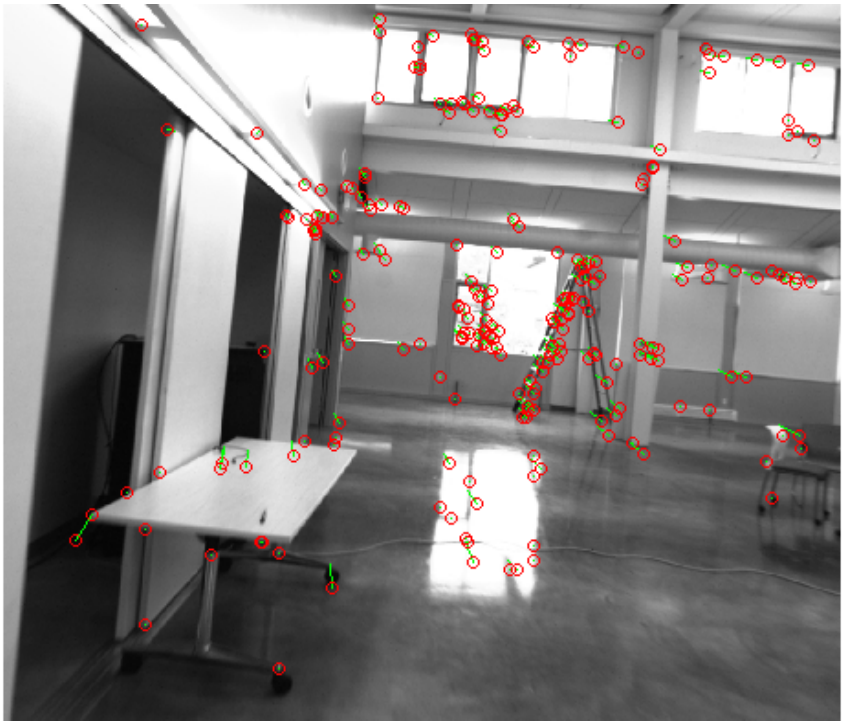
4.5.1 Image Feature Tracking

When a new image is available, the first step of the MSCKF algorithm is to find 2D feature points $\mathbf{z}$ that were present in the previous image. By tracking feature points sequentially through the images, we create feature tracks that contain the history of a where a point in 3D space appeared through a sequence of images. We perform tracking using a mixture of optical-flow and feature based tracking techniques as described below.

Optical-flow algorithms use image gradients from a set of images across multiple scales to generate motion estimates for points in an image [117]. Using the motion estimates, optical flow can predict the location of a point in the subsequent image. Iteratively applying the optical-flow algorithm tracks the feature until it is no longer visible. However, because the trajectory is built by summing incremental motion, accumulated errors can cause the trajectory to drift and become erroneous.

We address this issue using feature matching techniques. When a feature track is initialized, we cache a binary descriptor of the local image statistics. Binary descriptors are both computed quickly using thresholding and matched efficiently using Hamming distance [118]. Many binary descriptors, such as BRIEF [119] and ORB [120], have been proposed. We utilize the ORB feature descriptor in this work because it was designed to provide rotational invariance.

When a feature point is propagated from time t to $t+1$ using the optical-flow algorithm,



(a) Optical Flow Example

(b) Lost Feature Trajectories

Figure 4.3: An example of image feature tracking. (a): Optical-flow based motion vectors computed from sequential images. (b): The full trajectory of features that were unable to be located in the current image.

we extract a binary descriptor of its propagated location. We then compare the extracted descriptor against the database of cached descriptors from the image where the feature was first observed and find its nearest neighbor. By enforcing that the new descriptor be nearest neighbors with the cached descriptor from the start of the feature's trajectory, we eliminate both erroneous optical-flow results and feature points that are not sufficiently descriptive. If a feature is unable to be propagated via optical flow or fails descriptor matching, it is flagged as lost and used in a subsequent MSCKF update to be described in Section 4.5.5.

Figure 4.3 shows an example of feature tracking. Fig. 4.3(a) shows motion vectors computed for an example image. The green lines indicate the estimated motion between successive image frames and the red circles indicate the propagated location of the feature point in the current frame. Fig. 4.3(b) shows the full trajectory of features that were no longer tracked in the example image.

Once feature tracking is complete, new feature tracks must be created for previously untracked features and thus new feature points are detected using the FAST corner detector [121]. The FAST feature detector searches for corner points by comparing pixel values in a local neighborhood around a point. Since the FAST corner detector does not use multiple scales for detection, it is subject to poor performance under motion blur. Nevertheless, its speed and high average feature detection count make it an attractive option for odometry solutions.

4.5.2 Residual Definition

Common to all Kalman Filter formulations is the notion of a measurement process that allows the filter to estimate quantities observed via sensors and utilize the error, known as the residual, as feedback to improve its state vector. In this section, we describe the measurement model and associated Jacobians of the measurement function utilized by our EKF estimator during IMU-camera data fusion.

Each time a new image is available, the filter's sliding window of poses is augmented with the IMU's orientation, position, and velocity $\mathbf{x}_\pi$ at the reported time of the image t, as defined in 4.6. Since the images and IMU data may arrive asynchronously, the temporally closest IMU measurements are used to estimate $\mathbf{x}_\pi$ by integrating the nearest IMU body state $\mathbf{x}_B$ using the linearized filter dynamics of Eq. 4.16. Then, the MSCKF algorithm performs feature tracking to collect all feature trajectories which were no longer tracked in the current image as described in 4.5.1. Only feature tracks corresponding to features that are no longer tracked are used to avoid using information from the same measurement multiple times. The observation function that relates the j^{th} 3D feature point ${}^{G}\mathbf{p}_{f_j}$ and its observation $\mathbf{z}_{i,j}$ from the i^{th} pose in the sliding window $\mathbf{x}_{\pi_i}$ is found by transforming the feature point into world coordinates and then projecting it onto the image plane.

$$\mathbf{z}_{i,j} = \mathbf{h}({}^{C_i}\mathbf{p}_{f_j}) \tag{4.35}$$

The position of the 3D feature point ${}^{C_i}\mathbf{p}_{f_j}$ in the i^{th} camera's coordinates is computed by transforming position of the point in global coordinates ${}^{G}\mathbf{p}_{f_j}$ using the camera extrinsic calibration, $\{{}^{C}_{B}\mathbf{R}, {}^{C}\mathbf{p}_{B}\}$ temporal calibration t_d, and the stored position ${}^{G}\mathbf{p}_{B_i}$ and orientation

${}^{B_i}_{G}\mathbf{R}$ from the sliding window $\mathbf{x}_{\pi_i}$.

$$ {}^{C_i}\mathbf{p}_{f_j} = {}^{C}_{B}\mathbf{R}\,{}^{B_i}_{G}\mathbf{R}(t+t_d)\Big({}^{G}\mathbf{p}_{f_j} - {}^{G}\mathbf{p}_{B_i}(t+t_d)\Big) + {}^{C}\mathbf{p}_{B} \tag{4.36} $$

In the above equation, the true time the image was captured is computed as $t+t_d$ where t_d is the camera timestamp bias. The projection function $\mathbf{h}()$ that projects 3D points in camera coordinates is the same of Eq. 2.2. The estimate of the observation $\hat{\mathbf{z}}_{i,j}$ is then computed by combining Eq. 4.35 and 4.36 using the estimate of the i^{th} state from the sliding window $\hat{\mathbf{x}}_{\pi_i}$.

$$ \hat{\mathbf{z}}_{i,j} = \hat{\mathbf{h}}\left({}^{C}_{B}\hat{\mathbf{R}}\,{}^{B_i}_{G}\hat{\mathbf{R}}(t+\hat{t}_d)\Big({}^{G}\hat{\mathbf{p}}_{f_j} - {}^{G}\hat{\mathbf{p}}_{B_i}(t+\hat{t}_d)\Big) + {}^{C}\hat{\mathbf{p}}_{B}\right) \tag{4.37} $$

It is important to note that Eq. 4.37 requires an estimate of the 3D location of the feature ${}^{G}\mathbf{p}_{f_j}$ in world coordinates. A key difference between the MSCKF and other visual odometry methods is that the MSCKF does not track the positions of the features ${}^{G}\mathbf{p}_{f_j}$ in the state vector. In order to evaluate $\hat{\mathbf{z}}_{i,j}$ the poses and observations from the sliding window are used to recover an estimate of ${}^{G}\hat{\mathbf{p}}_{f_j}$ using inverse depth triangulation [122]. Inverse depth triangulation solves for the 3D position of a feature ${}^{C_i}\mathbf{p}_{f_j}$ by iteratively minimizing the reprojection error using a non-linear least squares optimization and the inverse depth parameterization. The residual is then formed by subtracting the observed location of the feature $\mathbf{z}_{i,j}$ and its estimate $\hat{\mathbf{z}}_{i,j}$.

$$ \begin{aligned} \mathbf{r}_{i,j} &= \mathbf{z}_{i,j} - \hat{\mathbf{z}}_{i,j} \\ &\approx \mathbf{H}_{\pi_i}\tilde{\mathbf{x}}_{\pi_i} + \mathbf{H}_{c}\tilde{\mathbf{x}}_{c} + \mathbf{H}_{f_{ij}}\,{}^{G}\tilde{\mathbf{p}}_{f_j} + \mathbf{n}_{i,j} \end{aligned} \tag{4.38} $$

The matrices $\mathbf{H}_{\pi_i}$, $\mathbf{H}_{c}$, and $\mathbf{H}_{f_ij}$ are the Jacobians of the linearized MSCKF residual from Eq. 4.35 with respect to the sliding window body pose $\mathbf{x}_{\pi_i}$, camera calibration states, and 3D feature point respectively. The noise vector $\mathbf{n}_{i,j}$ is assumed to be a zero mean Gaussian that is independent of the state vector and whose covariance is $\mathbf{R}_{i,j} = \sigma_f^2\mathbf{I}_{2\times 2}$. The image feature noise σ_f represents the noise from imperfect feature detection and tracking and is in practice assumed to be 1 pixel.

The Jacobians from Eq. 4.38 are required to apply the EKF update equations and are computed by applying first order Taylor approximation to Eq. 4.38 and ignoring higher order terms. For a full derivation of the Jacobians, see the work of [114]. The Jacobian with respect to the sliding window IMU body state $\mathbf{H}_{\pi_i}$ is given by [114].

$$ \mathbf{H}_{\pi_i} = \mathbf{J}_{\mathbf{h}}\,{}^{C}_{B}\hat{\mathbf{R}}\,{}^{B_i}_{G}\hat{\mathbf{R}}(t+\hat{t}_d)\begin{bmatrix}\lfloor {}^{G}\hat{\mathbf{p}}_{f_j} - {}^{G}\hat{\mathbf{p}}_{B_i}(t+\hat{t}_d)\times\rfloor & -\mathbf{I}_{3\times 3} & -\hat{t}_d\mathbf{I}_{3\times 3}\end{bmatrix} \tag{4.39} $$

where $\mathbf{J}_{\mathbf{h}}$ is the Jacobian of Eq. 2.2 with respect to the 3D position of the feature $\mathbf{p}_{f_j}$ expressed the camera coordinate frame. The Jacobian of the residual $\mathbf{H}_{c}$ with respect to the camera calibration states $\mathbf{x}_{c}$ is given as

$$\mathbf{H}_c = \begin{bmatrix} \frac{\delta \mathbf{r}}{\delta {}^C_B\tilde{\boldsymbol{\theta}}} & \frac{\delta \mathbf{r}}{\delta {}^C\tilde{\mathbf{p}}_B} & \frac{\delta \mathbf{r}}{\delta \tilde{t}_d} & \frac{\delta \mathbf{r}}{\delta \tilde{\mathbf{f}}} & \frac{\delta \mathbf{r}}{\delta \tilde{\mathbf{o}}} & \frac{\delta \mathbf{r}}{\delta \tilde{\mathbf{k}}} & \frac{\delta \mathbf{r}}{\delta \tilde{\mathbf{t}}} \end{bmatrix} \tag{4.40}$$

The extrinsic and temporal components of $\mathbf{H}_c$ are found using first order Taylor expansion of the residual and neglecting higher order terms.

$$\begin{aligned} \frac{\delta \mathbf{r}}{\delta \tilde{\boldsymbol{\theta}}^C} &\approx \mathbf{J}_\mathbf{h} {}^C_B\hat{\mathbf{R}} \lfloor {}^B_G\hat{\mathbf{R}}(t+\hat{t}_d)({}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_{B_i}(t+\hat{t}_d)) \times \rfloor \\ \frac{\delta \mathbf{r}}{\delta {}^C\tilde{\mathbf{p}}_B} &\approx \mathbf{J}_\mathbf{h} \\ \frac{\delta \mathbf{r}}{\delta \tilde{t}_d} &\approx -\mathbf{J}_\mathbf{h} {}^C_B\hat{\mathbf{R}} \Big({}^B_G\hat{\mathbf{R}}(t+\hat{t}_d) {}^G\hat{\mathbf{v}}(t+\hat{t}_d) \\ &\qquad + \lfloor \hat{\boldsymbol{\omega}}(t+\hat{t}_d) \times \rfloor {}^B_G\hat{\mathbf{R}}(t+\hat{t}_d)({}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_{B_i}(t+\hat{t}_d)) \Big) \end{aligned} \tag{4.41}$$

Here the best estimate of the rotational velocity $\hat{\boldsymbol{\omega}}(t+\hat{t}_d)$ is found by linearly interpolating the observed rotational velocity to time $t+\hat{t}_d$ using the temporally closest IMU readings and subtracting the best estimate of the gyroscope bias $\mathbf{b}_g$. The remaining elements of $\mathbf{H}_c$ correspond to the intrinsic calibration of the camera and can be found by differentiating Eq. 2.2 with respect to the appropriate calibration parameters.

The final Jacobian needed $\mathbf{H}_{f_{ij}}$ is the Jacobian of the residual with respect to the 3D feature position estimate ${}^G\mathbf{p}_{f_j}$.

$$\mathbf{H}_{f_{ij}} = \mathbf{J}_\mathbf{h} {}^C_B\hat{\mathbf{R}} {}^{B_i}_G\hat{\mathbf{R}} \hat{\mathbf{R}}(t+\hat{t}_n) \tag{4.42}$$

Once the residual vector $\mathbf{r}_{i,j}$ and Jacobian matrices $\mathbf{H}_{\pi_i}$, $\mathbf{H}_c$, and $\mathbf{H}_{f_{ij}}$ are computed for each of the n cameras that view feature ${}^G\mathbf{p}_{f_j}$, they are stacked to create a single residual vector.

$$\begin{aligned} \mathbf{r}_j &= \begin{bmatrix} \mathbf{r}_{1,j}^T & \dots & \mathbf{r}_{n,j}^T \end{bmatrix}^T \\ &\approx \mathbf{H}_\mathbf{x}\tilde{\mathbf{x}} + \begin{bmatrix} \mathbf{H}_{f_{1j}} \\ \vdots \\ \mathbf{H}_{f_{nj}} \end{bmatrix} {}^G\tilde{\mathbf{p}}_{f_j} + \mathbf{n}_i \\ &\approx \mathbf{H}_\mathbf{x}\tilde{\mathbf{x}} + \mathbf{H}_{f_j} {}^G\tilde{\mathbf{p}}_{f_j} + \mathbf{n}_i \end{aligned} \tag{4.43}$$

Here the aggregate Jacobian $\mathbf{H}_\mathbf{x}$ is the concatenation of $\mathbf{H}_{\pi_i}$ and $\mathbf{H}_c$. The stacked residual is used in a subsequent Kalman filter update.

4.5.3 Feature Error Marginalization

Unfortunately, the residual vector from Eq. 4.43 cannot be used directly by the filter. The residual vector contains a reliance on the feature error ${}^{G}\tilde{\mathbf{p}}_{f_j}$ which is not tracked by the filter. To marginalize out the feature error we project the residual vector $\mathbf{r}_j$ onto the left nullspace of $\mathbf{H}_{\mathbf{f}_j}$. The left nullspace $\mathbf{N}_j$ of $\mathbf{H}_{\mathbf{f}_j}$ is the matrix defined such that $\mathbf{N}_j^T\mathbf{H}_{\mathbf{f}_j} = \mathbf{0}$. By projecting Eq. 4.43 onto $\mathbf{N}_j$ the MSCKF marginalizes away the feature error.

$$\begin{aligned} \mathbf{N}_j^T\mathbf{r}_j &\approx \mathbf{N}_j^T(\mathbf{H}_{\mathbf{x}}\tilde{\mathbf{x}} + \mathbf{H}_{\mathbf{f}_j}{}^{G}\tilde{\mathbf{p}}_{f_j} + \mathbf{n}_i) \\ \mathbf{r}_j^{'} &\approx \mathbf{H}_{\mathbf{x}}^{'}\tilde{\mathbf{x}} + \mathbf{n}_i^{'} \end{aligned} \tag{4.44}$$

where $\mathbf{H}_{\mathbf{x}}^{'} \triangleq \mathbf{N}_j^T\mathbf{H}_{\mathbf{x}}$ and $\mathbf{n}_i^{'} \triangleq \mathbf{N}_j^T\mathbf{n}_i$. The new residual and Jacobians can now be used by the Kalman filter in a subsequent update. The new noise vector has the same noise characteristics as the original noise vector because $\mathbf{N}_j^T\mathbf{N}_j = \mathbf{I}$.

4.5.4 Outlier Rejection

Before incorporating the new residual and Jacobian into the state estimate using an EKF update, the MSCKF algorithm applies a statistical outlier rejection step. A Chi-squared test is performed in order to determine if the observed residual is statistically significant compared to the residual's estimate covariance [123, 124]. The Chi-squared test begins by computing the Mahalanobis distance [125] between the expected and observed measurement assuming that the measurements are normally distributed as $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$.

$$\gamma = \sum_{i=1}^{N} \left(\frac{X_i - \mu_i}{\sigma_i}\right)^2 \tag{4.45}$$

The Mahalanobis distance maps exactly to the norm of the residual weighted by the inverse of its covariance.

$$\gamma_j = \mathbf{r}_j^{'T}(\mathbf{H}_{\mathbf{x}}^{'}\mathbf{\Sigma}\mathbf{H}_{\mathbf{x}}^{'T} + \mathbf{R}_j)^{-1}\mathbf{r}_j^{'} \tag{4.46}$$

The distance γ_j, is then compared to the value X for which the Chi-squared cumulative distribution function (CDF) with $|\mathbf{r}^{'}|$ degrees of freedom takes on a 3σ value of 0.95. If γ_j is less than X then the Chi-squared test shows statistical significance and the feature is used in the next EKF update.

4.5.5 EKF Update

After discarding outliers the MSCKF algorithm stacks the residuals and Jacobians from all F inlier features into a single residual and Jacobian.

$$\begin{bmatrix} \mathbf{r}_1' \\ \vdots \\ \mathbf{r}_F' \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{\mathbf{x}1}' \\ \vdots \\ \mathbf{H}_{\mathbf{x}F}' \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} \mathbf{n}_1' \\ \vdots \\ \mathbf{n}_F' \end{bmatrix} \tag{4.47}$$
$$\mathbf{r} = \mathbf{H}\tilde{\mathbf{x}} + \mathbf{n}$$

An EKF update would normally involve computing the Kalman Gain and correction vectors, however, when the number of features becomes large, the complexity of computing the Kalman gain grows rapidly. To reduce the computational cost of an EKF update, the Jacobian $\mathbf{H}$ from Eq 4.47 is decomposed using QR factorization [126].

$$\mathbf{H} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{\Lambda} \\ \mathbf{0} \end{bmatrix} \tag{4.48}$$

The unitary matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ form the range and nullspace of $\mathbf{H}$ respectively. Premultiplying Eq. 4.47 by $\mathbf{Q}^T$ yields a lower dimensional system.

$$\begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix}^T \mathbf{r} = \begin{bmatrix} \mathbf{\Lambda} \\ \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix}^T \mathbf{n}$$
$$\begin{bmatrix} \mathbf{Q}_1^T\mathbf{r} \\ \mathbf{Q}_2^T\mathbf{r} \end{bmatrix} = \begin{bmatrix} \mathbf{\Lambda} \\ \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} \mathbf{Q}_1^T\mathbf{n} \\ \mathbf{Q}_2^T\mathbf{n} \end{bmatrix} \tag{4.49}$$
$$\mathbf{r}_Q \approx \mathbf{\Lambda}\tilde{\mathbf{x}} + \mathbf{n}_Q$$

In the above expression $\mathbf{r}_Q \triangleq \mathbf{Q}_1^T\mathbf{r}$ and $\mathbf{n}_Q \triangleq \mathbf{Q}_1^T\mathbf{n}$. The last approximation of Eq. 4.49 is justified because the lower portions of the new residual vector and Jacobian correspond to only input noise and thus can be discarded.

Finally, the EKF update is performed using the lower dimensional residual and Jacobian.

$$\begin{aligned} \mathbf{K} &= \mathbf{\Sigma}_{n+1|n}\mathbf{\Lambda}^T(\mathbf{\Lambda}\mathbf{\Sigma}_{n+1|n}\mathbf{\Lambda}^T + \mathbf{R}_Q)^{-1} \\ \Delta\mathbf{x} &= \mathbf{K}\mathbf{r}_Q \\ \mathbf{\Sigma}_{n+1|n+1} &= (\mathbf{I} - \mathbf{K}\mathbf{\Lambda})\mathbf{\Sigma}_{n+1|n}\mathbf{\Lambda})^T + \mathbf{K}\mathbf{R}_Q\mathbf{K}^T \end{aligned} \tag{4.50}$$

The matrix $\mathbf{R}_Q$ is the noise covariance matrix and is computed as:

$$\mathbf{R}_Q = \mathrm{E}[\mathbf{n}_Q\mathbf{n}_Q^T] \tag{4.51}$$

The state vector is then updated using the computed correction vector.

$$\hat{\mathbf{x}}_{n+1|n+1} = \hat{\mathbf{x}}_{n+1|n} + \Delta\mathbf{x} \tag{4.52}$$

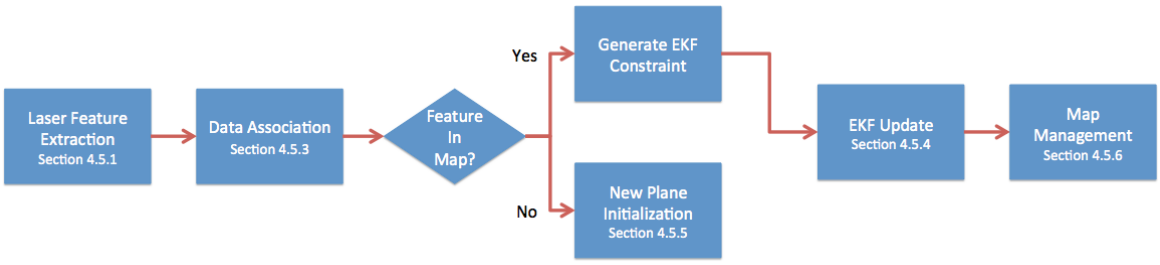


Figure 4.4: The flowchart for the laser data fusion algorithm.

Algorithm 4 Laser Data Fusion Algorithm

features = $generate\_line\_features$(data)
for all feature ∈ features **do**
  $perform\_data\_association$($\mathbf{x}_p$, feature)
  if feature ∈ $\mathbf{x}_p$ **then**
    $generate\_ekf\_constraint$(feature, $\mathbf{x}$)
  else
    $generate\_new\_plane$(feature, $\mathbf{x}$)
  end if
end for
$perform\_ekf\_update$($\mathbf{x}$)
$prune\_old\_planes$($\mathbf{x}_p$)

4.6 Laser Data Fusion

This section details the laser data fusion portion of the Extended Kalman Filter estimator. Each time a new laser data reading is available from one of the system's 2D laser scanners, an EKF measurement update is performed using the data from that laser scanner. In doing so, we modularly handle laser data from multiple laser scanners even if their data arrives into the system asynchronously.

Figure 4.4 and Algorithm 4 describe the steps required for using a laser scan reading during laser data fusion. First, the raw laser scan points must be converted into laser measurement features that can be used by the EKF. This is done by first grouping the laser scan points into continuous line segments and computing the line parameters using non-linear optimization. Then the laser measurement features must be associated with a plane stored in the EKF's plane map $\mathbf{x}_p$. If no plane is found, then the line measurement feature is used to initialize a new plane. Once data association is completed, all laser measurement features that were successfully associated with an existing plane are used in an EKF update to correct the state estimate. Lastly, the plane map is pruned of undesirable planes to limit the computational complexity of the filter.

Laser data fusion is carried out by applying an EKF update whenever a new laser reading is available. In general, in order for a sensor's data readings to be used in an EKF estimator, some measurement $\mathbf{z}_L$ is required that is either measured directly by sensor data or can be derived from the state vector, e.g. planes, and sensor data:

$$\begin{aligned}\mathbf{r} &\triangleq \mathbf{z}_L - \hat{\mathbf{z}}_L \\ &\approx \mathbf{H}_L\tilde{\mathbf{x}} + \mathbf{\Gamma}_L\tilde{\mathbf{u}}\end{aligned} \tag{4.53}$$

The matrix $\mathbf{H}_L$ is the Jacobian of the measurement $\mathbf{z}_L$ with respect to the error state vector, $\tilde{\mathbf{x}}$, and $\mathbf{\Gamma}_L$ is the Jacobian of the measurement $\mathbf{z}_L$ with respect to the measurement noise $\tilde{\mathbf{u}}$. Since 2D laser scanners only measure a set of discrete points sampled from a set of coplanar lines, they are incapable of making direct measurement of the planes from the compact map representation chosen in Eq. 4.5. In order for the raw laser points to be used as an EKF measurement, we must first extract meaningful features to be used as measurements $\mathbf{z}_L$ for the EKF.

The rest of this section is organized as follows. First, Section 4.6.1 describes how raw laser data is converted into line features that can be used by the EKF estimator. Section 4.6.2 then defines the two constraints provided to the EKF for each of the line features detected in the raw sensor data. The first constraint exploits the orthogonal relationship between any point in a line and its normal vector to constrain the orientation of the laser with respect to the plane. Similarly, the second constraint measures the orthogonal distance from the plane to the global origin in order to constrain the laser's position with respect to the plane. In order for the constraints from Section 4.6.2 to be evaluated, the laser line features must be associated with one of the planes stored in the filter's plane map $\mathbf{x}_p$ and this process is described in Section 4.6.3. Section 4.6.4 then describes how generated constraints are used by the EKF to update the filter's state estimate. For all line features that are not associated with an existing member of $\mathbf{x}_p$, Section 4.6.5 details how they are used to initialize new planes and expand the filter's map of the environment. Finally, Section 4.6.6 describes our map management strategy to maintain the filter's linear computational complexity with respect to acquisition time.

4.6.1 Laser Feature Extraction

Fundamentally, the laser scanners measure distances from the laser to various objects in the scene. As the laser traverses the environment, the scanning plane intersects the structural components of the building. The intersection of the laser scan plane with various horizontal or vertical planar building elements are considered the EKF measurements and are exploited as feedback for the EKF.

Figure 4.5 shows an illustration of a laser measurement. The laser scan plane intersects the i^{th} plane $\mathbf{\Pi}_i$ through the 3D j^{th} line segment $\boldsymbol{l}_{j\parallel}$ along viewing direction $\boldsymbol{l}_{j\perp}$. The lines $\boldsymbol{l}_{j\parallel}$ and $\boldsymbol{l}_{j\perp}$ are formed using the IMU body state, the transformation between laser and body coordinates, and a 2D line segment in the laser scanner's coordinates. The angle and orthogonal distance to the 2D line segment, as measured by the laser scanner, are denoted as ϕ_j and ρ_j respectively.

Unfortunately, 2D laser scanners do not measure line segments in the environment. A 2D laser scanner, such as the UTM-30LX, only measures distances from the laser scanner to objects in the environment. Using assumed bearings, the range measurements are converted into an ordered collection of discrete 2D points expressed in polar coordinates.

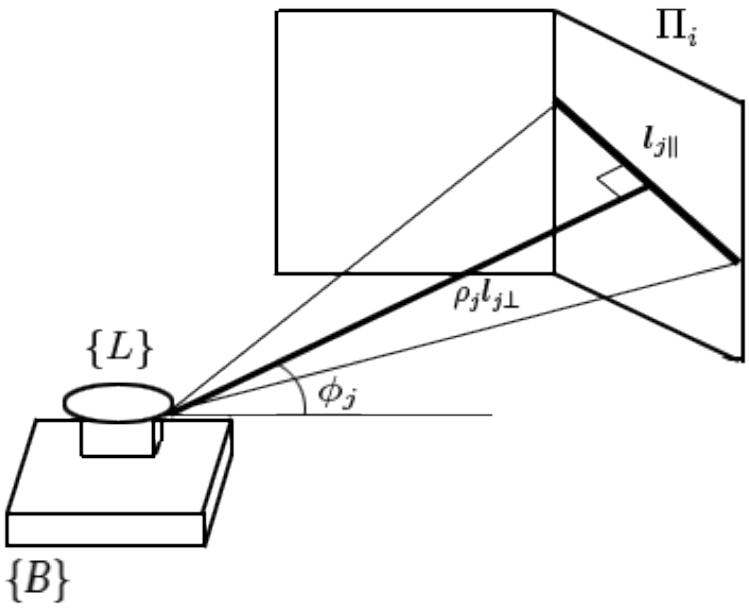


Figure 4.5: An illustration of the laser measurements used by the EKF. The laser scanner scan plane intersects vertical plane $\boldsymbol{\Pi}_i$ through line $\boldsymbol{l}_{j\|}$ along the measured direction $\boldsymbol{l}_{j\perp}$. The angle and orthogonal distance as measured by the laser scanner are denoted as ϕ_j and ρ_j respectively.

Before the EKF can use the laser scan in a measurement update, the points must be converted into line segments. First, the 2D points are segmented using the Split-and-Merge algorithm [127]. The Split-and-Merge algorithm recursively splits the points using a predefined error tolerance to group consecutive laser readings and then utilizes the ordered nature of the data to merge neighboring lines. Applying the Split-and-Merge algorithm groups the raw scan points into range readings originating from the same planar surface in the environment.

At this stage, filtering is done to remove point groupings that are either too sparse or represent a line feature that is below a predefined length. Removing undesirable point groupings serves two purposes. First, removing small line features keeps the EKF from modeling small, possibly non-planar, objects. Second, it prevents the state size from becoming intractably large. A minimal point grouping of 20 points and line length of 1 meter were used throughout this work.

It is important to note that although each point groupings represents a single planar object in the environment, the rolling-shutter nature of the laser scanner means that the entirety of a point groupings is not collected simultaneously. In order to model this aspect of the sensor, the point groupings are further subdivided into smaller groups. The size of the subdivided groups represents a trade-off between accurately modeling the rolling-shutter and having enough points for line fitting. A final group size was chosen empirically as 20

points, or 5° degrees of the Hokuyo UTM-30LX's field of view. This correlates to a maximal rolling-shutter time quantization of 0.15ms.

The line parameters (ϕ_j, ρ_j) and their covariances $\mathbf{Q}_{l_j}$ are estimated using Levenberg-Marquardt non-linear optimization [128, 129]. Specifically, given a set of N points in polar coordinates $\{r_i, \alpha_i\}_{i=1}^{N}$ parameterized by radius r_i and angle α_i, we assume that the points are corrupted with Gaussian noise $n_r \sim \mathcal{N}(0, \sigma_r^2)$ and $n_\alpha \sim \mathcal{N}(0, \sigma_\alpha^2)$. The orthogonal distance between the sampled point (r_i, α_i) line parameters and (ϕ_j, ρ_j) is given with the following expression f:

$$f(r_i, \alpha_i, \phi_j, \rho_j) = (r_i + n_r)\cos(\phi_j - (\alpha_i + n_\alpha)) - \rho_j \tag{4.54}$$

The non-linear line fitting procedure finds the line parameters (ϕ_j, ρ_j) that minimize the weighted, orthogonal distance to the points.

$$\sum_{i=1}^{N} \|r_i \cos(\phi_j - \alpha_i) - \rho_j\|_{W_{ij}}^2 \tag{4.55}$$

where the weighted norm is defined as

$$\|\mathbf{x}\|_{\mathbf{W}} = \mathbf{x}^T \mathbf{W}^{-1} \mathbf{x} \tag{4.56}$$

and for this particular problem the weighting factor is a single scalar W_ij calculated using the Jacobian of Eq. 4.54 with respect to the noise parameters n_r and n_α.

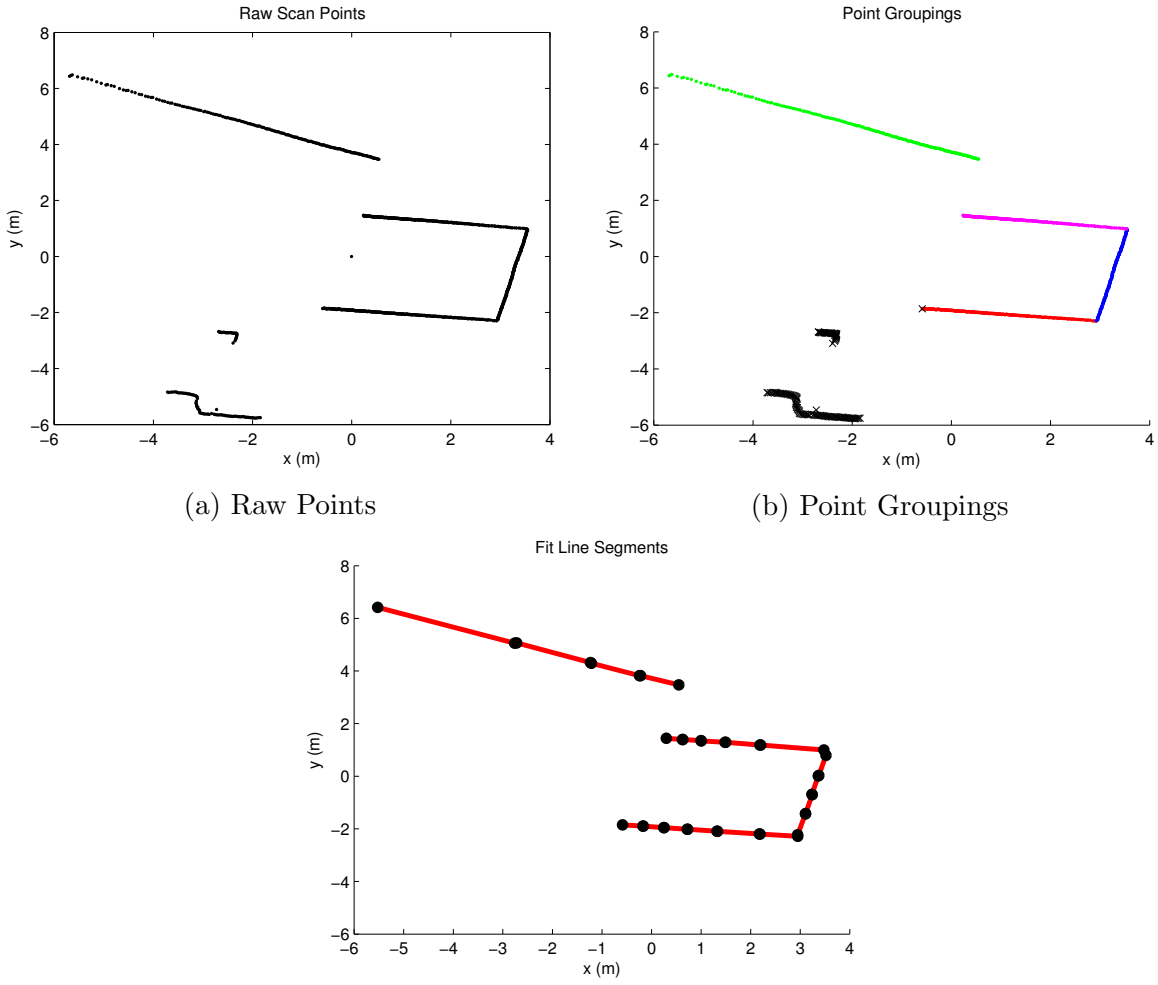
$$W_{ij} = \sigma_r^2 \cos^2(\phi_j - \alpha_i) + \sigma_\alpha^2 r_i^2 \sin^2(\phi_j - \alpha_i) \tag{4.57}$$

The advantage of this formulation is that it allows the EKF to account for both the noise in the range measurements as well as inaccuracies in the assumed bearings of the range measurements. In practice the variance of the input noise is chosen to match the characteristics of the UTM30-LX scanner used in our hardware systems $\sigma_r = 1$cm and $\sigma_\alpha = 0.25°$.

Figure 4.6 shows an example of the line feature extraction. Fig. 4.6(a) shows the raw scan points reported by the laser scanner. Fig. 4.6(b) shows the results of the Split-and-Merge algorithm. Note that different colors are used to denote the point groupings and black ×s are used to denote points that are not part of any line. Fig. 4.6(c) shows the final extracted line features. The black dots indicate the start and end points of each line measurement observation.

4.6.2 Laser Residual Definition

This section describes the constraints and Jacobians needed to use the extracted line features from Section 4.6.1 in the EKF estimator. After line features are extracted from the raw laser readings, they are used to define constraints in the EKF. A 2D line feature must originate



(a) Raw Points

(b) Point Groupings

(c) Extracted Line Features

Figure 4.6: An example of extracting line features from a 2D laser scan. (a): The raw points reported by the laser scanner. (b): The results of the Split-and-Merge algorithm. Different colored points denote the recovered point groupings and black ×s denote points not belonging to a line feature. (c): The lines resulting from the non-linear weighted line fitting algorithm. The black dots indicate the start and end points of each line measurement observation.

from some planar object in the environment and thus we treat it as a measurement of a line that lies on the plane of some existing element of the environment map $\mathbf{x}_p$. In this way, we derive both a constraint on the angle $\mathbf{z}_{l_{1_j}}$ and a constraint on the orthogonal distance $\mathbf{z}_{l_{2_j}}$ between the existing plane and the global origin. Intuitively, the angular constraint $\mathbf{z}_{l_{1_j}}$ exploits the fact that any line that originates from inside a plane must be orthogonal to its normal vector. Similarly, the second constraint, $\mathbf{z}_{l_{2_j}}$, takes advantage of the fact that the projection of the vector between all points on a plane and origin along the normal of the plane is the same for all points on the plane. These constraints allow for the EKF to update its estimate of the laser's orientation and position respectively. Since the system's orientation and position are correlated with these estimates through the laser calibration parameters, the EKF is able to additionally optimize the extrinsic and temporal calibration parameters for each of the system's laser scanners.

Once we have the individual constraints, we derive the required Jacobians and combine both the estimation residual and measurement Jacobians into a single EKF update to correct the filter's state estimate $\hat{\mathbf{x}}$. For a full derivation of the Jacobian matrices, see Appendix A.

The j^{th} laser line feature defines two constraints, $\mathbf{z}_{l_{1_j}}$ and $\mathbf{z}_{l_{2_j}}$, that are used as feedback for the EKF. First, the angle of the line measurement ϕ_j combined with the laser extrinsic calibration and current IMU body pose defines a 3D line, $\boldsymbol{l}_{j\|}$ that lies on the i^{th} scanned plane $\Pi_i = \{(d_i, \theta_i)\}$.

$$\boldsymbol{l}_{j\|} = \begin{bmatrix} \sin\phi_j \\ -\cos\phi_j \\ 0 \end{bmatrix} \tag{4.58}$$

The line $\boldsymbol{l}_{j\|}$ must trivially lie on plane Π_i and therefore must be perpendicular to its normal vector. If the normal vector is defined as $\boldsymbol{\pi}_i$, then the first constraint is obtained by rotating the line $\boldsymbol{l}_{j\|}$ into global coordinates

$${}^{G}\boldsymbol{l}_{j\|} = {}^{B}_{G}\mathbf{R}^{T}(t + t_d + \frac{n}{N}t_r){}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\|} \tag{4.59}$$

and then computing the dot product against the plane's normal vector $\boldsymbol{\pi}_i$.

$$\mathbf{z}_{l_{1_j}} = \boldsymbol{\pi}_i^{TG}\boldsymbol{l}_{j\|} = 0 \tag{4.60}$$

Although the right hand side of the above equation depends on both the ith plane and jth line feature, a line feature can only be associated with a single plane and thus by abuse of notation we drop i from the left hand side.

In order to compute the estimate $\hat{\mathbf{z}}_{l_1}$, the IMU body states must be interpolated to the actual time the line was observed $t_n = t_d + t_r n/N$. This actual time represents shifting the reported timestamp t using the offset bias t_d and the rolling-shutter readout delay $t_r n/N$ where N is the total number of points in a single laser scan and n is the index number of a particular point. The current IMU body state is integrated forward to time $t + t_d + t_r n/N$ using the system dynamics from Section 4.4.1. The total time delay caused by both the

timestamping bias t_d and rolling-shutter delay $t_r n/N$ is denoted as t_n. This results in the following expression:

$$\hat{\mathbf{z}}_{l_{1j}} = \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t + \hat{t_d} + \frac{n}{N}\hat{t_r}){}^B_L\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\|} \tag{4.61}$$

The residual is then defined as:

$$\mathbf{r}_{l_{1j}} \triangleq \mathbf{z}_{l_{1j}} - \hat{\mathbf{z}}_{l_{1j}} = -\hat{\mathbf{z}}_{l_{1j}} \tag{4.62a}$$
$$\approx \mathbf{H}_{l_{1j}}\tilde{\mathbf{x}} + \boldsymbol{\Gamma}_{l_{1j}}\tilde{\mathbf{u}}_j \tag{4.62b}$$

where $\mathbf{H}_{l_{1j}}$ and $\boldsymbol{\Gamma}_{l_{1j}}$ are the Jacobians of Eq. 4.60 with respect the error state $\tilde{\mathbf{x}}$ and the line measurement parameters $\tilde{\mathbf{u}}_j = (\rho_j, \phi_j)$ respectively. The second equality of Eq. 4.62(a) is a direct result of Eq. 4.60, i.e. $\mathbf{z}_{l_{1j}} = 0$. Although the residual also depends on which plane $\boldsymbol{\Pi}_i$ the line is associated with, subindex i is omitted from Eq. 4.62 for ease of notation.

The contribution to the residual from the error state $\mathbf{H}_{l_{1j}}\tilde{\mathbf{x}}$ can logically be split into contributions from the IMU body states $\mathbf{H}_{l_{1j}B}$, the laser calibration states as defined in Eq 4.4, $\mathbf{H}_{l_{1j}L}$, and the plane states $\mathbf{H}_{l_{1j}P_i}$.

$$\mathbf{H}_{l_{1j}}\tilde{\mathbf{x}} = \mathbf{H}_{l_{1j}B}\tilde{\mathbf{x}}_B + \mathbf{H}_{l_{1j}L}\tilde{\mathbf{x}}_l + \mathbf{H}_{l_{1j}P_i}\tilde{\mathbf{x}}_{p_i} \tag{4.63}$$

By applying first order Taylor expansion [130] and ignoring the higher order terms, the Jacobian of the residual with respect to the IMU body state is found to only be a function of the orientation error $\delta\hat{\boldsymbol{\theta}}^G(t)$ of the IMU body state pose.

$$\mathbf{H}_{l_{1j}B} = \begin{bmatrix} -\hat{\boldsymbol{\pi}}_i^T \lfloor {}^B_G\hat{\mathbf{R}}^T(t + \hat{t}_n){}^B_L\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\|} \times \rfloor & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} \end{bmatrix} \tag{4.64}$$

Recall that ${}^B_L\hat{\mathbf{R}}$ represents a rotation from coordinate frame $\{L\}$ to coordinate frame $\{B\}$ and $t_n = t_d + t_r n/N$. Similarly, by differentiating the residual with respect to the laser calibration parameters and ignoring the higher order terms, the Jacobian is found to only be a function of the orientation and temporal calibration parameters.

$$\mathbf{H}_{l_{1j}L} = \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t + \hat{t}_n) \begin{bmatrix} {}^B_L\hat{\mathbf{R}}\lfloor \hat{\boldsymbol{l}}_{j\|} \times \rfloor & \mathbf{0}_{3\times3} & \mathbf{M}_j & \frac{n}{N}\mathbf{M}_j \end{bmatrix} \tag{4.65}$$

The matrix $\mathbf{M}_j$ is defined as

$$\mathbf{M}_j = \lfloor \hat{\boldsymbol{\omega}}(t + \hat{t}_n) \times \rfloor {}^B_L\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\|} \tag{4.66}$$

and the best estimate of the rotational velocity $\hat{\boldsymbol{\omega}}(t + \hat{t}_n)$ is found by linearly interpolating the IMU rotational velocity to the best estimate of time $t + \hat{t}_n$ using the temporally closest IMU readings and removing the bias $\hat{\mathbf{b}}_g$.

$$\hat{\boldsymbol{\omega}}(t+\hat{t}_n) = \boldsymbol{\omega}_m(t+\hat{t}_n) - \hat{\mathbf{b}}_g \tag{4.67}$$

The Jacobian of the residual with respect to the plane parameters must be handled more carefully. A distinction must be made depending on whether the scanned plane is a vertical or horizontal plane. If the plane is horizontal, then there is no dependence on the plane parameters and thus the Jacobian is identically zero.

$$\mathbf{H}_{l_{1j}P_H} = \mathbf{0}_{1\times 1} \tag{4.68}$$

This can be intuitively understood because $\mathbf{z}_{l_{1j}}$ only relies on the plane's normal vector $\boldsymbol{\pi}_i$ and horizontal planes have a fixed constant normal vector. However, for vertical planes, the Jacobian contains a reliance on the error in the plane's normal angle θ_i.

$$\mathbf{H}_{l_{1j}P_V} = \begin{bmatrix}\mathbf{0}_{1\times 1} & \mathbf{e}_z^T \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor {}^B_G\hat{\mathbf{R}}^T (t+\hat{t}_n) {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\parallel}\end{bmatrix} \tag{4.69}$$

Here $\mathbf{e}_z$ is the elementary basis vector $\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$.

The Jacobian of the residual with respect to the line observation parameters $\boldsymbol{\Gamma}_{L_{1j}}$ is a matter of taking the derivative of Eq. 4.62 with respect to the parameters (ρ_j, ϕ_j). As expected, the Jacobian is only a function of the angle of the line.

$$\boldsymbol{\Gamma}_{l_{1j}} = \begin{bmatrix}\mathbf{0}_{1\times 1} & -\hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T (t+\hat{t}_n) {}^B_L\hat{\mathbf{R}} \lfloor\, \hat{\boldsymbol{l}}_{j\parallel} \times \rfloor \mathbf{e}_z\end{bmatrix} \tag{4.70}$$

The second constraint $\mathbf{z}_{l_2}$ is an estimate of the orthogonal distance from the origin to the observed plane. Using the line measurement parameters, we define the line $\boldsymbol{l}_{j\perp}$ such that $\boldsymbol{l}_{j\parallel} \perp \boldsymbol{l}_{j\perp}$.

$$\boldsymbol{l}_{j\perp} = \begin{bmatrix}\cos\phi_j \\ \sin\phi_j \\ 0\end{bmatrix} \tag{4.71}$$

By multiplying the line $\boldsymbol{l}_{j\perp}$ by the line parameter ρ_j, we generate a point on plane $\boldsymbol{\Pi}_i$ in the laser's coordinates. Once converted into global coordinates, the orthogonal distance from $\boldsymbol{\Pi}_i$ to the origin is computed by projecting it along the plane's normal vector $\boldsymbol{\pi}_i$.

$$d_i = \boldsymbol{\pi}_i^T \left({}^G\mathbf{p}_B + {}^B_G\mathbf{R}^T ({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\, \boldsymbol{l}_{j\perp}) \right) \tag{4.72}$$

The second constraint is formulated by moving all terms in Eq. 4.72 to the right hand side and setting the expression equal to 0.

$$\mathbf{z}_{l_2} = \boldsymbol{\pi}_i^T \left({}^G\mathbf{p}_B(t+t_n) + {}^B_G\mathbf{R}^T(t+t_n) ({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\, \boldsymbol{l}_{j\perp}) \right) - d_i = 0 \tag{4.73}$$

Similar to Eq. 4.61, the estimate $\hat{\mathbf{z}}_{l_2}$ is formed by interpolating the IMU body states to the best estimate of the observed feature's timestamp $\hat{t}_n$ and then evaluating Eq. 4.73.

$$\hat{\mathbf{z}}_{l_{2_j}} = \hat{\boldsymbol{\pi}}_i^T \big({}^G\hat{\mathbf{p}}_B(t+\hat{t}_n) + {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) \big) - \hat{d}_i \tag{4.74}$$

The second residual vector is defined as the error between the true $\mathbf{z}_{l_{2_j}} = 0$ and the observed measurement $\hat{\mathbf{z}}_{l_{2_j}}$.

$$\begin{aligned} \mathbf{r}_{l_{2_j}} &= \mathbf{z}_{l_{2_j}} - \hat{\mathbf{z}}_{l_{2_j}} = -\hat{\mathbf{z}}_{l_{2_j}} && \text{(4.75a)} \\ &\approx \mathbf{H}_{l_{2_j}} \tilde{\mathbf{x}} + \boldsymbol{\Gamma}_{l_{2_j}} \tilde{\mathbf{u}}_j && \text{(4.75b)} \\ &\approx \mathbf{H}_{l_{2_j}B} \tilde{\mathbf{x}}_B + \mathbf{H}_{l_{2_j}L} \tilde{\mathbf{x}}_l + \mathbf{H}_{l_{2_j}P_i} \tilde{\mathbf{x}}_{p_i} + \boldsymbol{\Gamma}_{l_{2_j}} \tilde{\mathbf{u}}_j && \text{(4.75c)} \end{aligned}$$

where $\mathbf{H}_{l_{2_j}B}$ is the Jacobian with respect to the IMU body states $\tilde{\mathbf{x}}_B$, $\mathbf{H}_{l_{2_j}L}$ is the Jacobian with respect to the laser calibration $\tilde{\mathbf{x}}_L$, $\mathbf{H}_{l_{2_j}P_i}$ is the Jacobian with respect to the plane parameters $\tilde{\mathbf{x}}_p$ and $\boldsymbol{\Gamma}_{l_{2_j}}$ is the Jacobian with respect to the input noise $\tilde{\mathbf{u}}_j$. The second equality of Eq. 4.75(a) is a direct consequence of Eq 4.73.

Similar to Eq. 4.63, the expression for the residual vector has been logically split based on its components. The Jacobian for the second residual vector with respect to the IMU body states is found by applying Taylor expansion and neglecting any higher order terms.

$$\mathbf{H}_{l_{2_j}} = \hat{\boldsymbol{\pi}}_i^T \begin{bmatrix} -\lfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) \times \rfloor & \mathbf{I}_{3\times3} & \hat{t}_n \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{4.76}$$

Notice that unlike in Eq. 4.64, the Jacobian for $\mathbf{r}_{l_{2_j}}$ contains both a dependence on the position and velocity errors. In a similar fashion, we derive the expression for $\mathbf{H}_{l_{2_j}L}$.

$$\mathbf{H}_{l_{2_j}L} = \hat{\boldsymbol{\pi}}_i^T \begin{bmatrix} {}^B_G\hat{\mathbf{R}}^T(\hat{t}_n)\hat{\rho}_j {}^B_L\hat{\mathbf{R}} \lfloor \hat{\boldsymbol{l}}_{j\perp} \times \rfloor & {}^B_G\hat{\mathbf{R}}^T(\hat{t}_n) & \mathbf{S}_j & \frac{n}{N}\mathbf{S}_j \end{bmatrix} \tag{4.77}$$

where the matrix $\mathbf{S}_j$ is defined as

$$\mathbf{S}_j = {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) \lfloor \hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor ({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) + {}^G\hat{\mathbf{v}}_B(t+\hat{t}_n) \tag{4.78}$$

The Jacobian $\mathbf{H}_{l_{2_j}P_i}$ again must be defined separately for horizontal and vertical planes. For horizontal planes P_H, $\mathbf{r}_{l_{2_j}}$ contains only the offset parameter d_i and thus the Jacobian is simply the negated 1×1 identity matrix.

$$\mathbf{H}_{l_{2_j}P_H} = -\mathbf{I}_{1\times1} \tag{4.79}$$

On the other hand, if the plane is vertical, the Jacobian also contains a dependence on the error in the plane's normal vector $\boldsymbol{\pi}_i$ and thus the plane's normal angle θ_i:

$$\mathbf{H}_{l_{2_j}P_V} = \begin{bmatrix} -\mathbf{I}_{1\times1} & \mathbf{e}_z^T \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor \big({}^G\hat{\mathbf{p}}_B(t+\hat{t}_n) + {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) \big) \end{bmatrix} \tag{4.80}$$

The final Jacobian needed, $\mathbf{\Gamma}_{l_{2j}}$, is found by differentiating the residual $\mathbf{r}_{l_{2j}}$ with respect to the laser measurement parameters (ρ_i, ϕ_i).

$$\mathbf{\Gamma}_{l_{2j}} = \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^B_L\hat{\mathbf{R}} \begin{bmatrix} \hat{\boldsymbol{l}}_{j\perp} & -\hat{\rho}_j \lfloor \hat{\boldsymbol{l}}_{j\perp} \times \rfloor \mathbf{e}_z \end{bmatrix} \tag{4.81}$$

After assembling the Jacobians and residuals for all m line feature measurements, they are concatenated into a large $2m$ residual vector and Jacobian matrix. The resulting system is then used by the EKF in the subsequent update step.

$$\begin{aligned} \mathbf{r}_l &= \begin{bmatrix} \mathbf{r}_{l_{11}} & \dots & \mathbf{r}_{l_{1m}} & \mathbf{r}_{l_{21}} & \dots & \mathbf{r}_{l_{2m}} \end{bmatrix}^T \\ &\approx \mathbf{H}_l \tilde{\mathbf{x}} + \mathbf{\Gamma}_l \tilde{\mathbf{u}} \end{aligned} \tag{4.82}$$

4.6.3 Laser Data Association

In order to evaluate the Jacobians $\mathbf{H}_l$ and $\mathbf{\Gamma}_l$ from Eq. 4.82, line measurement observations must be matched to a previously observed plane $\mathbf{\Pi}_i$. We perform data association using a combination of the projected distance of the laser line features to the candidate plane and probabilistic inlier thresholding. It is important to note that although in Sec. 4.6.1 we subdivided the point groupings into smaller line segments (ρ_j, ϕ_j), we desire to associate all segments originating from a single point grouping to the same previously observed plane. As such, we denote all line segments originating from a single point grouping as $\mathbf{u}^o = \{\rho_j, \phi_j\}_{j=1}^N$ and denote the stacked residual vectors, Jacobians, and noise matrices for $\mathbf{u}^o$ as $\mathbf{r}^o$, $\mathbf{H}_l^o$, and $\mathbf{Q}_l^o$ respectively.

The Chi-squared test is used to probabilistically test if a set of line segments should be associated with a previously observed plane $\mathbf{\Pi}_i$ [123, 124]. The Chi-squared test for associating a line feature measurement $\mathbf{u}^o$ with plane $\mathbf{\Pi}_i$ is identical to that of Section 4.5.4 with the exception that the Mahalanobis distance now takes the following form:

$$\gamma = \mathbf{r}^{oT}(\mathbf{H}_l^o \mathbf{\Sigma} \mathbf{H}_l^{oT} + \mathbf{\Gamma}_l^o \mathbf{Q}_l^o \mathbf{\Gamma}_l^{oT})^{-1} \mathbf{r}^o \tag{4.83}$$

Unfortunately, as the number of planes in the state vector increases, the computational cost of calculating probabilistic data association increases. To reduce computation, we only evaluate Eq. 4.83 for a small subset of the total planes in the map. The subset of candidate planes are chosen using fast geometric operations. First, we automatically eliminate all planes that have an average distance to $\mathbf{u}^o$ greater than 3 meters. The rationale for this is that we expect the filter's position estimate to be locally accurate to better than 3 meters. Then, we project the line segments $\mathbf{u}^o$ onto the remaining planes and find the minimal distance of the projection to the plane's bounding box. Figure 4.7 shows a diagram of this distance metric. By using the shortest distance between the projected line and the plane's bounding box, we prevent the filter from associating line segments with planes that are have similar parameterization but appear in separate parts of the environment. If the minimal distance between the projection and the plane's bounding box exceeds 0.5 meters, then the plane is no longer considered.

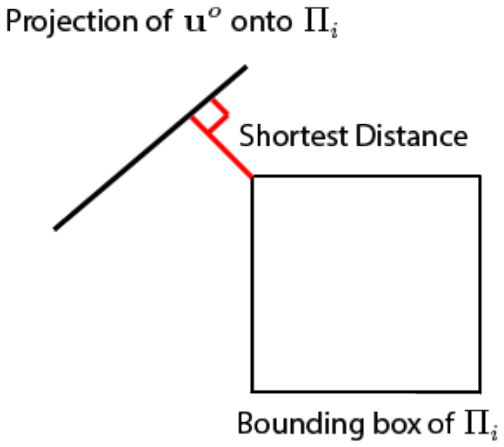


Figure 4.7: A diagram describing the distance metric used in the bounding box projection test during laser data association. The line grouping $\mathbf{u}^o$ is projection onto plane $\mathbf{\Pi}_i$. The shortest distance, shown in red, to the plane's bounding box is used as a rejection criteria for data association.

The remaining planes are then ordered temporally using the timestamp of their first observation and γ is computed until a plane association passes the Chi-squared test. By processing the remaining planes in this order, we introduce a slight bias towards older planes to encourage the filter to reuse and expand older planes.

4.6.4 Laser EKF Update

After forming the stacked Jacobians and residual vectors from Eq. 4.82, we apply an EKF update correct the state vector. The EKF update begins by first computing the Kalman Gain $\mathbf{K}$.

$$\begin{aligned}\mathbf{K} &= \mathbf{\Sigma}\mathbf{H}_l^T\mathbf{S}^{-1} \\ &= \mathbf{\Sigma}\mathbf{H}_l^T(\mathbf{H}_l\mathbf{\Sigma}\mathbf{H}_l^T + \mathbf{\Gamma}_l\mathbf{Q}_l\mathbf{\Gamma}_l)^{-1}\end{aligned} \tag{4.84}$$

Recall that the EKF's covariance matrix is denoted as $\mathbf{\Sigma}$ and the laser measurement noise matrix $\mathbf{Q}_l$ is computed from the weighted-line fitting procedure of Section 4.6.1 and $\mathbf{H}_l$ and $\mathbf{\Gamma}_l$ are from Eq. 4.82. The Kalman Gain can be efficiently computed without explicitly taking the inverse of the residual covariance $\mathbf{S}$ using QR factorization [126]. The correction vector $\mathbf{\Delta x}$ and updated covariance $\mathbf{\Sigma}_{n+1|n+1}$ are then computed using standard Kalman filter update equations.

$$\mathbf{\Sigma}_{n+1|n+1} = (\mathbf{I} - \mathbf{K}\mathbf{H}_l)\mathbf{\Sigma}_{n+1|n}\mathbf{H}_l)^T + \mathbf{K}\mathbf{\Gamma}_l\mathbf{Q}_l\mathbf{\Gamma}_l^T\mathbf{K}^T \tag{4.85}$$

$$\mathbf{\Delta x} = \mathbf{K}\mathbf{r}_l \tag{4.86}$$

4.6.5 Laser Based New Plane Initialization

This section defines the process for initializing new planes that will be tracked by the Extended Kalman Filter estimator. Since the environment map is not known a priori, observations corresponding to new planes must be used to expand the filter's environment map. In doing so, we augment the state vector $\mathbf{x}$ and error state covariance matrix $\mathbf{\Sigma}$ with estimates of the new plane's parameters and error covariance to incorporate them into the EKF's plane map $\mathbf{x}_p$. This allows the operator to explore and map new locations not already contained in the filter's environment map. The remainder of this section describes the process for computing the initial estimate of a new plane's parameters and Jacobians required for augmenting the state vector and error state covariance matrix.

If an observation does not get matched to any plane $\mathbf{\Pi}_i$ that already exists in the state vector, then we attempt use the observations $\mathbf{u}^o$ to initialize a new plane. Since the horizontal and vertical planes are modeled separately in the state vector, the line observations $\mathbf{u}^o$ must be classified as either a horizontal nor vertical line segment. Unfortunately, since a line contains only a single degree of freedom, classifying the line as horizontal or vertical is challenging even if we restrict ourselves to environments with only horizontal and vertical planes.

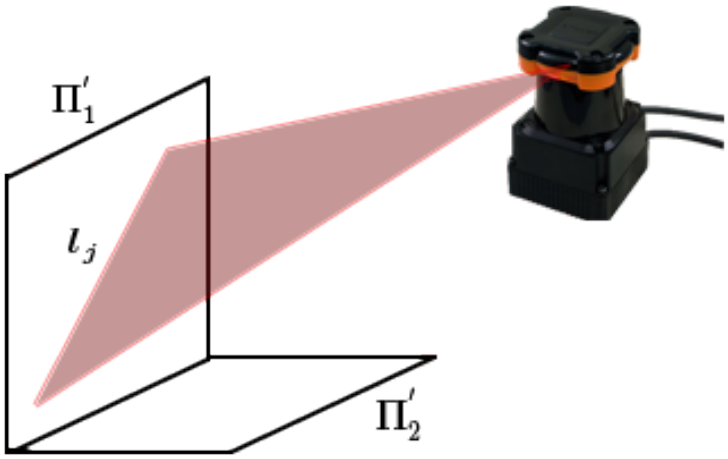
Figure 4.8 shows examples of non-degenerate and degenerate configurations for declaring the orientation of a line in an environment made from only horizontal and vertical planes. Figure 4.8(a) depicts a line feature that traverses a vertical plane through some angle. This can safely be classified as non-horizontal. However, if the line originated from a horizontal plane, then there arises a twofold ambiguity. The line can perfectly be fit to both a horizontal and vertical plane. In this scenario it becomes impossible to label the line as either originating from a horizontal or vertical plane.

The task of labeling lines becomes even more complicated when we consider real-world environments. Real buildings contain many planes, such as ramps, that are neither horizontal or vertical. For planes of arbitrary orientation, three non-collinear points are required to specify it uniquely. Using only a single group of observations, $\mathbf{u}^o$, it is not possible to classify the orientation of the line as horizontal, vertical, or neither. However, if we consider the line segments observed from another non-coplanar laser scanner, the intersection of two lines will uniquely define the orientation of the line segments $\mathbf{u}^o$.

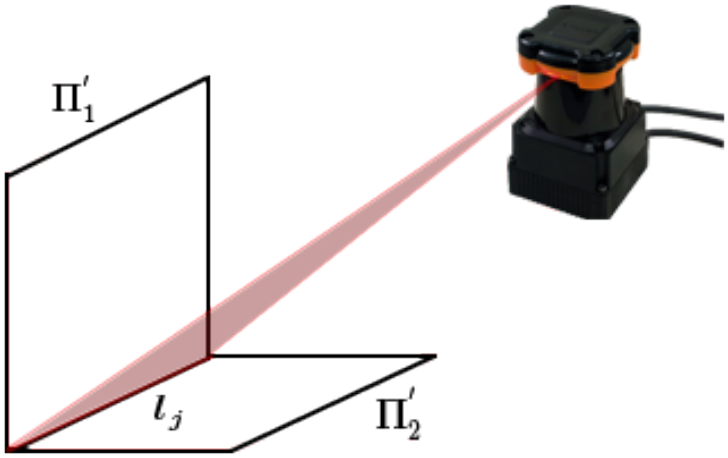
Motivated by this observation, we use the following strategy to classify the orientation of line segments. We first collect the temporally closest laser scans from all other non-coplanar scanners on the system and fit line segments using the weighted line-fitting algorithm described in Section 4.6.1. Then, we use the current estimate of the state vector and laser calibration to transform the line segments into the global coordinate frame $\{G\}$. Finally, we search for line segments that intersect $\mathbf{u}_j^o$ and define the surface normal as the cross product of the intersecting line segments. By comparing the surface normal to the global z vector, we declare the line as horizontal, vertical, or neither using simple thresholding.

Figure 4.9 shows an example of this process. The algorithm uses all non-coplanar black lines and attempts to intersect them with the colored query lines. If an intersection is possible then the lines are colored green for horizontal or blue for vertical. If no intersection is possible or the line is neither horizontal or vertical the line is colored red.

The example shown in Fig. 4.9 also highlights the conservativeness of our approach. Of



(a) Non-degenerate case.



(b) Degenerate case.

Figure 4.8: Non-degenerate and degenerate scenarios for declaring a line horizontal or vertical in a perfectly 2.5D world. (a): A line that scans a vertical plane at an angle can safely be declared non-horizontal. (b): Any line that lies on a horizontal plane can also perfectly fit a vertical plane.

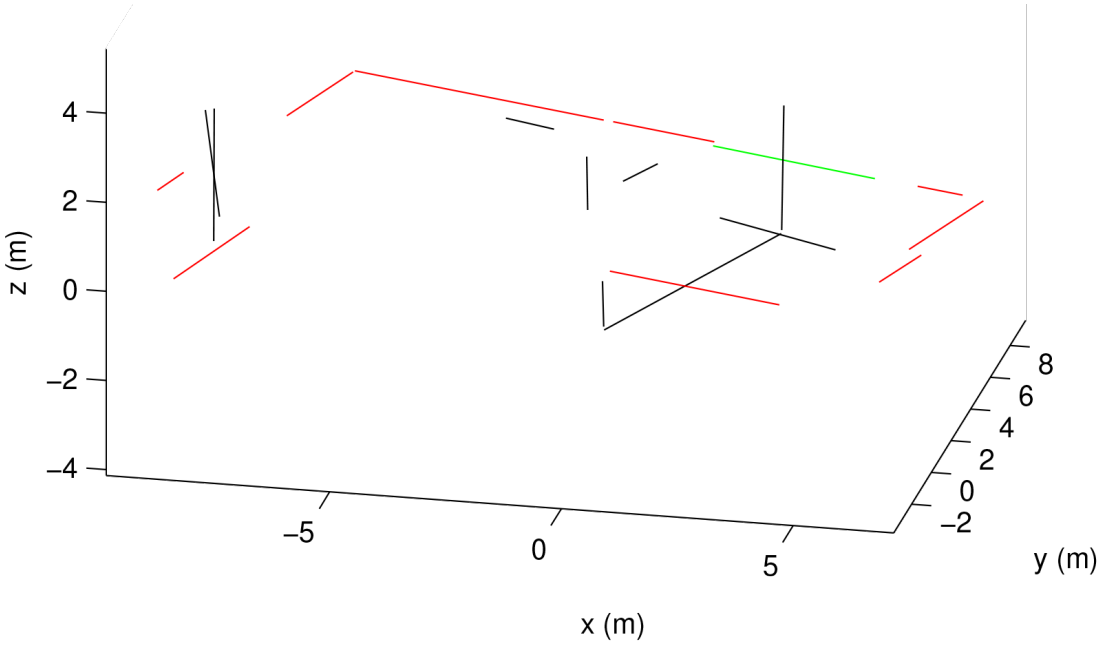


Figure 4.9: An example depicting how lines are declared either as horizontal, vertical, or neither. The colored query line segments are intersected with the line segments from the other scanners to determine the surface normal of the plane. In this example, green lines are declared horizontal, red lines are unknown, and black lines are the lines detected from the other laser scanners.

the 10 query lines, in only a single instance are we able to declare the orientation of the line. This however is generally not a major limitation. The high rate of the laser scanners allows numerous opportunities for the plane to be initialized using a later observation. Since the EKF is a least-squares approach, it is important to not pollute the filter's state estimate with outliers.

Once the orientation of the line has been determined, we must compute the new plane $\mathbf{\Pi}_i$ parameters $\mathbf{x}_{p_i}$ and an estimate of its error covariance $\mathbf{\Sigma}_{p_i}$. The error covariance is found by writing the expression for the plane parameters in error state notation.

$$\begin{aligned} \tilde{\mathbf{x}}_{p_i} &= \mathbf{x}_{p_i} - \hat{\mathbf{x}}_{p_i} \\ &\approx \mathbf{H}_{aug}\tilde{\mathbf{x}} + \mathbf{\Gamma}_{aug}\tilde{\mathbf{u}}_j \end{aligned} \tag{4.87}$$

where $\mathbf{H}_{aug}$ and $\mathbf{\Gamma}_{aug}$ are the Jacobians of $\tilde{\mathbf{x}}_{p_i}$ with respect to $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}_j$ respectively. The covariance of the new plane's P_i parameters are then computed using the properties of linear systems of Gaussian random variables.

$$\mathbf{\Sigma}_{p_i} = \mathbf{H}_{aug}\mathbf{\Sigma}\mathbf{H}_{aug}^T + \mathbf{\Gamma}_{aug}\mathbf{Q}_{l_j}\mathbf{\Gamma}_{aug}^T \tag{4.88}$$

Once the new plane state $\mathbf{x}_{p_i}$ and covariance matrix are known, they are augmented into the state vector $\mathbf{x}$ and error state covariance matrix $\mathbf{\Sigma}$ in order to track them as part of

the filter's plane map $\mathbf{x}_p$. In doing so, we expand the plane map and allow the operator to explore and map previously unknown regions of the environment.

$$\hat{\mathbf{x}}_{aug} = \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{x}}_{p_i} \end{bmatrix} \tag{4.89}$$

$$\boldsymbol{\Sigma}_{aug} = \begin{bmatrix} \mathbf{I} \\ \mathbf{H}_{aug} \end{bmatrix} \boldsymbol{\Sigma} \begin{bmatrix} \mathbf{I} \\ \mathbf{H}_{aug} \end{bmatrix}^T + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Gamma}_{aug} \mathbf{Q}_{l_j} \boldsymbol{\Gamma}_{aug}^T \end{bmatrix} \tag{4.90}$$

In the above expression, the augmented state vector $\hat{\mathbf{x}}_{aug}$ and covariance matrix $\boldsymbol{\Sigma}_{aug}$, are the state vector and error state covariance matrix after the new plane's parameters and error covariance have been added.

We now define the initial estimate $\hat{\mathbf{x}}_{aug}$ and corresponding Jacobians $\mathbf{H}_{aug}$ required for new plane initialization. Specifically, we require the Jacobians of horizontal plane parameter d_{H_i} and vertical plane parameters $\{d_{V_i}, \theta_{V_i}\}$ with respect to the state vector $\mathbf{x}$. Since the plane parameterization depends on the orientation of the plane, we must derive the plane parameters and Jacobians separately for the cases of horizontal and vertical planes.

For horizontal planes, the i^{th} plane is parametrized only by the offset from the origin d_{H_i}. An expression for computing the offset $\hat{d}_{H_i}$ is readily available by rearranging the laser observation from Eq. 4.74 to isolate the offset to the right hand side of the equation.

$$\hat{d}_{H_i} = \mathbf{e}_z^T \big({}^G\hat{\mathbf{p}}_B(t+\hat{t}_n) + {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}} \hat{\boldsymbol{l}}_{j\perp}) \big) \tag{4.91}$$

Equation 4.91 is in terms of only known or observed quantities and can readily be evaluated to obtain an estimate of the plane offset $\hat{d}_{H_i}$. Note that although the right hand side depends on both the i^{th} plane and j^{th} line, by abuse of notation we omit j from the right hand side because the plane is only initialized using a single line estimate. The covariance $\boldsymbol{\Sigma}_{p_i}$ for d_{H_i} is estimated by writing Eq. 4.91 in its error form and linearizing around the current state estimate. For a full derivation of the following Jacobians, see Appendix B.

$$\begin{aligned} \tilde{d}_{H_i} &= d_{H_i} - \hat{d}_{H_i} \\ &\approx \mathbf{H}_{d_H B} \tilde{\mathbf{x}}_B + \mathbf{H}_{d_H L} \tilde{\mathbf{x}}_L + \boldsymbol{\Gamma}_{d_H j} \tilde{\mathbf{u}}_j \end{aligned} \tag{4.92}$$

where $\mathbf{H}_{d_H B}$, $\mathbf{H}_{d_H L}$, $\boldsymbol{\Gamma}_{d_H j}$, are the Jacobians of Eq. 4.91 with respect to the IMU body state $\mathbf{x}_B$, the laser calibration $\mathbf{x}_L$ and the input noise $\mathbf{u}_j$ respectively. The Jacobian matrices are computed by applying first order Taylor approximation and neglecting higher-order terms. The resulting Jacobians $\mathbf{H}_{d_H B}$, $\mathbf{H}_{d_H L}$, and $\boldsymbol{\Gamma}_{d_H j}$,are identical to those from Eqs. 4.76, 4.77, and 4.81 respectively.

The vertical planes are parameterized using both the orthogonal distance to the origin d_{V_i} and the angle of the normal vector in the xy plane θ_{V_i}. Since the offset d_{V_i} depends on the normal vector $\boldsymbol{\pi}_i$, we must first solve for θ_{V_i}. The normal angle is defined as the angle the normal vector makes in the global xy plane.

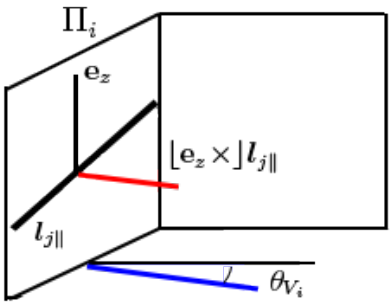


Figure 4.10: A diagram describing the geometry for vertical plane normal angle initialization. The normal angle θ_{V_i} is generated by projecting the cross product between observed line $\boldsymbol{l}_{j\|}$ and the global z vector $\mathbf{e}_z$.

$$\theta_{V_i} = \text{atan2}(v, u) \tag{4.93}$$

where $\text{atan2}(v, u)$ is the two argument arctangent function and $\begin{bmatrix} u & v \end{bmatrix}^T$ is the projection of the normal vector into the global xy plane. The projection of normal vector is formed by projecting the cross product of the global z vector and the line $\boldsymbol{l}_{j\|}$ expressed in world coordinates. Figure 4.10 shows a diagram of this process. The cross product between observed line $\boldsymbol{l}_{j\|}$ and global z vector $\mathbf{e}_z$, shown in red, is projected to a vector in the global xy plane, shown in blue, to form the normal angle θ_{V_i}.

$$\begin{aligned} \mathbf{y} = \begin{bmatrix} u \\ v \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \lfloor \mathbf{e}_z \times \rfloor {}^B_G\mathbf{R}^T (t + t_n) {}^B_L\mathbf{R}\, \boldsymbol{l}_{j\|} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \lfloor \mathbf{e}_z \times \rfloor \mathbf{f}(\mathbf{x}, \mathbf{u}_j) \end{aligned} \tag{4.94}$$

Evaluating Eq. 4.93 using the current state estimate $\hat{\mathbf{x}}$ yields an estimate of the normal angle $\hat{\theta}_{V_i}$. The offset $\hat{d}_{V_i}$ is found by evaluating the expected orthogonal distance from the plane to the origin.

$$\hat{d}_{V_i} = \hat{\boldsymbol{\pi}}_i^T \left({}^G\hat{\mathbf{p}}_B(t + \hat{t}_n) + {}^B_G\hat{\mathbf{R}}^T(t + \hat{t}_n)({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}} \hat{\boldsymbol{l}}_{j\perp}) \right) \tag{4.95}$$

It is important to note that Eq. 4.93 is undefined if both u and v are equal zero. This only occurs when the laser measurement observation is perfectly vertical. In this case, any vertical plane will fit the observation perfectly, and thus the situation is degenerate. We handle this situation by only allowing a line to initialize a new vertical plane when $u^2 + v^2 > 0$.

In order to estimate the covariance of the vertical plane we first write the plane's normal angle θ_{V_i} in error state form using first-order Taylor expansion.

$$\begin{aligned}\tilde{\theta}_{V_i} &= \theta_{V_i} - \hat{\theta}_{V_i} \\ &\approx \mathbf{J}_{\mathbf{x},\mathbf{u}}(\theta_{V_i})\big|_{\mathbf{x}=\hat{\mathbf{x}},\mathbf{u}=\hat{\mathbf{u}}} \begin{bmatrix}\tilde{\mathbf{x}} \\ \tilde{\mathbf{u}}_j\end{bmatrix}\end{aligned} \tag{4.96}$$

The Jacobian $\mathbf{J}_{\mathbf{x},\mathbf{u}}(\theta_{V_i})$ in the above expression is computed by applying the chain rule to Eqs. 4.93 and 4.94 and allows us to estimate covariance of $\tilde{\theta_{V_i}}$ as a function of the current error state $\tilde{\mathbf{x}}$.

$$\begin{aligned}\mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\theta_i) &= \mathbf{J}_{\mathbf{y}}(\theta_i)\mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\mathbf{y}) \\ &= \mathbf{J}_{\mathbf{y}}(\theta_i)\begin{bmatrix}1 & 0 & 0 \\ 0 & 1 & 0\end{bmatrix}\lfloor \mathbf{e}_z \times \rfloor \mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\mathbf{f}(\mathbf{x},\mathbf{u}_j))\end{aligned} \tag{4.97}$$

The Jacobian $\mathbf{J}_{\mathbf{y}}(\theta_i)$ of the four-quadrant arctangent function is found by taking the partials of the function $\text{atan2}(v,u)$ with respect to the inputs u and v.

$$\mathbf{J}_{\mathbf{y}}(\theta_i) = \begin{bmatrix}\frac{-v}{u^2+v^2} & \frac{u}{u^2+v^2}\end{bmatrix} \tag{4.98}$$

The Jacobian $\mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\mathbf{f}(\mathbf{x},\mathbf{u}_j))$ is found by rewriting the expression for $\mathbf{f}(\mathbf{x},\mathbf{u}_j)$ in error state form and applying first order Taylor expansion.

$$\begin{aligned}\tilde{\mathbf{f}}(\mathbf{x},\mathbf{u}_j) &= \mathbf{f}(\mathbf{x},\mathbf{u}_j) - \mathbf{f}(\hat{\mathbf{x}},\hat{\mathbf{u}}_j) \\ &\approx \mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\mathbf{f}(\mathbf{x},\mathbf{u}_j))\big|_{\mathbf{x}=\hat{\mathbf{x}},\mathbf{u}=\hat{\mathbf{u}}} \begin{bmatrix}\tilde{\mathbf{x}} \\ \tilde{\mathbf{u}}_j\end{bmatrix} \\ &\approx \mathbf{H}_{\theta B}\tilde{\mathbf{x}}_B + \mathbf{H}_{\theta L}\tilde{\mathbf{x}}_L + \mathbf{\Gamma}_{\theta j}\tilde{\mathbf{u}}_j\end{aligned} \tag{4.99}$$

The Jacobian $\mathbf{H}_{\theta B}$ is the derivative with respect to the IMU body states is given by:

$$\mathbf{H}_{\theta B} = -\begin{bmatrix}\lfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n){}^B_L\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\parallel}\times\rfloor & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\end{bmatrix} \tag{4.100}$$

and the Jacobian $\mathbf{H}_{\theta L}$ is the Jacobian with respect to the laser calibration states:

$$\mathbf{H}_{\theta L} = \begin{bmatrix}{}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n){}^B_L\hat{\mathbf{R}}\lfloor\,\hat{\boldsymbol{l}}_{j\parallel}\times\rfloor & \mathbf{0}_{3\times3} & {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)\mathbf{M}_j & \frac{n}{N}{}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)\mathbf{M}_j\end{bmatrix} \tag{4.101}$$

where $\mathbf{M}_j$ is the matrix expression from Eq. 4.66. The last Jacobian from Eq. 4.99, $\mathbf{\Gamma}_\theta$, needed to evaluate Eq. 4.96 is then simply the derivative with respect to the laser measurement parameters $\mathbf{u}_j$.

$$\mathbf{\Gamma}_{\theta j} = -\begin{bmatrix}\mathbf{0}_{3\times1} & {}^B_G\hat{\mathbf{R}}^T(\hat{t}_n){}^B_L\hat{\mathbf{R}}\lfloor\,\hat{\boldsymbol{l}}_{j\parallel}\times\rfloor\mathbf{e}_z\end{bmatrix} \tag{4.102}$$

The Jacobian expressions for $\tilde{d}_{V_i}$ are more complex. Using Eq. 4.95, we factor the expression for d_{V_i} into the product of a function relying only on the normal angle θ_{V_i} and one relying on the state vector $\mathbf{x}$ and line measurement observations $\mathbf{u}_j$.

$$\begin{aligned} d_{V_i} &= \boldsymbol{\pi}_i^T \big({}^G\mathbf{p}_B(t+t_n) + {}^B_G\mathbf{R}^T(t+t_n)({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\boldsymbol{l}_{j\perp}) \big) \\ &= \mathbf{f}(\theta_{V_i})^T \mathbf{g}(\mathbf{x}, \mathbf{u}_j) \end{aligned} \tag{4.103}$$

where $\mathbf{f}(\theta_{V_i}) \triangleq \boldsymbol{\pi}_i$ and $\mathbf{g}(\mathbf{x}, \mathbf{u}_j)$ is the remainder of the right hand side of Eq. 4.103. We then write the error form of Eq. 4.103 and recover the needed Jacobians by again using first order Taylor expansion.

$$\begin{aligned} \tilde{d}_i &= d_i - \hat{d}_i \\ &= \mathbf{f}(\theta_i)^T \mathbf{g}(\mathbf{x}, \mathbf{u}_j) - \mathbf{f}(\hat{\theta}_i)^T \mathbf{g}(\hat{\mathbf{x}}, \hat{\mathbf{u}}_j) \\ &\approx \mathbf{J}_{\mathbf{x},\mathbf{u}_j} \big(\mathbf{f}(\theta_i)^T \mathbf{g}(\mathbf{x}, \mathbf{u}_j) \big) \big|_{\mathbf{x}=\hat{\mathbf{x}}, \mathbf{u}_j = \hat{\mathbf{u}}_j} \begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{u}}_j \end{bmatrix} \end{aligned} \tag{4.104}$$

The total Jacobian $\mathbf{J}_{\mathbf{x},\mathbf{u}_j} \big(\mathbf{f}(\theta_i)^T \mathbf{g}(\mathbf{x}, \mathbf{u}_j) \big)$ is found by applying the multidimensional product rule for derivatives to the factored expression from Eq. 4.103.

$$\mathbf{J}_{\mathbf{x},\mathbf{u}_j} \big(\mathbf{f}(\theta_i)^T \mathbf{g}(\mathbf{x}, \mathbf{u}_j) \big) = \mathbf{g}^T(\mathbf{x}, \mathbf{u}_j) \mathbf{J}_{\theta_i} \big(\mathbf{f}(\theta_i) \big) \mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\theta_i) + \mathbf{f}^T(\theta_i) \mathbf{J}_{\mathbf{x},\mathbf{u}_j} \big(\mathbf{g}(\mathbf{x}, \mathbf{u}_j) \big) \tag{4.105}$$

Here the Jacobian $\mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\theta_i)$ from Eq. 4.105 can be obtained from Eq. 4.97 and the Jacobian of $\mathbf{f}(\theta_{V_i})$ with respect to θ_{V_i} is obtained through simple differentiation as follows:

$$\mathbf{J}_{\theta_i} \big(\mathbf{f}(\theta_i) \big) = \begin{bmatrix} -\sin(\theta_{V_i}) \\ \cos(\theta_{V_i}) \\ 0 \end{bmatrix} \tag{4.106}$$

The final Jacobian needed for Eq. 4.105, $\mathbf{J}_{\mathbf{x},\mathbf{u}_j} \big(\mathbf{g}(\mathbf{x}, \mathbf{u}_j) \big)$, is found by writing $\mathbf{g}(\mathbf{x}, \mathbf{u}_j)$ in error form and linearizing around the current state estimate.

$$\begin{aligned} \tilde{\mathbf{g}}(\mathbf{x}, \mathbf{u}_j) &= \mathbf{g}(\mathbf{x}, \mathbf{u}_j) - \mathbf{g}(\hat{\mathbf{x}}, \hat{\mathbf{u}}_j) \\ &\approx \mathbf{H}_{\mathbf{g}B} \tilde{\mathbf{x}}_B + \mathbf{H}_{\mathbf{g}L} \tilde{\mathbf{x}}_L + \boldsymbol{\Gamma}_{j\mathbf{g}} \tilde{\mathbf{u}}_j \end{aligned} \tag{4.107}$$

The Jacobians $\mathbf{H}_{\mathbf{g}B}$, $\mathbf{H}_{\mathbf{g}L}$, and $\boldsymbol{\Gamma}_{j\mathbf{g}}$ are almost identical to those of Eqs. 4.76, 4.77, and 4.81 respectively, with the exception that they are not pre-multiplied by $\hat{\boldsymbol{\pi}}_i^T$.

$$\mathbf{H}_{\mathbf{g}B} = \begin{bmatrix} -\lfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) \times \rfloor & \mathbf{I}_{3\times3} & \hat{t}_n \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{4.108}$$

$$\mathbf{H}_{\mathbf{g}L} = \begin{bmatrix} {}^B_G\hat{\mathbf{R}}^T(\hat{t}_n) \hat{\rho}_j {}^B_L\hat{\mathbf{R}} \lfloor \hat{\boldsymbol{l}}_{j\perp} \times \rfloor & {}^B_G\hat{\mathbf{R}}^T(\hat{t}_n) & \mathbf{S}_j & \frac{n}{N}\mathbf{S}_j \end{bmatrix} \tag{4.109}$$

$$\mathbf{\Gamma}_{j\mathbf{g}} = {}^{B}_{G}\hat{\mathbf{R}}^{T}(t+\hat{t}_n){}^{B}_{L}\hat{\mathbf{R}}\begin{bmatrix}\hat{\boldsymbol{l}}_{j\perp} & -\hat{\rho}_j\lfloor\hat{\boldsymbol{l}}_{j\perp}\times\rfloor\mathbf{e}_z\end{bmatrix} \tag{4.110}$$

The matrix $\mathbf{S}_j$ is in the previous equation is identical to the one defined in Eq. 4.78.

Once the Jacobians for $\tilde{d}_{V_i}$ and $\tilde{\theta}_{V_i}$ have been computed, we stack them to mimic the form found in Eq. 4.87. We then proceed by augmenting the state with the new plane parameters via Eq. 4.89 and covariance via Eq. 4.90.

4.6.6 Map Management

As the size of the map grows, the computational complexity to perform a Kalman filter update grows as $O(p^2)$ where p is the number of planes currently tracked in the map $\mathbf{x}_p$. In order to keep the computation bounded, we must limit the planes that are currently tracked by the EKF. We address the size of the map using three strategies: we enforce a minimum plane size, we forget planes that have not been observed recently, and we merge duplicate planes to be represented by a single set of states.

Real-world environments contain many small or insignificant planar surfaces. Since the EKF relies on viewing a plane more than once to update the state estimate, tracking small planes is typically not useful. Small surfaces, such as desks or pillars, are unlikely to be viewed multiple times and thus will not contribute to the overall state estimate. Although a naive approach that simply forbids using a line to initialize a small plane seems attractive, a slightly more sophisticated approach must be taken. Since planes are initialized from a single line measurement, any plane may be initialized with a small area regardless of the actual size of the plane. To avoid prematurely eliminating large planes, we prune planes that either have too small of a support area or were not viewed enough times only after they have existed for longer than a minimum time. For experiments presented, we found that a minimum time of 3 seconds, a minimum observation count of 15, and minimum area of 2 square meters works well in practice.

As the size of the scanned area increases, the number of large planes tracked by the EKF increases. To keep the computation time bounded, we must limit the size of the error covariance matrix. We achieve this by marginalizing out planes that the EKF has not been seen recently. We choose to forget any planes that have not been seen for over 120 seconds. The EKF models the error's joint probability distribution as a set of Gaussian random variables, and thus marginalization of a plane requires only dropping the associated rows and columns from the covariance matrix [131].

Our final plane management strategy is to eliminate duplicate planes by merging them together. Duplicate planes are detected by checking if planes $\mathbf{\Pi}_i$ and $\mathbf{\Pi}_j$ have parameters that have converged on the same estimate. In practice, this means that for horizontal planes we check that $\|d_{H_i} - d_{H_j}\| < \beta_d$ and for vertical planes we check that $\|d_{V_i} - d_{V_j}\| < \beta_d$ and $\|\theta_{V_i} - \theta_{V_j}\| < \beta_\theta$ for some thresholds β_d and β_θ. Furthermore, because the planes are mathematically represented as infinite in extent, we ensure that the observed support of plane $\mathbf{\Pi}_i$ and $\mathbf{\Pi}_j$ overlaps before merging. For all experiments presented, we used the thresholds of $\beta_d = 5$cm and $\beta_\theta = 5°$.

Although it is possible to simply marginalize away the duplicate plane, that would lead to sub-optimal filtering. If the duplicate plane $\mathbf{\Pi}_j$ was used for an EKF update, then some information about its duplicate $\mathbf{\Pi}_i$ is contained in its estimate. Marginalizing $\mathbf{\Pi}_j$ away

would neglect that information and lead to sub-optimal filtering. We merge the planes in an optimal fashion by first marginalizing away $\mathbf{\Pi}_j$ and then using it as a direct observation of plane $\mathbf{\Pi}_i$. Expressed in error state form, the residual expression is expressed trivially.

$$\begin{aligned} \tilde{\mathbf{x}}_{p_i} &= \mathbf{H}\tilde{\mathbf{x}} + \mathbf{\Gamma}\tilde{\mathbf{x}}_{p_j} \\ \tilde{\mathbf{x}}_{p_i} &= \mathbf{0}\tilde{\mathbf{x}} + \mathbf{I}\tilde{\mathbf{x}}_{p_j} \\ &= \tilde{\mathbf{x}}_{p_j} \end{aligned} \tag{4.111}$$

This form allows the EKF to incorporate the information from $\mathbf{\Pi}_j$ into its estimate of $\mathbf{\Pi}_i$ using the standard EKF update machinery. Although it appears that Eq. 4.111 is a contradictory statement, collapsing estimates of $\tilde{\mathbf{x}}_{p_i}$ and $\tilde{\mathbf{x}}_{p_j}$ directly via an EKF update averages their two estimates together into the optimal linear estimate.

4.7 Filter Initialization

Non-linear estimators, such as the EKF, require a reasonable initial condition of the state vector to ensure proper filter operation. This is in general a difficult problem for IMU based fusion techniques because the accelerometer is corrupted with both gravity and bias terms. Furthermore, in order for the IMU to correctly fuse data from other sensors, a reasonable initial condition of the extrinsic rotation between the IMU and external sensors is required. To that end, we initialize the filter using a two-stage process. First, we perform a zero velocity update (ZUPT) [132] on the system to estimate the gyroscope bias and gravity vector by averaging the data from the IMU. A ZUPT is simply the process of forcing the IMU to be still in order to constrain the velocity be a constant $\mathbf{0}$. Then, we run the fusion algorithms while exciting all rotational and translation axis between the sensors and the IMU. Specifically, we first translate along all three major axes to excite the translational degrees of freedom followed by independently rotating around all three major axes to excite the rotational degrees of freedom. In doing so, we optimize away any error in the bias and calibration parameters before exploring the environment.

4.8 Path Smoothing

A direct consequence of our plane management strategies is that it does not allow the EKF to perform arbitrary loop closure. If the EKF estimator was allowed to maintain an infinitely large map, the system would be able to perform loop closure simply be associating new measurements with planes seen arbitrarily in the past. By marginalizing out planes that have not been viewed recently, we lose the ability to loop closure if we return and view that plane again at a later time. As a result, we are not guaranteed to have bounded error in the position and heading components of our output trajectory. This will lead to inconsistencies in the reconstructed model and must be addressed.

We handle the long term loop closure problem using the following approach. First, we subdivide the trajectory into small sections and create local submaps. We then recursively create a pose graph using the EKF's recovered trajectory as odometry estimates. Specifically,

since the EKF generates an estimate of the systems pose after each IMU reading, camera reading, and laser reading, we first subsample the trajectory to 20Hz to limit computation time. Although pose estimates corresponding to IMU reading are less accurate than those from either the laser or camera readings, in practice a naive subsampling algorithm was determined to be sufficient. We then sequentially add nodes to the graph corresponding to the subsampled poses and edges generated from the corresponding odometry estimates. Each time a new pose corresponding to a submap is added to the pose graph, we search in a small neighborhood of the current pose estimate for other submap locations. If we locate another submap, a loop closure constraint is added by matching the submaps using the iterative closest point (ICP) algorithm [82]. The pose graph is then input to an incremental graph optimizer to compute an optimized estimate of the trajectory using the iSAM2 algorithm [33]. This process is iterated until all odometry measurements have been used. The final, optimized 20Hz trajectory is used for point cloud generation.

4.8.1 Submap Generation

We generate submaps by subdividing the trajectory into small, overlapping sections. Each time the odometry estimates the system has traversed N meters, a new submap is generated by combining the previous $2N$ meters and current N meters sections of the path. In doing this, we generate submaps that contain $3N$ meters of path trajectory and overlap with neighboring submaps by 2/3. We empirically found a value of $N = 2$ meters provided a good balance between submap size and locality.

The submaps are built from the range readings from the laser scanners. The laser points are converted into world coordinates using both the best estimate of the laser calibration $\hat{\mathbf{x}}_L$ and body state estimate $\hat{\mathbf{x}}_B$ at the time when the laser scan point was captured. An example local submap is displayed in Fig. 4.11. Note that the roof has been manually removed and the submap has been colored according to height for visualization purposes.

4.8.2 Loop Closure Generation

Loop closure constraints are generated using a greedy, distance-based heuristic. Odometry measurements from the EKF generated path are sequentially added to the trajectory until we have traversed $3N$ meters. We then search in a ball around the current pose estimate for other submaps. If any submap poses are found within the search radius, we greedily select the closest submap and attempt to create a loop closure constraint. ICP is then run between the local submaps to generate an estimate of the transformation $\hat{\mathbf{T}}$ between the two submap poses. We utilize the point-to-plane distance metric and reciprocal nearest-neighbor rejection criterion to accurately match local submaps even in the presence of a large number of outliers [133, 134].

Although our loop closure selection method is highly dependent on accurate odometry measurements, this is not a major limitation because iSAM2 incrementally re-estimates the trajectory each time a new loop closure is added. In doing so, we only have unbounded error between loop closure constraints instead of throughout the entire dataset. Coupled with the accuracy of the EKF estimator, a simple neighborhood search was deemed sufficient for the majority of real-world datasets.

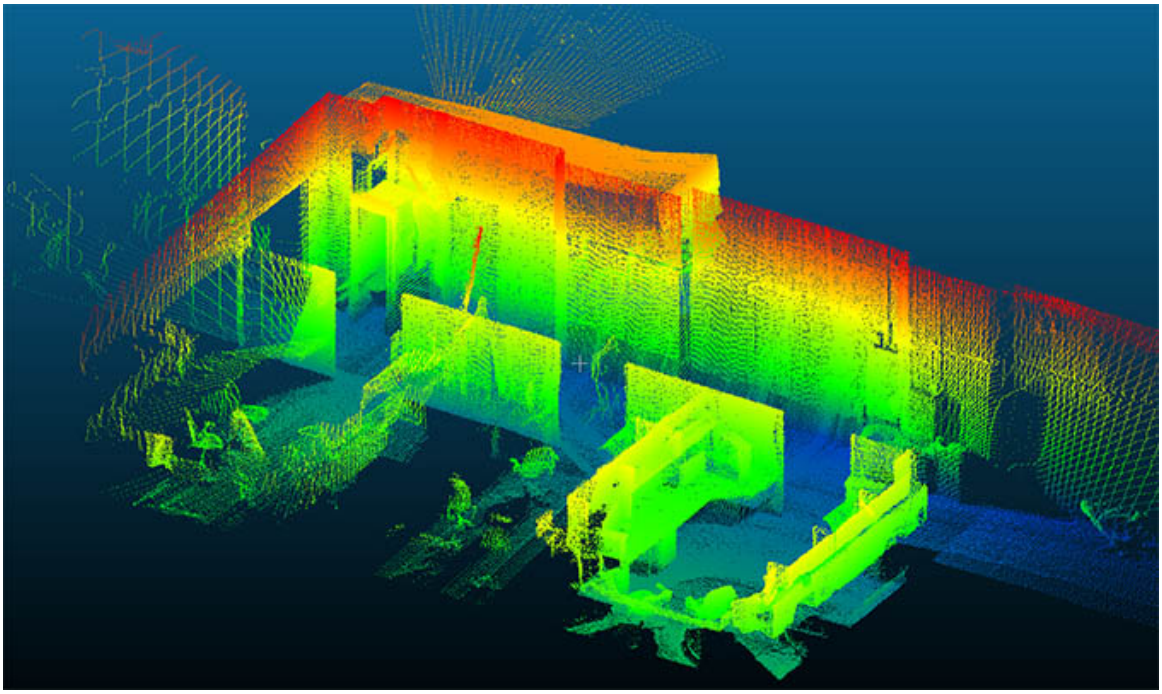
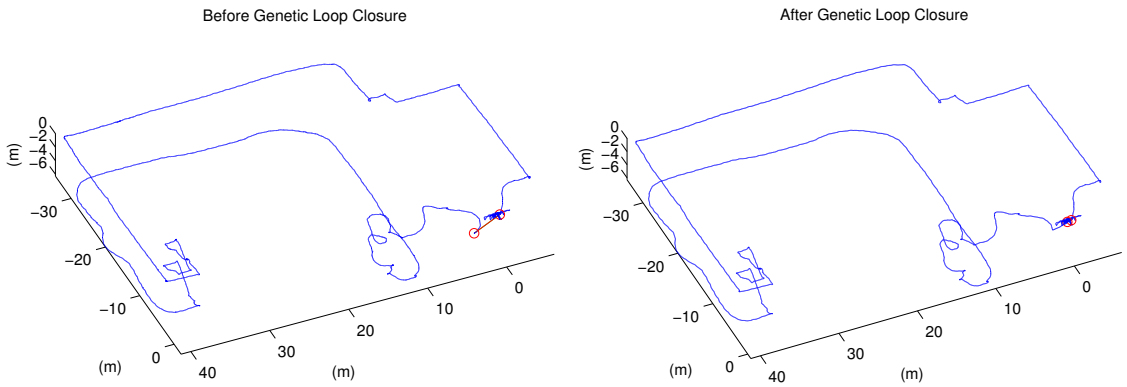


Figure 4.11: An example image of a local submap created from a 6 meter section of a test trajectory. Note that the roof has been manually removed and the points have been colored according to height for visualization purposes.

In the event that the initial condition supplied by the EKF estimator is insufficient for ICP to correctly estimate the relative transformation $\hat{\mathbf{T}}$, we utilize the genetic search strategy described in Section 3.3.2 to do a search over initial conditions. We detect ICP failures by comparing the final ICP alignment error to a threshold. If the final alignment error is deemed to high, we say ICP has failed an fall back on the genetic search algorithm. This strategy is utilized only in the event that standard ICP fails for two reasons. First, our genetic search strategy requires that ICP be run for each candidate initial condition. Since the submaps can contain hundreds of thousands of points, this process is computationally expensive. Furthermore, because the space of 3D transformations contains six parameters, the number of chromosomes, and thus computation time, required is substantially greater than for 2D scan matching.

Despite its limitations, the genetic search is needed to mitigate drift for datasets that contain large interior loops. If the operator traverses a large enough loop without ever revisiting a location, the accumulated drift will grow too large and the initial condition supplied to ICP will fail to converge. In this scenario, the genetic search algorithm is able to successfully converge to the correct transformation $\hat{\mathbf{T}}$ despite a poor initial condition.

Figure 4.12 shows an example of a large interior loop in a two story dataset taken from the interior of an academic building. The operator walked for 10 minutes covering a distance of approximately 500 meters without ever revisiting a location other than the starting location. Figure 4.12(a) shows the odometry results of the EKF estimator. The red circles denote the starting and ending positions while the red line indicates the 4 meters of accumulated drift. Figure 4.12(b) shows the resulting path after genetic loop closure was applied. In this



(a) (b)

Figure 4.12: An example of drift mitigation using the genetic search strategy for ICP. Red circles indicate the starting and ending positions of the trajectory while red lines indicate the accumulated drift. (a): The odometry estimate computed by the EKF estimator. Note that the starting and ending positions are separated by 4 meters of accumulated drift. (b): The resulting path after genetic search ICP was applied.

example, the drift distance was too large for standard ICP, but was completely eliminated by application of the genetic search strategy.

4.8.3 Incremental Graph Optimization

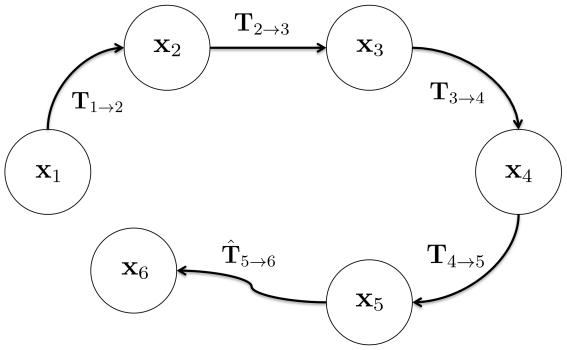
Once loop closure constraints are detected, they are passed to the incremental graph optimization backend. Graph optimization algorithms take as input an initial estimate of the trajectory and a set of loop closure constraints and attempts to find an optimal set of poses subject to a set of transform constraints that relate them. Some variants of graph optimization are able to account for the confidence in the loop closure constraints, however, for our purposes we do not utilize these methods. By constraining poses that are temporally far apart through loop closure, graph optimization enforces long term consistency in the path by finding a new trajectory that minimizes the following objective function:

$$J = \sum_{(i,j)\in\mathcal{G}} \|\mathbf{x}_j - \hat{\mathbf{T}}_{i\to j}(\mathbf{x}_i)\|^2 \tag{4.112}$$

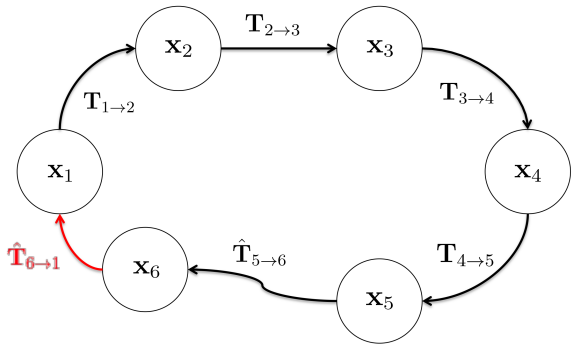
where (i, j) is an edge constraint from the graph $\mathcal{G}$ defined from the odometry and loop closure constraints, $\mathbf{x}_i$ is a pose of the IMU at time instant i that serve as nodes in graph $\mathcal{G}$ and $\mathbf{T}_{i\to j}$ is a constraint that relates pose i and pose j and serve as the edges of $\mathcal{G}$.

Since the EKF has an estimate each time a sensor reading is processed, around 200Hz, we subsample the poses to 20Hz for computational reasons. The graph optimization backend estimates a new trajectory by re-optimizing the set of input poses. We utilize the iSAM2 algorithm [33] implemented in the GTSAM [35] library to perform the optimization.

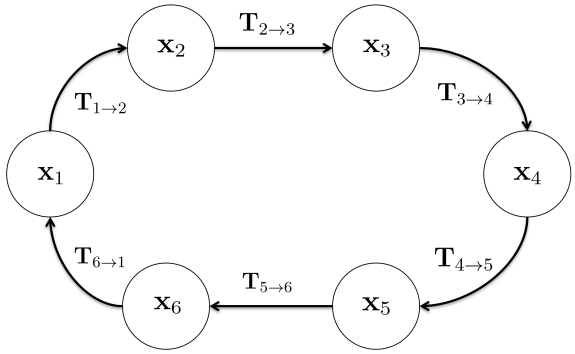
Figure 4.13 shows an example iteration of a fictitious incremental graph optimization problem. Fig. 4.13(a) shows a trajectory containing 5 poses. Odometry estimates from the



(a) Odometry Estimate



(b) Loop Closure Constraint Added



(c) Optimized Trajectory

Figure 4.13: An illustration depicting an iteration of incremental pose graph optimization. (a): Odometry estimates are used to predict the next location $\mathbf{x}_6$. (b): A loop closure constraint $\hat{\mathbf{T}}_{6\to1}$ is used to constrain the location of pose $\mathbf{x}_6$. (c): The optimization backend corrects the estimates of all previous states. Notice that pose $\mathbf{x}_6$ and constraint $\hat{\mathbf{T}}_{5\to6}$ have been corrected.

EKF are used to estimate the location of the next pose $\mathbf{x}_6$. Once $\mathbf{x}_6$ has been estimated, loop closures are detected based on proximity to nearby poses. ICP is then used to generate

an estimate $\hat{\mathbf{T}}_{6\to1}$ of the transform $\mathbf{T}_{6\to1}$ that relates $\mathbf{x}_6$ to $\mathbf{x}_1$. Fig. 4.13(b) depicts the loop closure constraint being added to the graph. Incremental graph optimization is then run to recover an optimized set of poses. Fig. 4.13(c) shows the resulting optimized poses.

We apply iSAM2 to incrementally solve the graph optimization problem similar to the illustrated example. We derive odometry estimates from the EKF trajectory to and incrementally add new poses to the graph optimizer. Each time a loop closure constraint is detected, we add a constraint to the graph and re-optimize the entire past history of poses. This process continues until all odometry measurements have been used. It is important to note that, in practice, if an erroneous loop closure is included or one must be added manually, we must supply that information and then rerun the graph optimization procedure from the beginning to include or exclude manual loop closure constraints.

4.8.4 Point Cloud Generation

After recovering the optimized trajectory using graph optimization, we next construct a point cloud representation of the environment using the range readings from the laser scanners. Each laser range readings is individually timestamped using the best estimates of the laser temporal calibration parameters. In doing so, we directly take into account the rolling shutter of the laser scanners and correct for them. We interpolate the pose to that of the corrected timestamp and convert the range reading into global coordinates using the optimized laser extrinsic calibration. The process is repeated for each laser range reading resulting in a dense point cloud representation of the scanned environment.

4.9 Results

We experimentally verified the proposed multi-sensor fusion algorithm on both simulated and real-world datasets. The results for Monte Carlo simulation and real-world verification of the laser calibration algorithm are presented in Section 4.9.1. Section 4.9.2 details the comparison study of the EKF against both the state-of-the-art MSCKF and the ICP based odometry algorithm of Chapter 3. Lastly, Section 4.9.3 presents end-to-end system results from a number of large scale datasets.

4.9.1 Laser Calibration Results

An important characteristic for EKF based algorithms is the observability of the different state vector components. The observability properties of the IMU body state and camera calibration parameters has already been either experimentally, or theoretically justified [40,51]. In this section, we provide experimental evidence to suggest that the laser calibration parameters, namely the rotation between laser and IMU body coordinates ${}^{B}\mathbf{q}_{L}$, the translation between laser and IMU body coordinates ${}^{B}\mathbf{p}_{L}$, the sensor time delay t_d and the rolling shutter readout parameter t_r, are also observable provided the filter is initialized with a sufficiently accurate initial estimate.

In order to experimentally demonstrate the observability of the laser calibration parameters, we ran a Monte Carlo simulation [135] using 500 trials of a simulated dataset. The

Parameter	Initial Standard Dev.	Final Standard Dev.
$\delta\mathbf{q}$	5°	2.2°
$\delta\mathbf{p}$	20 cm	5.6 cm
$\delta\mathbf{t}_d$	10 ms	3.9 ms
$\delta\mathbf{t}_r$	10 ms	4.8 ms

Table 4.2: Monte Carlo Simulation Results For Laser Calibration

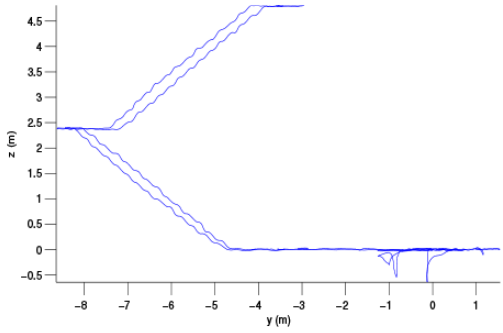
simulated trajectory and environment was chosen to provide rotation and translation across all axis of the system. The sensor noise characteristics were chosen to mimic that of the hardware used in real-world testing. For each trial, a random initial condition for the laser calibration was chosen and the EKF estimator was ran. We then compared the estimated calibration parameters to the ground truth to compute the final error. The covariance of the errors across all converging trials was computed and compared to the distribution of initial conditions.

Table 4.2 shows the results of the Monte Carlo simulation trials. Both the initial error standard deviation and final error standard deviation are reported for each calibration parameter. The results show almost a twofold decrease in deviation for the rotation and temporal calibration parameters. The translation parameters experience an even larger decrease of almost 4×. The results indicate that the EKF is able to decrease the uncertainty in the calibration parameters when the filter converges. We do however note that filter convergence is somewhat sensitive to initial condition as only 62% of the 500 trials converged. In practice, the filter will require a reasonable initial condition of the laser calibration parameters using either a CAD drawing or offline calibration procedure.

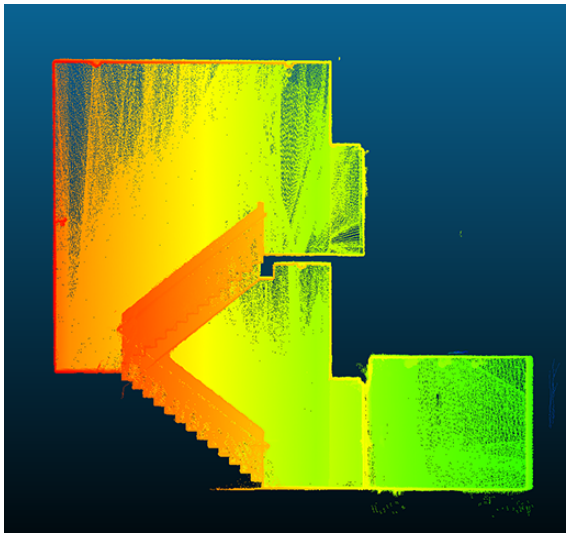
In order to demonstrate the importance of calibrating for the extrinsic and temporal laser calibration parameters, we collected a real-world dataset from a staircase environment. Our EKF data fusion algorithm was then run with and without optimizing the calibration parameters. The resulting geometry was then visually inspected to verify the results.

Figure 4.14 demonstrates the importance of calibrating the extrinsic and temporal laser calibration parameters. Fig. 4.14(a) shows the reconstructed trajectory from the small staircase environment. Fig. 4.14(b) shows a cross section of the reconstructed point cloud for reference. Fig. 4.14(c) shows the results of using only the CAD model derived calibration. Fig. 4.14(d) shows a comparison of the reconstructed point cloud if the EKF is allowed to optimize the laser calibration parameters starting from the CAD estimates. Notice that when calibration parameters are not optimized by the EKF, the algorithm is unable to account for timing errors and intra-scan warping. The resulting point cloud is clearly less accurate than when accounting for the calibration parameters in the EKF estimator.

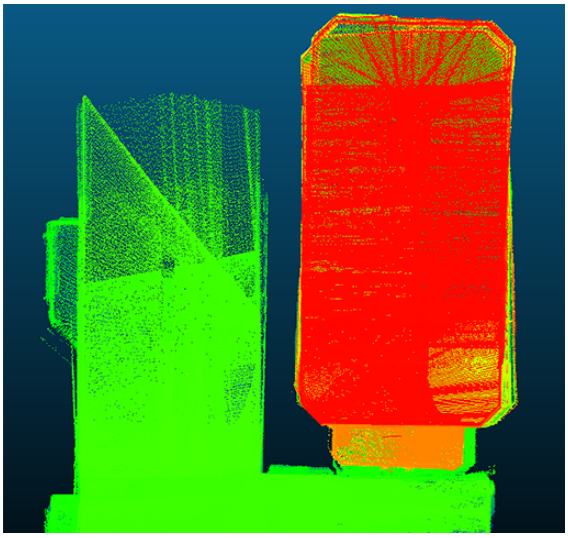
Figure 4.15 shows the covariance convergence plots for the results of applying the EKF estimator to the dataset shown in Fig. 4.14. The red lines in each plot indicate the 3σ confidence interval and the blue lines indicate the estimate of the error state. The results indicate that the rotation and timing parameters are rapidly determined by the EKF estimator due to the rapid convergence of the error state covariance. The translational components experience a slower convergence, but result in an error bound of approximately 4 mm at the end of the dataset. The results of Figure 4.15 indicate that the EKF estimator is able to



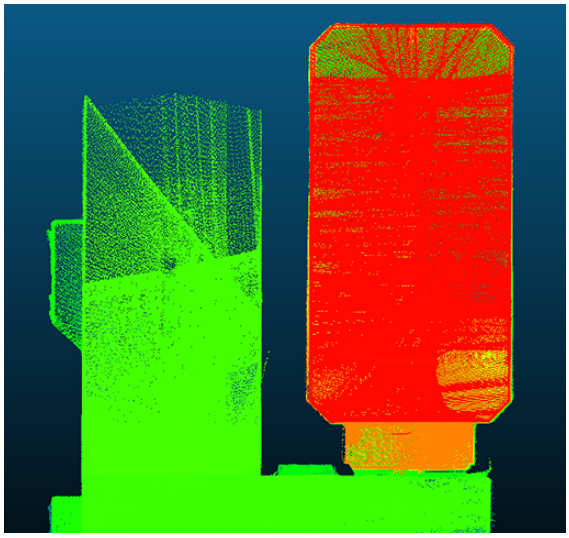
(a) Estimated Trajectory



(b) Reconstructed Point Cloud



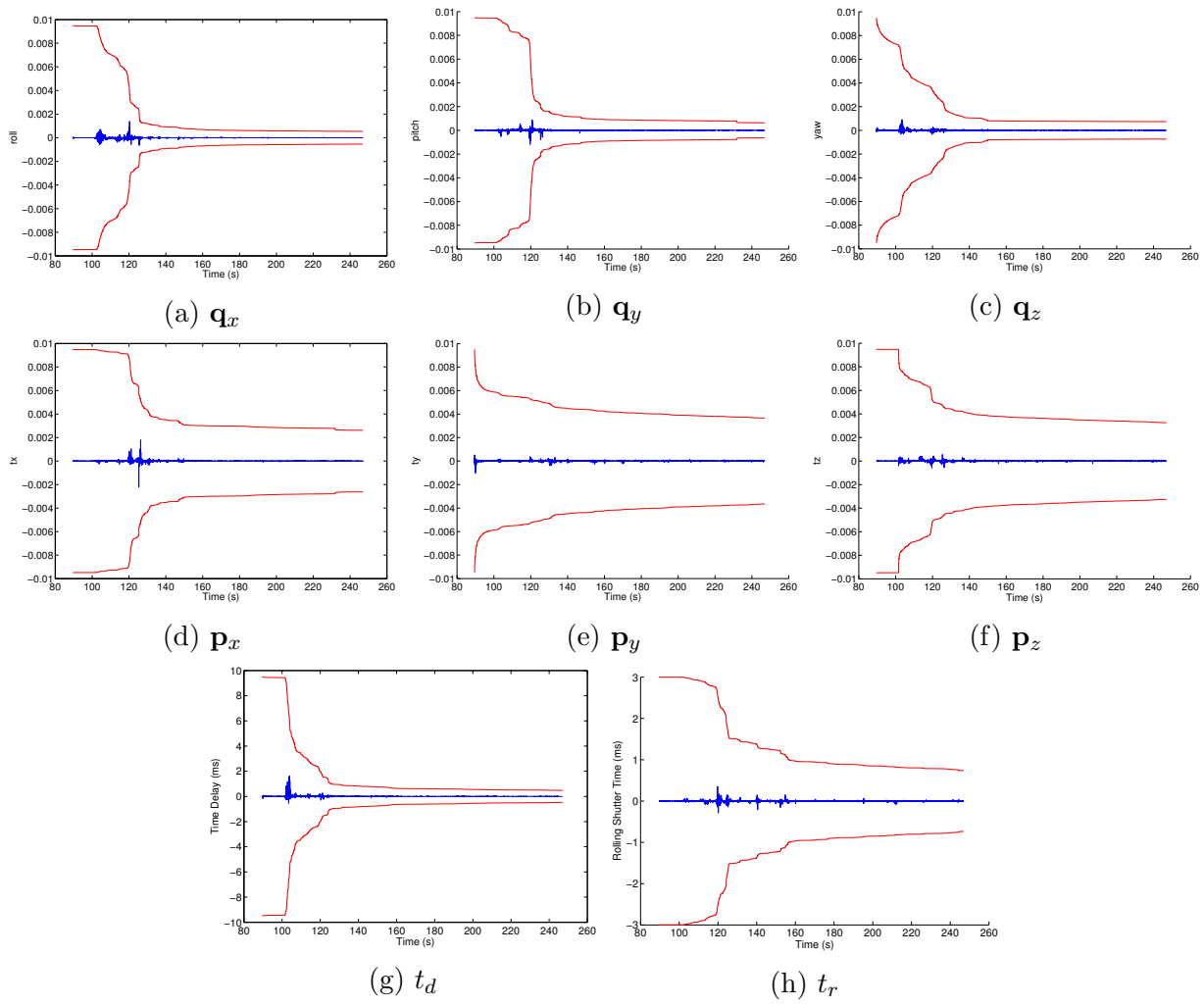
(c) Results Without Calibration



(d) Results With Calibration

Figure 4.14: A real-world example demonstrating the importance of temporal and extrinsic calibration of the laser parameters. (a): The trajectory estimated by the EKF. (b): A cross section of the reconstructed point cloud. (c): A close up of the reconstructed environment without optimized calibration parameters. (d): A close up of the reconstructed environment with optimized calibration parameters.

reduce the variance on the calibration parameters in both simulated and real-world datasets. It is important to note that the majority of the variance reduction happens at the start of dataset. This is attributed to the fact that a small "calibration sequence" is intentionally applied to excite all degrees of freedom of the system as described in Section 4.7.



(a) $\mathbf{q}_x$ (b) $\mathbf{q}_y$ (c) $\mathbf{q}_z$

(d) $\mathbf{p}_x$ (e) $\mathbf{p}_y$ (f) $\mathbf{p}_z$

(g) t_d (h) t_r

Figure 4.15: The covariance convergence plots for the laser calibration results from the dataset shown in Figure 4.14. The red lines indicate the 3σ bounds for the error state variance. The blue lines indicate the estimated error state at each EKF update. Note that all parameters experience a marked reduction in covariance as different axis of the system are excited. (a)–(c): The rotation calibration parameters. (d)–(f): The translation calibration parameters. (g): The time delay between the IMU and the laser scanner. (h): The rolling shutter parameter for the laser scanner's readout time.

4.9.2 Odometry Comparison Results

Next, we compare our multi-sensor EKF estimator against both the popular MSCKF algorithm [111] and the ICP based method of 3. In order to maintain a fair comparison, we only compare the odometry results across the three algorithms. No batch optimization or incremental graph optimization was used for this comparison. Two datasets were collected using our backpack data collection system. The first dataset was collected from Sutardja Dai Hall on University of California, Berkeley campus and totaled 160 meters. The second dataset was taken from Cory Hall, an academic building on University of California, Berkeley

	ICP [8]	MSCKF [111]	Laser+IMU	Laser+Camera+IMU
Odometry Dataset 1: Sutardja Dai Hall				
Estimated Distance (m)	152.39	170.50	161.49	160.93
Drift Distance (m)	3.90	1.07	0.43	0.41
Drift Rate (%)	2.56	0.63	0.26	0.25
Odometry Dataset 2: Cory Hall				
Estimated Distance (m)	591.60	644.12	605.38	602.90
Drift Distance (m)	13.61	8.57	0.61	0.48
Drift Rate (%)	2.30	1.33	0.10	0.08

Table 4.3: Odometry Comparison Results.

campus, and totaled 600 meters in length. In both dataset, the operator traveled through an indoor environment ensuring to start and stop in the same location and orientation. By comparing the first and last estimated poses and normalizing by distance traveled, we obtain an estimate of the drift rate of the algorithm on each dataset.

Table 4.3 shows the results of the odometry comparison. As expected, the ICP based odometry method of Chapter 3 performs the poorest. Not only does it appear to underestimate the total distance, it also has the highest drift rate of any of the algorithms. Comparing the EKF fusion algorithms, we note that using the IMU and lasers produces a lower drift rate than the MSCKF algorithm. This result matches our intuition that because the lasers are capable of measuring absolute distances to the environment, they provide a more accurate odometry estimate than the camera which an lacks independent observation of absolute scale. Furthermore, as expected we see that including both data sources in our estimates provides the lowest drift rate amongst the compared algorithms.

These two datasets were chosen because they represent the expected use case of the system. The first dataset, shown in Figures 4.16(a) and (c), is along a long narrow hallway while the second dataset, shown in Figures 4.16(b) and (d), are taken from an office type environment. Long, narrow hallways are challenging environments for laser based algorithms and thus the drift rate is higher for the Laser+IMU algorithm in the first dataset. On the other hand, office type environments provide many planes that can be utilized by the laser data fusion algorithm and thus the Laser+IMU obtains an even lower drift rate on the second dataset.

Figure 4.16 shows the estimated trajectory and environment map for both of the odometry comparison datasets. Figures 4.16(a) and 4.16(c) show the top-down and 3D views of Odometry Dataset 1 and Fig. 4.16(b) and 4.16(d) show the top-down and 3D views of Odometry Dataset 2. Notice that in Fig. 4.16(a) some of the planes near the top of the environment have been marginalized away due to the plane management strategies discussed in Sec. 4.6.6. However, we note that despite the incomplete plane map, the algorithm only has a 0.08% drift rate on this dataset.

Next, we compare estimated error state covariance between the MSCKF algorithm and

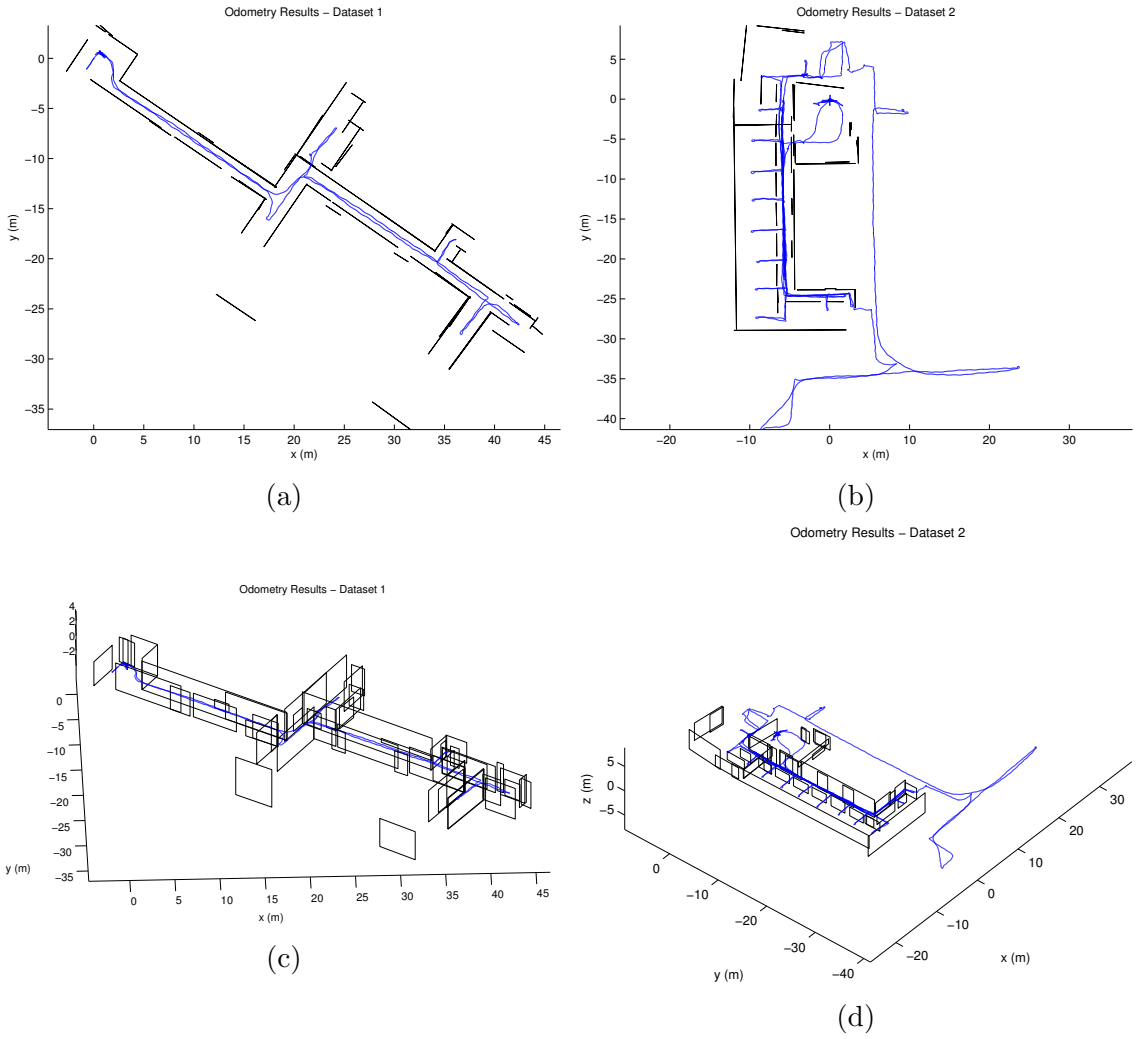
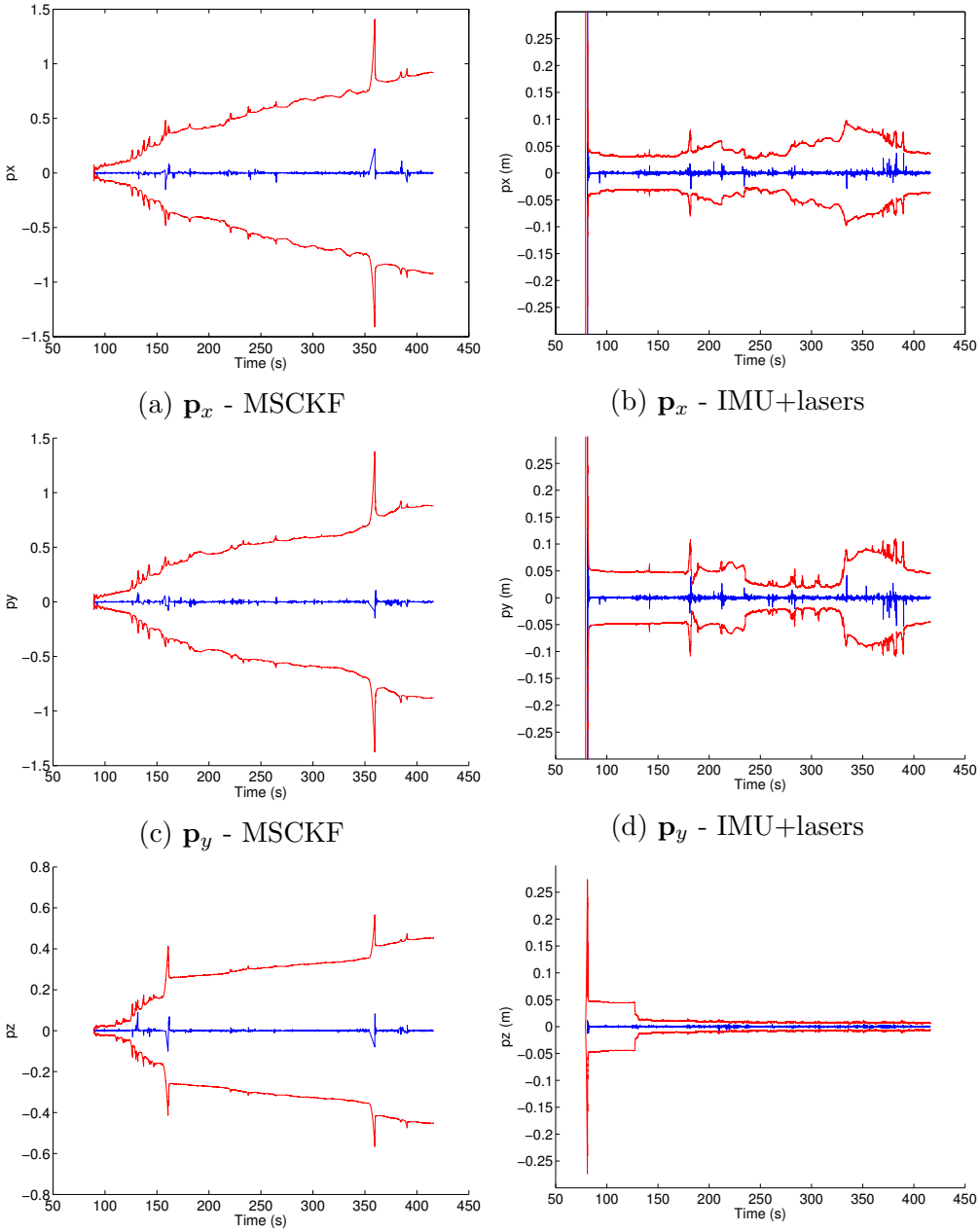


Figure 4.16: The reconstructed trajectories and environment maps of the odometry comparison datasets. Results are shown for the case of IMU+camera+lasers. Notice that some planes have been marginalized away and are missing from the plane map. (a): Top-down view of Odometry Dataset 1. (b): Top-down view of Odometry Dataset 2. (c): 3D view of Odometry Dataset 1. (d): 3D view of Odometry Dataset 2



(a) $\mathbf{p}_x$ - MSCKF

(b) $\mathbf{p}_x$ - IMU+lasers

(c) $\mathbf{p}_y$ - MSCKF

(d) $\mathbf{p}_y$ - IMU+lasers

(e) $\mathbf{p}_z$ - MSCKF

(f) $\mathbf{p}_z$ - IMU+lasers

Figure 4.17: Comparison results for the positional components of the error state standard deviation for the MSCKF and IMU+lasers cases. The red lines indicate the 3σ bounds while the blue lines indicate the estimate of the error state. (a),(c),(e): The standard deviation for the MSCKF case. (b),(d),(f): The standard deviation for the IMU+lasers case. After an initialization phase the standard deviation does not grow unbounded for the case of the IMU+lasers. Note the scale difference among the plots.

the proposed EKF when using only IMU+lasers. The position of the IMU body frame in VIO algorithms such as the MSCKF is not observable and thus we expect to see an unbounded growth in the error state variance of the position components. On the other hand, whenever our laser+IMU based EKF estimator revisits a location and re-observes a mapped plane, the EKF is able to reduce the positional variance. Figure 4.17 shows the graphs of the error state standard deviation for the positional components for both the MSCKF and IMU+laser algorithms on Odometry Dataset 1.

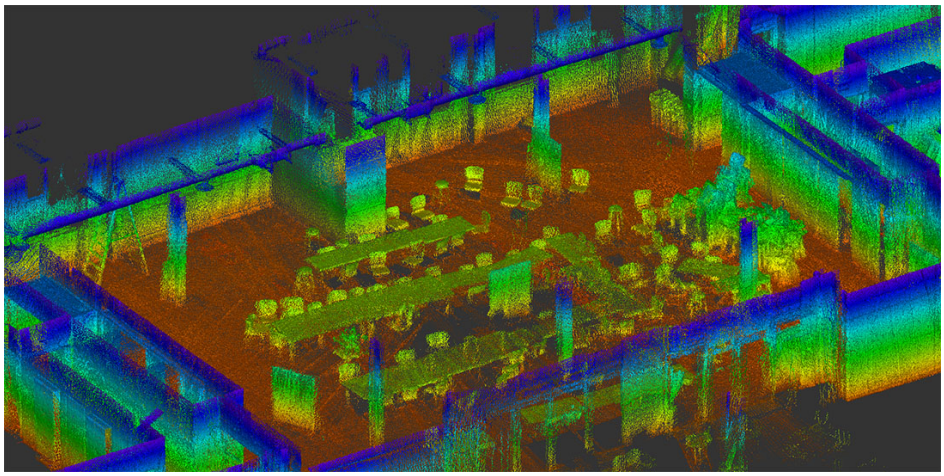
Figures 4.17(a),(c),(e) show the standard deviation of the positional components of the state vector across the dataset for the MSCKF algorithm. Note that the error grows unbounded for the entirety of the dataset. Figures 4.17(b),(d),(f) show the same plots but for the case of IMU+lasers. Notice that unlike the IMU+camera only configuration, since the lasers are able to map structural elements of the building, when the system revisits a location the EKF is able to reduce the error in the position. Note the scale difference different among the plots shown in Figure 4.17.

4.9.3 End-To-End System Results

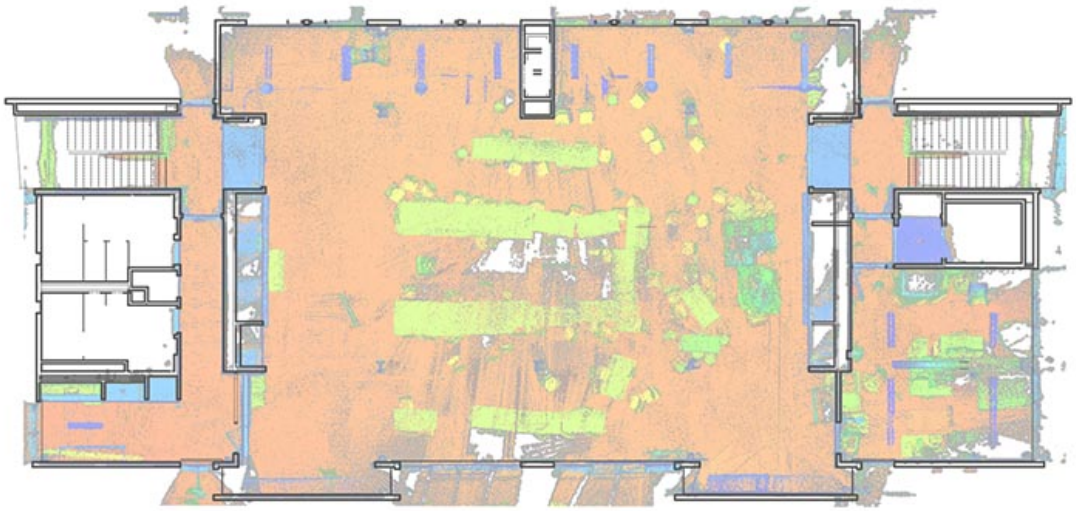
Finally, we test the scalability of our algorithm using 3 large-scale datasets against the end-to-end system of Chapter 3 [8]. Since it is difficult to obtain absolute ground truth positioning and a GPS comparison is unreliable in indoor environments, we compute the generated point clouds for each of the 3 large-scale datasets using both methods and compare against CAD drawings of the buildings. The recovered point clouds are then manually aligned to the CAD drawings' coordinate frames. We compare the scanned area against the CAD drawings to compare both the floor area scanned and the Intersection Over Union (IOU) between the two shapes. An IOU of 0 indicates that the two shapes are completely disjoint while an IOU indicates that the shapes are exactly identical. The results of this comparison test for all three large-scale datasets are summarized in Table 4.4

For dataset 1 of Table 4.4, we collected data from 4 floors of Jacobs Hall on the University of California, Berkeley campus. The operator walked for 20 minutes and covered a total walking distance of approximately 860m. The 4 floors were connected via 2 sets of stairwells located on opposite sides of the building. Figure 4.18 and 4.19 show the results from the first dataset. Fig. 4.18(a) shows a close up view of the reconstructed point cloud with the ceiling removed for visualization. Many detailed features such as ladders, tables, and chairs are clearly visible. Fig. 4.18(b) shows a comparison between the CAD drawings and a height projection of the point cloud with the ceiling removed. The black lines indicate the CAD derived floor plan while the colored points are from the reconstructed point cloud. Using the CAD estimate as a ground truth, the proposed method estimates the area of the building accurate to 97% and our reconstructed environment shares an IOU with the CAD drawings of 0.962. In comparison, the method of Chapter 3, only has an area accuracy of 89.2% and an IOU of 0.890. Figures 4.19(a) and 4.19(b) show 3D and top-down views of the reconstructed trajectory. Notice that the stairwells on each floor are vertically aligned into tight spirals.

The second large-scale dataset used was collected from 3 floors of Cory Hall on the University of California, Berkeley campus. During data collection the operator walked 1.1km over the duration of 23 minutes. Figures 4.20 and 4.21 show the results from dataset 2. Figure 4.20(a) shows a close up view of the point cloud. Note that small building features

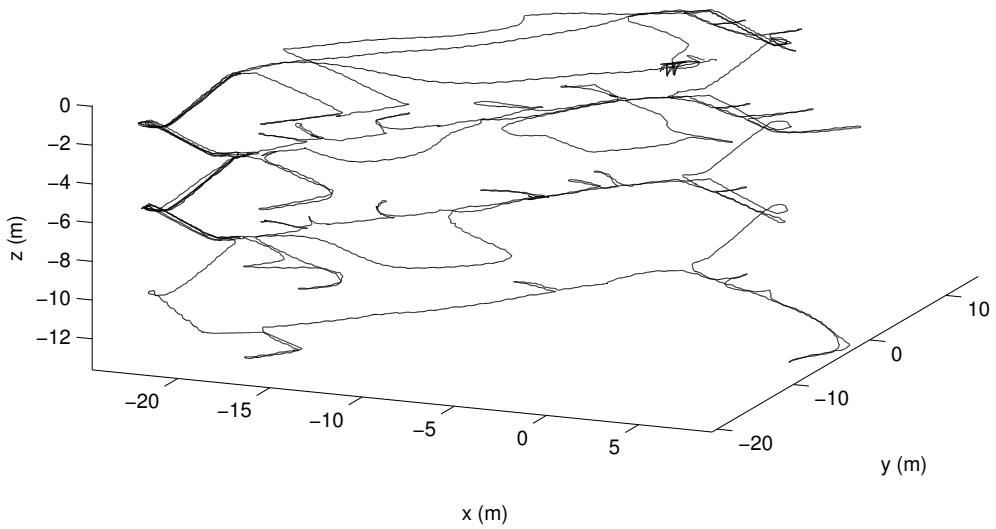


(a) Closeup View of Point Cloud

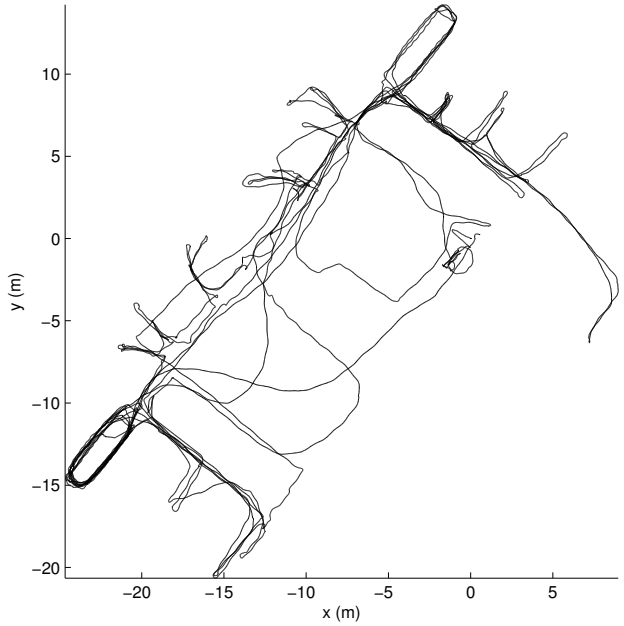


(b) Single Floor CAD Overlay

Figure 4.18: Results from large-scale dataset 1. (a): A close up view of one section of the reconstructed point cloud. (b): The CAD floor plan overlain on one floor of the reconstructed point cloud The black lines correspond to the CAD derived floor plan and the colored points are the height projection of the reconstructed point cloud after the ceiling has been removed.

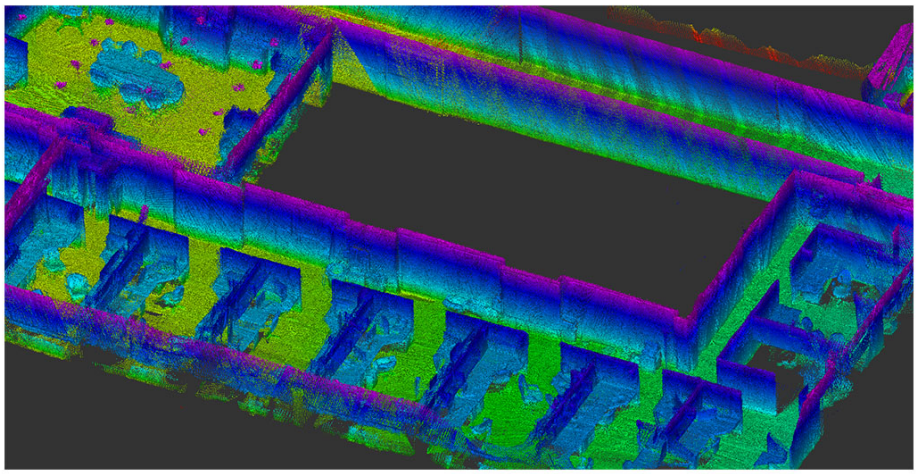


(a) 3D Trajectory View

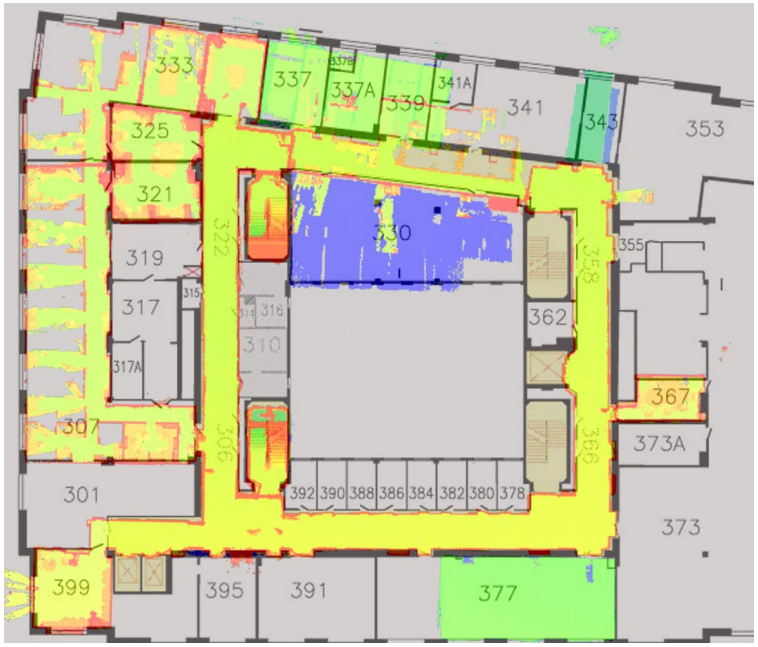


(b) Top-down Trajectory View

Figure 4.19: Results from large-scale dataset 1. (a): A 3D view of the reconstructed trajectory. (b): A top-down view of the dataset. Notice that the stairwell regions are nicely aligned into tight spirals.

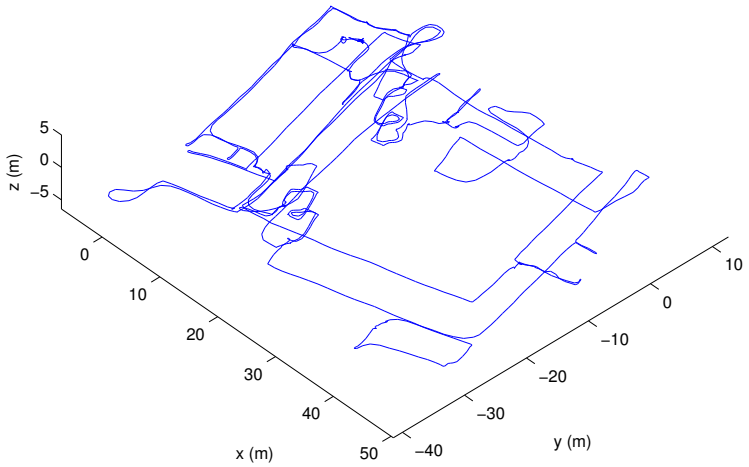


(a) Closeup View of Point Cloud

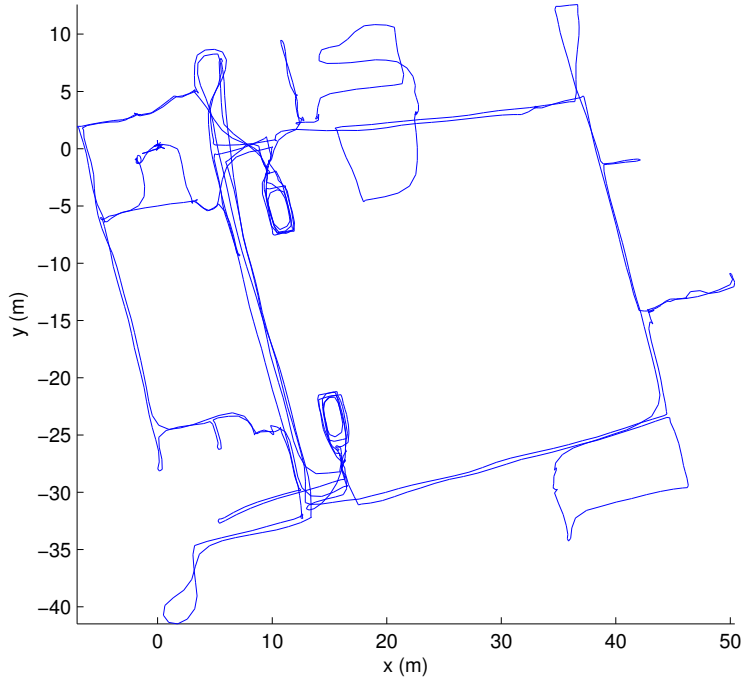


(b) Single Floor CAD Overlay

Figure 4.20: Results from large-scale dataset 2. (a): A close up view of a section of the reconstructed point cloud. Note that building elements such as desks, trashcans and computer monitors are visible in the reconstruction. (b): The CAD floor plan overlain on one floor of the reconstructed point cloud. The black lines correspond to the CAD derived floor plan and the colored points are from the projection of the reconstructed point cloud after the ceiling has been removed.

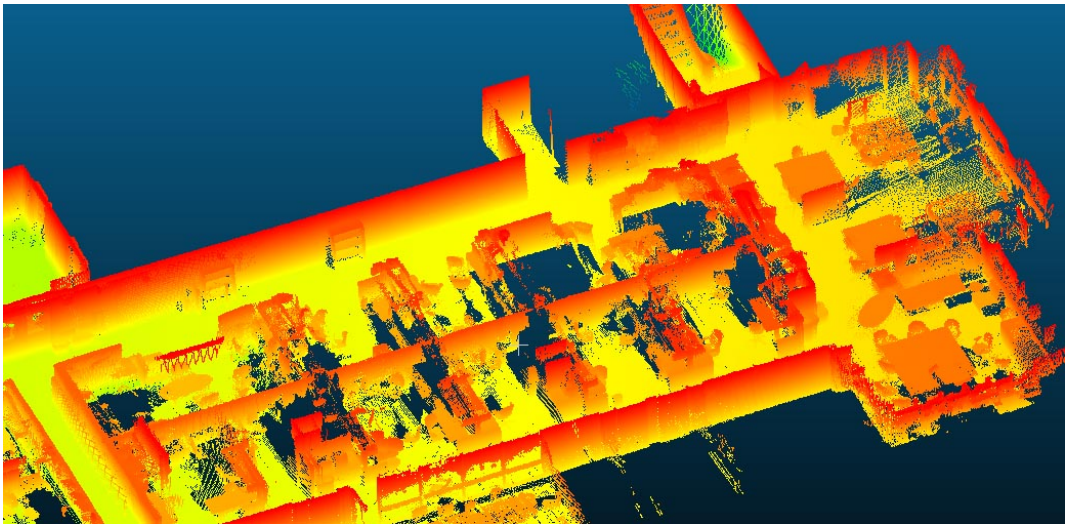


(a) 3D Trajectory View

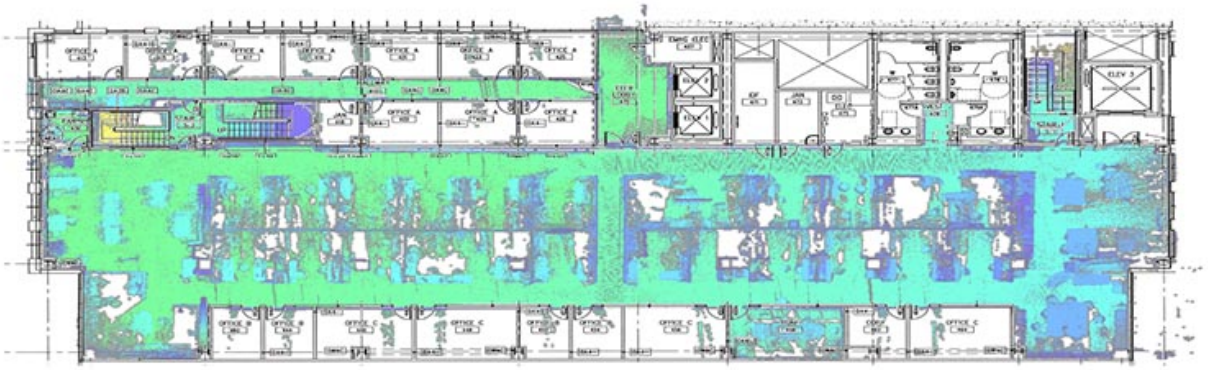


(b) Top-down Trajectory View

Figure 4.21: Results from large-scale dataset 2. (a): A 3D view of the reconstructed trajectory. (b): A top-down view of the dataset. Notice that hallways on all floors are nearly perfectly aligned vertically.

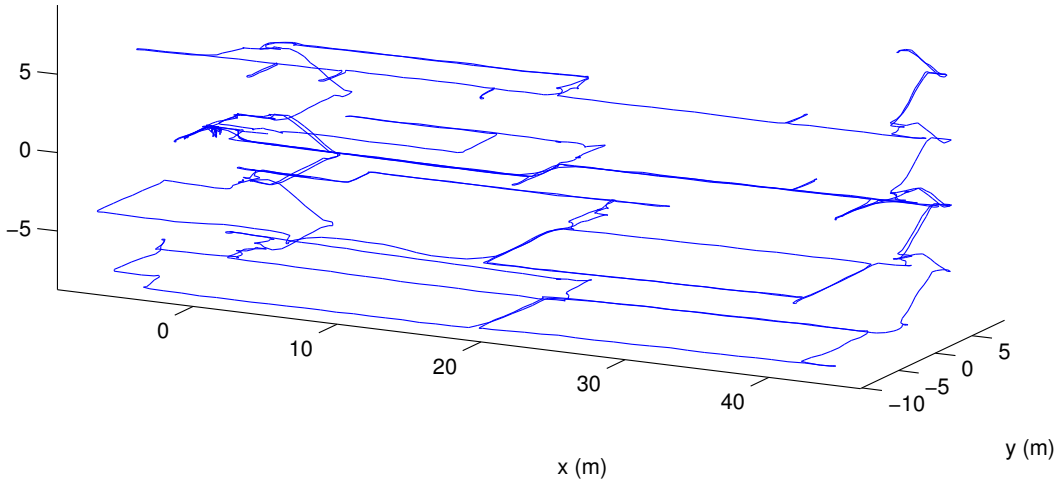


(a) Closeup View of Point Cloud

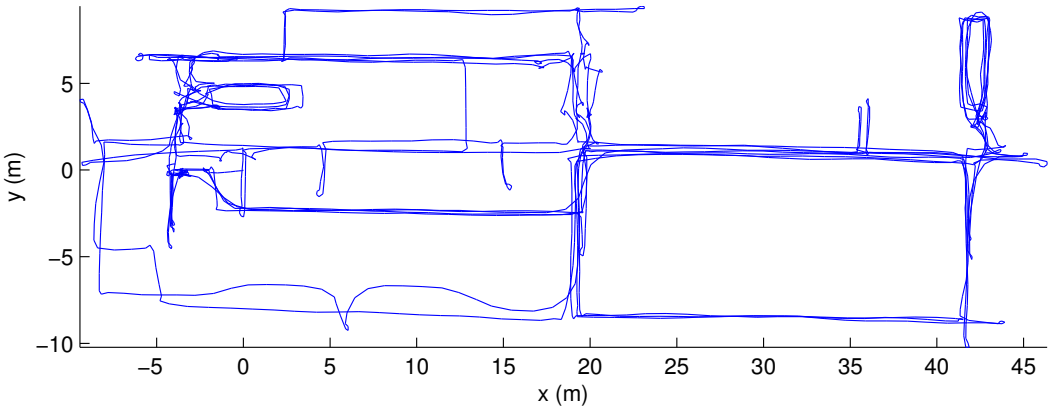


(b) Single Floor CAD Overlay

Figure 4.22: Results from large-scale dataset 3. (a): A close up view of one section of the reconstructed point cloud. (b): The CAD floor plan overlain on one floor of the reconstructed point cloud. The black lines correspond to the CAD derived floor plan and the colored points are from the projection of the reconstructed point cloud after the ceiling has been removed.



(a) 3D Trajectory View



(b) Top-down Trajectory View

Figure 4.23: Results from large-scale dataset 3. (a): A 3D view of the reconstructed trajectory. (b): A top-down view of the reconstructed trajectory. Notice how all hallways are vertically aligned between floors.

Large-scale Dataset	Duration	Distance	Area Acc. [Proposed]	IOU [Proposed]	Area Acc. Chapter 3	IOU Chapter 3
1	1200s	860m	97.0%	0.962	89.2%	0.890
2	1380s	1100m	96.4%	0.986	93.2%	0.808
3	1644s	1200m	99.6%	0.984	99.1%	0.917

Table 4.4: Large-scale dataset results.

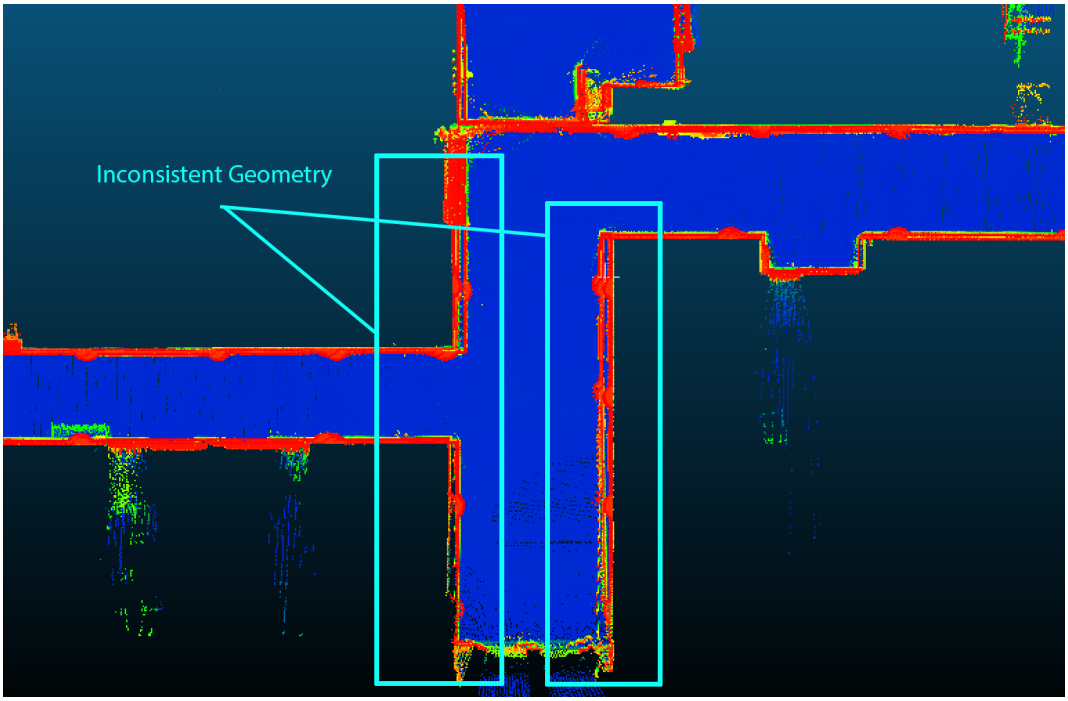
such as desks, computer monitors, and trashcans are clearly visible in the reconstruction. Figure 4.20(b) shows the a vertical projection of the point cloud overlain on the CAD drawings. For this dataset proposed method estimates the area of the scanned rooms accurate to 96.4% and has IOU of 0.986 shared with the CAD floor plans. Figures 4.21(a) and (b) show the 3D and top-down views of the reconstructed trajectory. Notice that the hallways on all floors are nearly perfectly aligned vertically.

Our last large-scale dataset was taken from a five story building, Sutardja Dai Hall, on the University of California, Berkeley campus containing offices, long narrow hallways, and large open areas connected by two sets of stairwells located on opposite sides of the building. The walking time for this dataset was 27.4 minutes and the trajectory was estimated to be 1.2km in length. Figures 4.22 and 4.23 show the results from this dataset. Figure 4.22(a) shows a close up view of one section of the reconstructed point cloud. Figure 4.22(b) shows the CAD overlay of a the ground truth floor plan on a single floor of the reconstructed point cloud. In this dataset, both methods estimate the area to roughly 99% accuracy, but the proposed method shares an IOU of 0.984 compared to the previous method's 0.917. Figures 4.23(a) and 4.23(b) show 3D and top-down views of the reconstructed trajectory. Note that hallways on all floors nicely align in the trajectory.

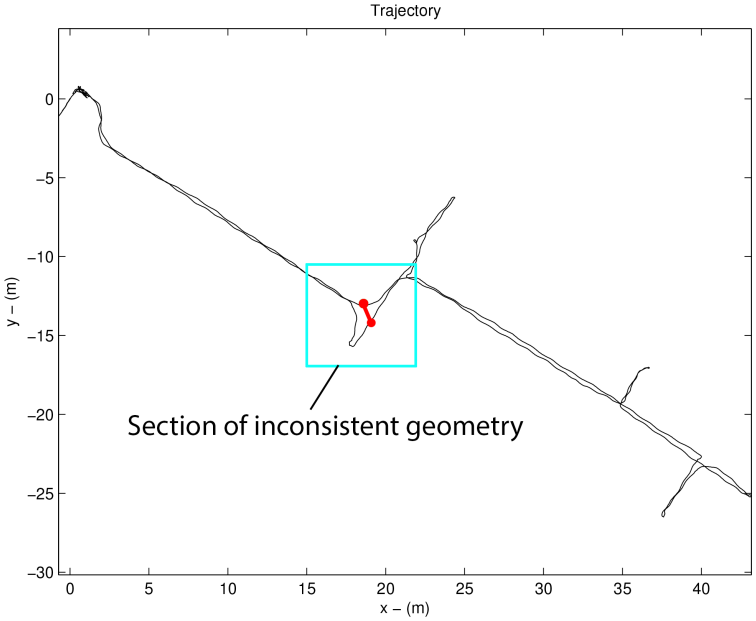
It is important to note that for this dataset two manually defined loop closures were required for correct operation. During the course of data collection, the operator inadvertently occluded the sensors by allowing doors to close while the laser scanners were still scanning them. Since the EKF estimator implicitly assumes that all large planar objects are static across time, including them into the estimation process added additional drift that was unable to automatically be mitigated using the loop closure strategies of Section 4.8.2.

Manual loop closures are added by first inspecting the point cloud produced by the 3D localization algorithm and noting any regions of inconsistent geometry. Then, the sections of the reconstructed trajectory that generated the inconsistent geometry are identified and a loop closure constraint is manually specified by running the genetic 3D ICP algorithm of Section 4.8.2 between submaps nearby to that region. Figure 4.25 shows an example of a point cloud containing geometric inconsistency and the corresponding section of the trajectory. Figure 4.25(a) shows two sections of wall geometries that have multiple unaligned copies. Figure 4.25(b) shows the corresponding section of trajectory. The red line indicates the manual loop closure that was chosen to fix this example. Notice that without visually inspecting the point cloud, it is not obvious that a manual loop closure is required in this section of the trajectory.

The resulting loop closure constraint is then manually added to previously computed loop closure constraints and the graph optimization process is rerun. Given that the loop



(a) Inconsistent Geometry In Point Cloud



(b) Corresponding Section of Trajectory

Figure 4.24: An example of a point cloud that requires manual loop closure due to geometric inconsistency. (a): A section of point cloud containing multiple unaligned copies wall geometry. (b): The corresponding section of the trajectory is highlighted in cyan. The red line indicates the location of the manual loop closure constraint used to fix this example.

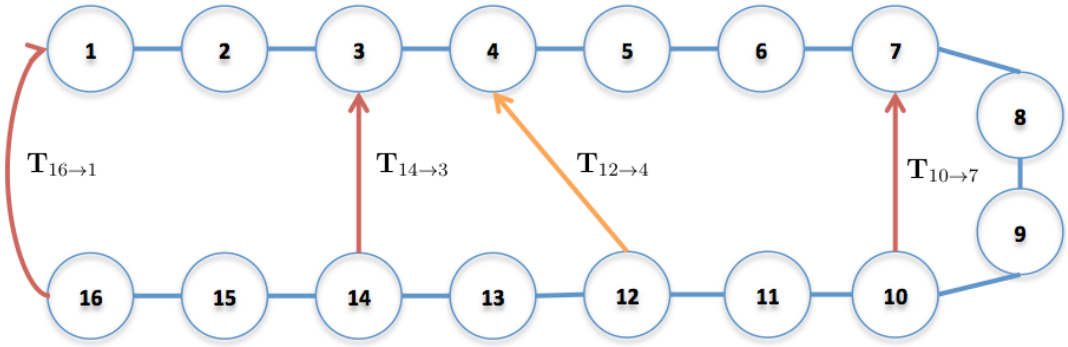


Figure 4.25: An illustrative example of manual loop closure re-optimization. When the manual loop closure, denoted in orange.

closure constraints are processed in a fixed order, the incremental optimization algorithm is Markovian, i.e. only the state of the graph optimization at time n and subsequent constraints are required to compute the next iteration of optimization. We exploit this property to speed up computation by saving the state of the optimization each time a new loop closure constraint is added so that we can restart the optimization from any arbitrary time. In doing so, we avoid unnecessary computation by restarting the optimization from the location of the manual loop closure and then only re-optimizing for subsequent constraints. Since incremental graph optimization scales approximately with $\mathcal{O}(n^{1.5})$ for SLAM problems [33], for m manually defined loop closures the cost of re-optimization is at worst $\mathcal{O}(mn^{1.5})$.

Figure 4.25 shows an illustrated example of graph re-optimization from a manually defined loop closure. The blue circles and arrows indicated the poses and odometry constraints. The red and orange arrows denote automatically and manually detected loop closures respectively. When the manual loop closure $\mathbf{T}_{12\to4}$ is inserted between poses 12 and 4, only the contributions from constraints that occur after pose 12, namely $\mathbf{T}_{14\to3}$ and $\mathbf{T}_{16\to1}$, must be recomputed. As long as the state of the graph from when pose 12 was added is known and the constraints are processed in the same order, the contributions from previous constraint $\mathbf{T}_{10\to7}$ do not need to be reprocessed.

Once the manual loop closure has been incorporated into the trajectory, the process of visual inspection and manual loop closure is iterated until the resulting point cloud no longer contained obvious geometric inconsistencies. The resulting two manual loop closures needed to correct for the problems of Large-Scale Dataset 3 took less than 20 minutes of human intervention. Since we efficiently cache the state of the optimization each time we add a loop closure constraint, re-optimizing the trajectory takes a negligible amount of processing time. Although processing the constraints at the same time via a single instance of batch optimization would be computationally more efficient, manual loop closures are added one at a time via post-processing and thus the incremental graph optimization significantly reduces the required processing time. For more discussion on limitations and best practices for scanning, see Section 4.10.1.

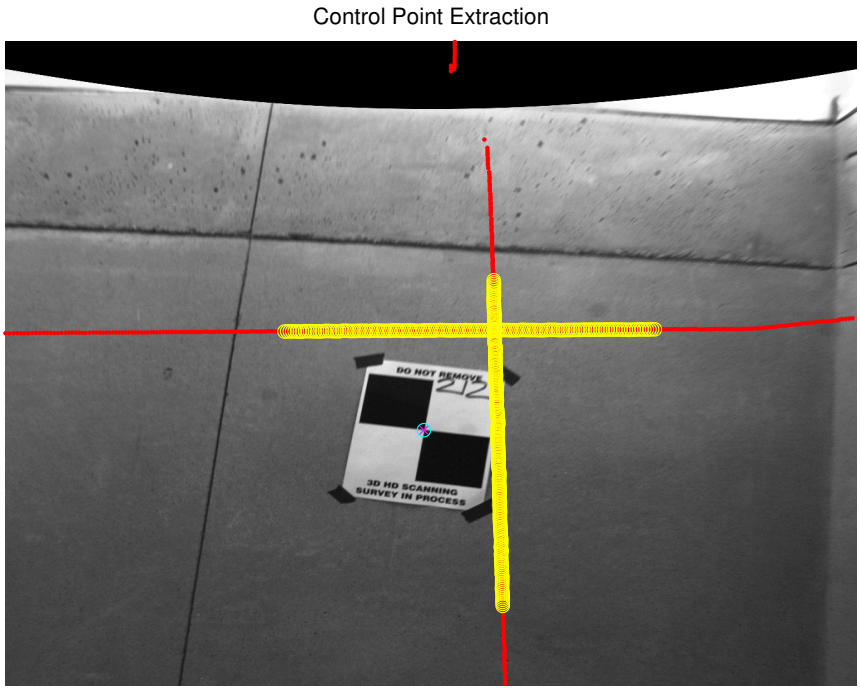


Figure 4.26: An example of control point estimation using laser and camera data. Once a control point is located, the center of the target was manually annotated by a pink ×. Then, the temporally closest 2D laser scans from our backpack system, denoted in red, were projected into the image and a subset of the points, denoted in yellow, were manually used to estimate the plane of the target. The reprojection of the estimated control point, shown in cyan, was used to visually verify the quality of the estimate.

4.9.4 In-Depth Analysis of Large-Scale Dataset 1

In this subsection, we provide in-depth analysis of Large-Scale Dataset 1. In order to characterize the global accuracy of the end-to-end 3D localization algorithms, we repeated the static scanning, control points experiment described in Section 3.5.6. Using a FARO Focus 3D, 43 high-density scans were automatically stitched together using 83 checkerboard targets scattered throughout the environment. The (x, y, z) ground truth locations of the center of the targets were extracted using the 3D static scanner to a reported accuracy of approximately 1 cm. We then ran the end-to-end system to create point cloud of the building using the methods of this chapter with the backpack system.

Each time a control point was visible in the system's camera imagery, we extracted an estimate of the control point for comparison against the surveyed ground truth values. Figure 4.26 shows an example of the control point extraction process. Once a control point was located in an image, the center of the target, denoted by a pink × was annotated manually. Then, the temporally closest 2D laser scans from our backpack system, denoted in red, were projected into the image using the optimized calibration values computed by the EKF. The operator then manually selected a subset of the laser points, denoted in yellow, lying on the same plane as the targeted point. The target's plane relative to the camera was fit using Principal Component Analysis [103] (PCA) and then ray-plane intersection

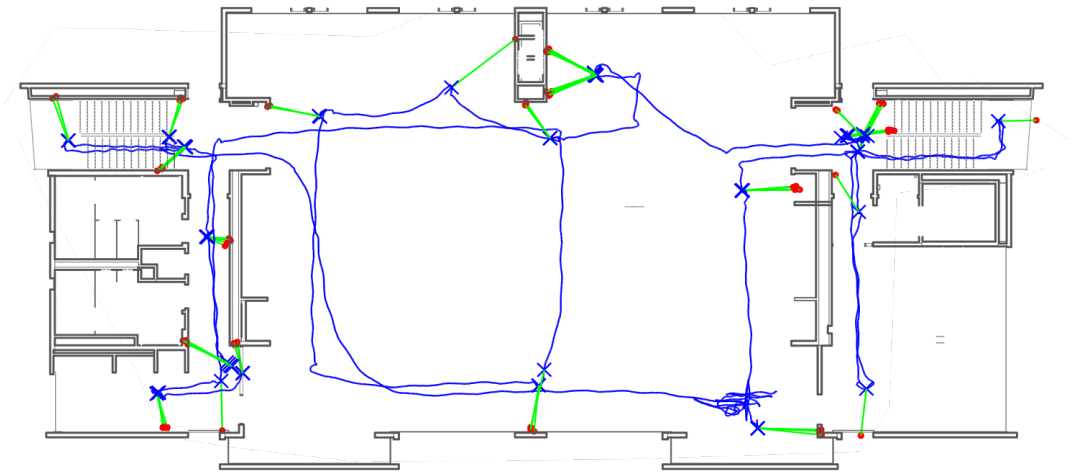
was used to calculate the depth of the target into the scene. This process, combined with the estimate of the cameras location and orientation, allows us to compute an estimate for the (x, y, z) location of the center of the target. The reprojection of the estimate, denoted as a cyan $\circ$, was used to visually verify the quality of the extracted control point. In order to minimize potential error in this process, only those images where the target was viewed head-on were used.

Performing control point extraction for all images across all floors of the dataset yielded 377 observations of the 83 control points scattered throughout the environment. The estimates of the control points were then aligned against the ground truth values using least-squares to compute the error in each observation. Intuitively, least-squares alignment must be applied because the internal coordinate system between the ground truth points and the coordinate system of the backpack's estimated points must be aligned.

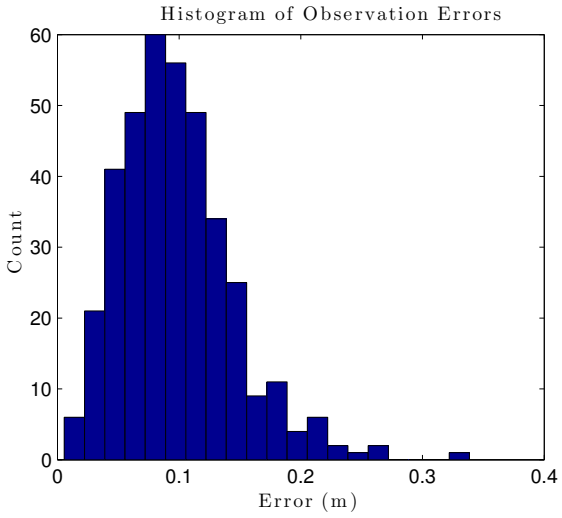
Figure 4.27 shows the results of the control point observation experiment. Figure 4.27(a) shows the estimated locations of the control points for a single floor of the dataset overlain on the CAD drawings of the building. The estimated trajectory of the system is shown in blue while the control points estimates are shown in red. Green lines drawn from the control points to the path denote a control point observation. We observe from this figure that the control points were evenly spread throughout the environment to not introduce any bias due to their locations. Figure 4.27(b) shows the histogram of the observation error between the control points estimated by the static scanning approach and the proposed algorithm. The average observation error was found to be 9.7 cm from the 377 observations of the 83 control points. In addition, the maximal error was found to be 33.8 cm while the standard deviation was computed to be 4.4 cm.

Table 4.5 shows the results computed by floor of the building. It is important to note that the results from Basement 1 were significantly better than the other floors of the building. This can attributed to the fact that in contrast to the basement, all other floors were primarily constructed of glass. Glass can appear either reflective, transparent, or opaque depending on its angle relative to the scanner. This makes detecting and tracking planes more difficult in these environments. When the algorithm is unable to utilize laser based feedback it must rely on only the camera data fusion algorithm for positioning. By reviewing Table 4.3, we see that the camera data fusion algorithm has a substantially larger drift rate and thus we see decreased performance on floors of the building made primarily of glass. In addition, Floor 1 was particularly difficult due to the large amount of foot traffic present during data acquisition. Dynamic objects present in the camera imagery and laser data violate the assumptions of the data fusion algorithms and contributes to additional drift during the reconstruction process.

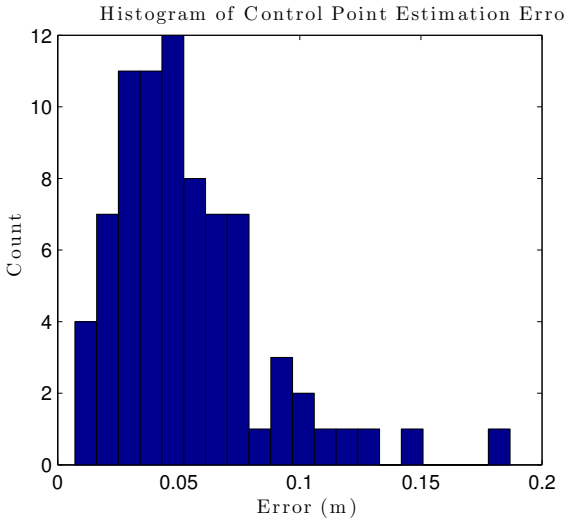
In order to make a few key observations about the results of Table 4.5, we examined the Basement 1 to understand the major remaining sources of error in the estimated trajectory. Figure 4.28(a) shows the estimated trajectory and the direction and estimated locations of each observation of a control point. The size of the red dots have been sized to reflect the maximal error in estimating each control point. Based on this plot, we conclude that with the exception of a few control points, the maximal estimation error for each point is fairly homogeneous. This can be attributed to the fact that least-squares alignment was used to align the coordinate systems of the data. Figure 4.28(b) shows the corresponding histogram of observation errors for this set. From the histogram we can easily see that there is a cluster



(a) Control Point Estimated Locations

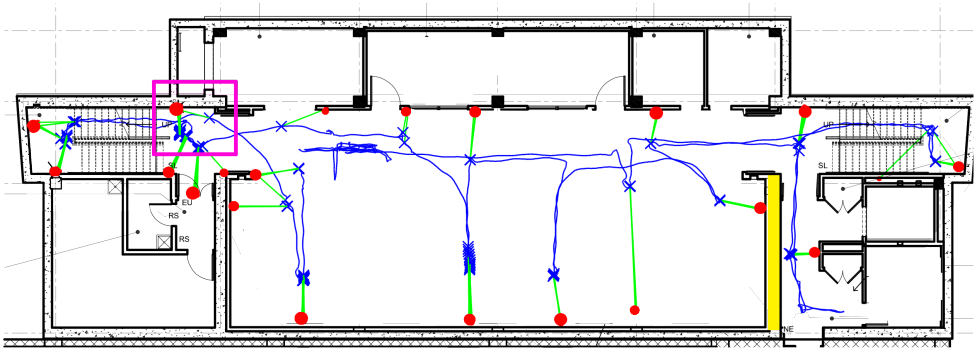


(b) Histogram of Observation Errors

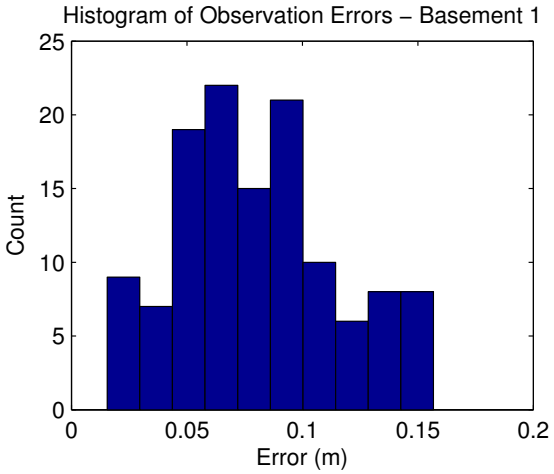


(c) Histogram of Control Point Estimation Errors

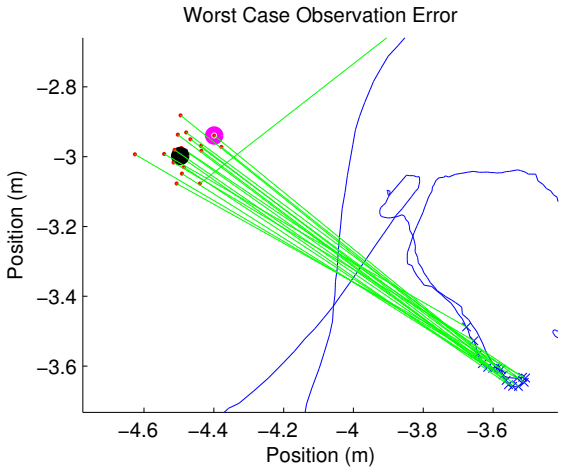
Figure 4.27: Results from the control point test for large-scale dataset 1. (a): The estimated locations of the control points from the third story of the dataset overlain on the CAD drawing. The estimated locations of the control points are shown in red while the estimated trajectory of the system is shown in blue. The green lines denote an observation of a control point. (b): The histogram of observation errors for control points from all floors of large-scale dataset 1. (c): The histogram of control point estimation errors.



(a) Diagram of Control Point Observation Locations



(b) Histogram of Observation Errors



(c) Close Up View Of Worst Case Errors

Figure 4.28: The observation error results for Basement 1 of Large-Scale Dataset 1. (a): The estimated trajectory of the system annotated with the locations of each observation of a control point. The blue line indicates the estimated trajectory while the green lines and red dots indicate the directions and estimated locations of the control points. The red dots are sized to reflect the maximal error in each grouping of control point estimates. The yellow line indicates the region of the room not visible from the pink box due to the range limit of the 2D scanners. (b): The histogram of observation errors for all observations from Basement 1. (c): A close up view of the worst case errors from the pink box shown in (a). The black dot indicates the ground truth position of the control point while the pink dot indicates the observation with the maximal error.

Floor	Mean (cm)	Max (cm)	Standard Deviation (cm)
Basement 1	8.1	15.7	3.4
Floor 1	11.3	33.8	5.4
Floor 2	9.0	20.6	3.9
Floor 3	10.1	24.9	4.3

Table 4.5: Observation Error Results for Large-Scale Dataset 1.

Floor	Mean (cm)	Max (cm)	Standard Deviation (cm)
Basement 1	4.4	12.5	2.7
Floor 1	5.2	11.3	2.7
Floor 2	6.2	14.8	3.2
Floor 3	5.8	18.7	3.8

Table 4.6: Control Point Estimation Error Results for Large-Scale Dataset 1.

of errors centered around the maximal value of 12-15 cm. Figure 4.28(c) shows a zoomed in view of the worst case errors. The spread of estimates from this control point is very large. Of the 7 worst case estimates from the histogram of Fig. 4.28(b), 6 of them are from estimating this control point.

This result can be understood by examining the trajectory of the system and the geometry of the environment. The main room of Basement 1 is an open area around 30 meters in length. When the operator transitions from the large main room to the left-hand stairwell, the laser scanner is unable to scan the right side, shown in yellow in Figure 4.28, of the room large due to the 30 meter range limit of the Hokuyo UTM30-LX. When this happens, the laser scanners can only see planes whose normals span $\mathbb{R}^2$ and thus one dimension of the accelerometer bias is unobservable to the laser data fusion algorithm. Furthermore, since the null of the laser scanner is pointed forward, the system is unable to see geometry in front of itself as it transitions from known to unknown space. During these degenerate transitional regions, the system must rely on the MSCKF for feedback. Since the drift rate of the MSCKF algorithm is higher than that of the laser data fusion, the position error grows in those sections at a much higher rate compared to the rest of the trajectory.

In addition to the observation error metric, we also computed the best estimate of the control point and compare it against the surveyed ground truth. The key difference in this metric over the metric of the previous paragraphs is that we first pre-average all observations of a control point into a single estimate before performing least-squares alignment to align the backpack's coordinate system against the ground truth coordinate system. In doing so, we average out the noise in the path and characterize the error in the control point estimate instead of the error in the reconstructed trajectory. The mean error using this metric was computed to be 5.4 cm over 83 control point estimates. The maximum error was found to be 18.7 cm and the standard deviation of all errors was 3.2 cm. The corresponding histogram for this error metric is shown in Fig. 4.27(c). Table 4.6 summarizes the results separated by building floor. For all floors of the building, the error in control point estimation is

substantially lower than that shown in Table 4.5. This result is intuitive as the first metric penalizes each observation while the second metric allows errors to cancel. Since the 3D estimation algorithm utilizes a least-squares EKF based approach, it is not surprising that averaging the estimates produces a substantially lower error.

Although, it was not possible to perform a direct comparison against the environment shown in Fig. 3.20, a comparison between the 2.5D and 3D localization methods was made using the Basement 1 of Large-Scale Dataset 1. Basement 1 most closely approximates the environment of Fig. 3.20 as they both are planar environments with few windows. Since the results reported in Fig. 3.20 only utilize the control point estimation error metric, we compare the results from Table 4.6 against those reported in Figure 3.20. In Figure 3.20, the mean and maximum errors resulting from the 2.5D localization algorithms was reported as 10.66 cm and 27.73 cm respectively. For the Basement 1 section of Large-Scale Dataset 1, we see that the control point estimation error has a mean error of 4.4 cm and a maximum error of 15.7 cm. The reduction in control point estimation error can be attributed to two main factors. First, the EKF based 3D approach performs tight sensor fusion of the IMU data instead of combining them in an ad hoc method. Any errors in the pitch and roll values of the IMU are corrected and updated instead of being trusted blindly. Secondly, the EKF optimizes the calibration parameters online instead of using values derived from a CAD model. Since the control point estimation process requires combining data sources from multiple sensors and correctly orienting them in the global frame of reference, any errors in the CAD derived calibration parameters or estimated roll and roll directly leads to increased error in control point estimation.

In order to further compare the end-to-end 3D mapping and online calibration algorithms against the 2.5D methods presented in Chapter 3, Basement 1 of Large-Scale Dataset 1 was also processed using our 2.5D methods. During the graph optimization step of the 2.5D localization algorithm, we manually added loop closures 10 at a time and computed both metrics to characterize the performance of the system. This process was repeated until 100 manual loop closures were defined. Figure 4.29 depicts the performance of the 2.5D methods of Chapter 3 in estimating the control point locations for both observation and control point error metrics. Notice that after around 40 manual loop closures have been added, additional loop closures have little to no effect on the error metrics. The reasoning behind this phenomena is that loop closures can only make the trajectory self-consistent and cannot correct for errors such as scaling. In this dataset, the noise floor accuracy for both metrics is approximately 30 cm. Since the 2.5D method tends to underestimate the distance traveled, the primary source of global error is due to a scaling of the resulting geometry. Since pre-averaging the control point observations does not remove bias as from the estimates, we do not see any substantial difference between the two metrics after around 60 manual loop closures. In contrast, the 3D localization algorithms of this chapter obtain a mean error of 8.1 cm for the observation error metric and a mean error of 4.4 for the control point estimation metric.

In order to illustrate another difference between the 2.5D and 3D localization methods, the thickness of the walls in the reconstructed point cloud were examined. Figure 4.30 shows an example comparison from the basement of large-scale dataset 1. Figure 4.30(a) shows an a small subsection of the point cloud. Figures 4.30(b) and (c) show the reconstructed wall thicknesses for the 2.5D and 3D methods respectively. Notice that the wall thickness

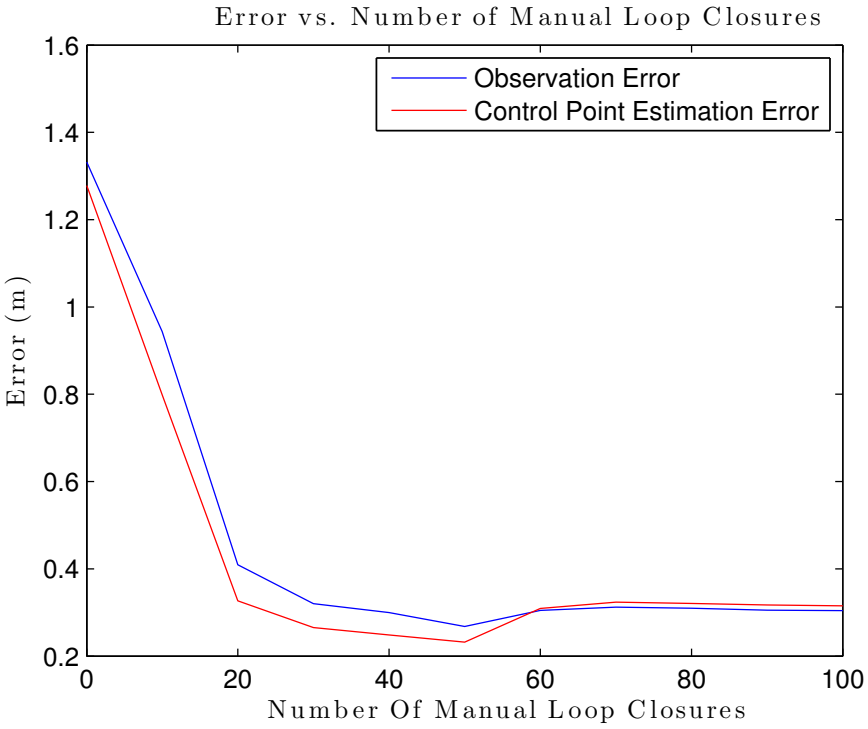


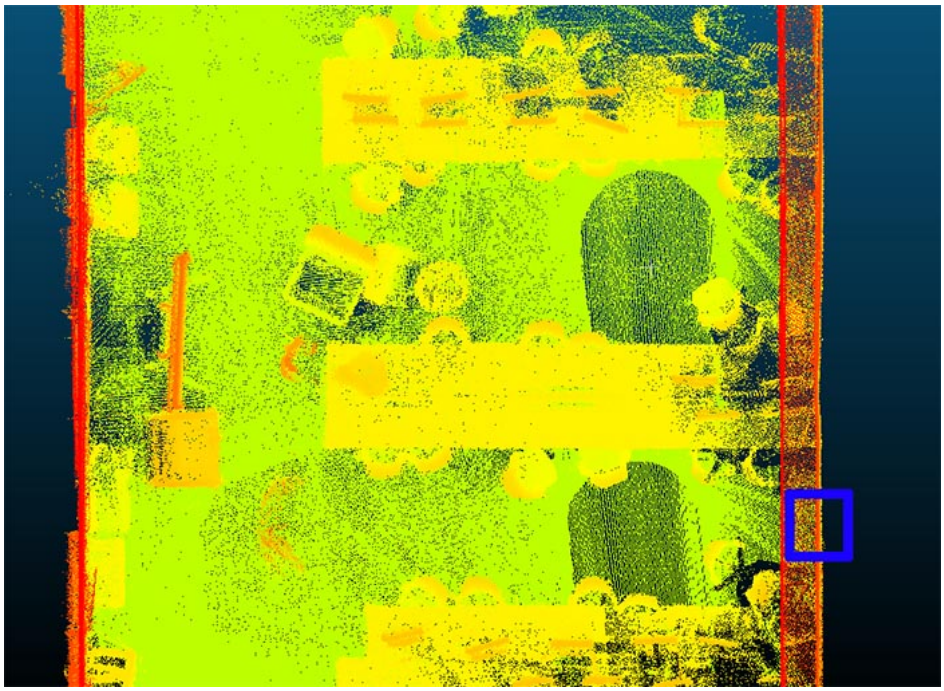
Figure 4.29: The effect of manual loop closures on both observation and control point estimation errors. After a certain number of manual loop closures have been added, adding more provides diminishing returns.

has been reduced from approximately 12.5 cm to approximately 7 cm. The example of Fig. 4.30 is representative of the dataset as a whole as the majority of wall thicknesses in the 2.5D reconstruction range between 8-20 cm while the range of wall thicknesses for the 3D methods is 4-12 cm. The explanation for this result is that the laser data fusion portion of the EKF explicitly attempts to align the laser data readings into planar objects, while the 2.5D method does not explicitly enforce this condition.

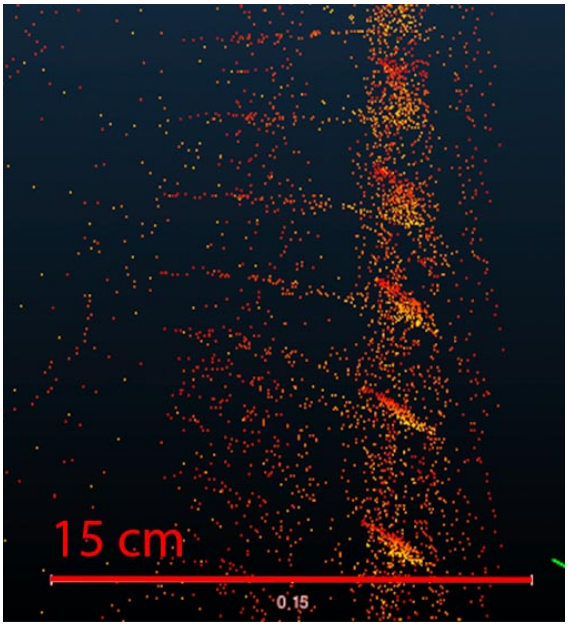
4.9.5 Scanner Configuration Results

In order for the laser data fusion algorithm to successfully provide feedback to all states of the EKF, the union of the normal vectors of all observed planes must span $\mathbb{R}^3$. When the system is unable to observe normal vectors that span $\mathbb{R}^3$, one or more dimensions of the IMU biases are unobservable to the laser data fusion algorithm and the EKF must rely on the MSCKF algorithm which has a higher drift rate than the Lasers+IMU for feedback. The position estimate drifts at a faster rate during periods of geometric degeneracy degrading the accuracy of the reconstructed trajectory.

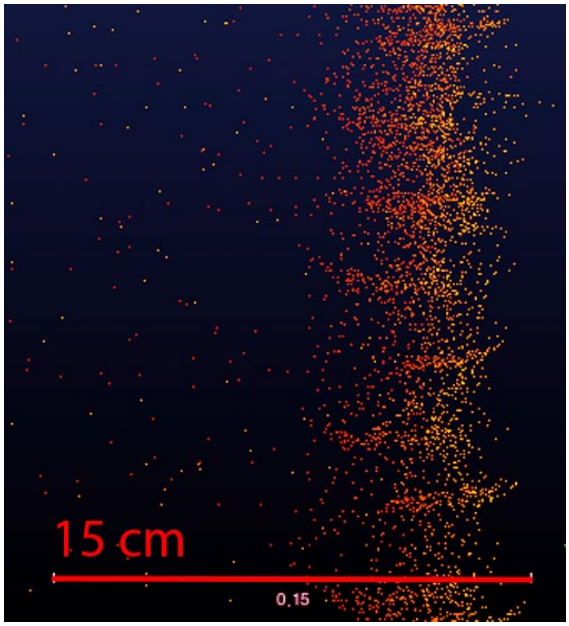
Motivated by the results of Section 4.9.4, we experimented with the orientation of the system's horizontally mounted laser in order to understand the relationship between the laser's orientation and the algorithm's performance. As shown in Figure 2.5(b), the neck of the operator occupies the 90° null in the horizontally mounted laser's field of view. This



(a) Point Cloud Section From Large-Scale Dataset 1

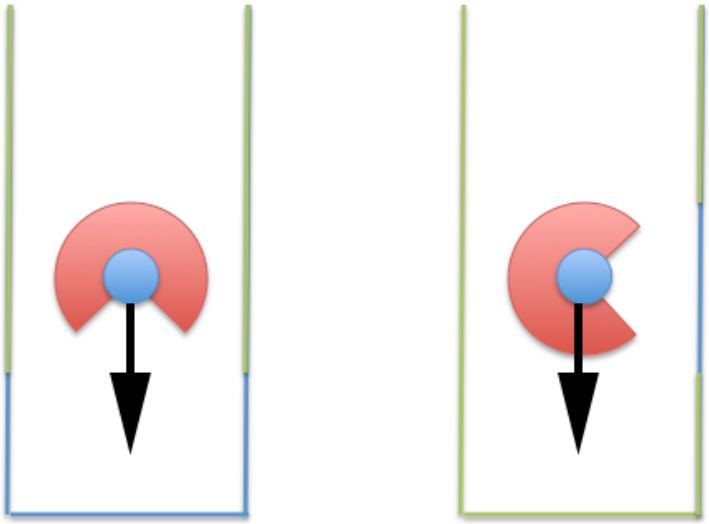


(b) Wall Thickness From 2.5D Methods



(c) Wall Thickness From 3D Methods

Figure 4.30: Wall thickness comparison between 2.5D and 3D localization methods from a large-scale dataset 1. (a): A close up view of one section of the reconstructed point cloud. (b): The wall thickness is approximately 12.5 cm for the 2.5D method. (c): The wall thickness is approximately 7 cm for the 3D method.



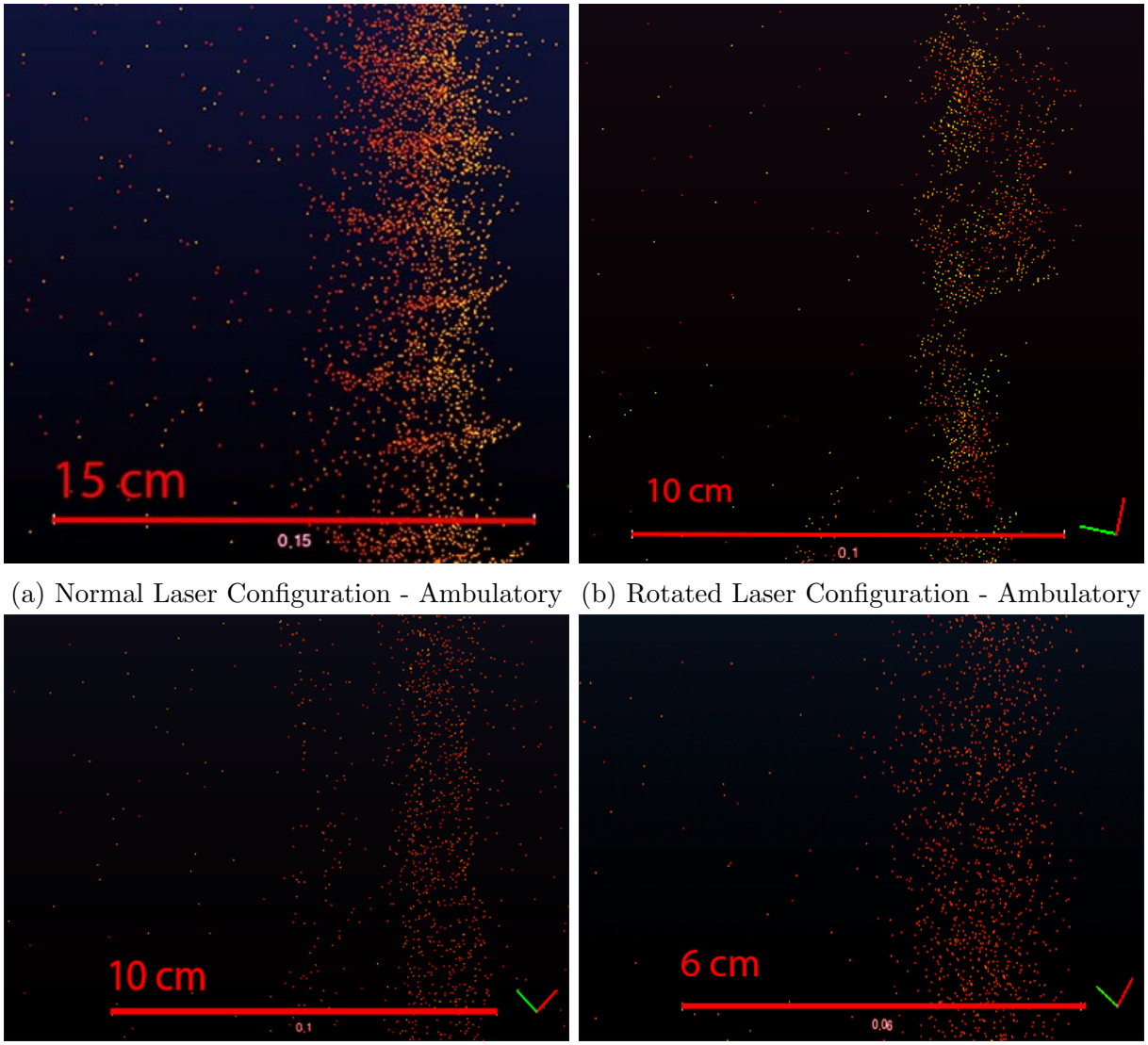
(a) Normal Laser Configuration (b) Rotated Laser Configuration

Figure 4.31: An illustration depicting how rotating the null of the laser scanner can avoid some situations with geometric degeneracy. The field of view of the laser is shown in red while the visible portions of the geometry are shown in green. The direction of travel is indicated using a black arrow. (a) Since the system is unable to see geometry directly in front of itself, as the operator walks down a long hallway, the geometry will become degenerate. (b) By rotating the laser 90°, the laser can see in front and behind itself thus eliminating the degeneracy.

means that in the standard configuration the system is unable to scan objects directly in front of itself. Some geometric configurations, such as large rooms and long hallways, the chosen orientation of the horizontally mounted laser scanner can cause degeneracies during the laser data fusion algorithm. By rotating the scanner 90°, the laser scanner can see 30 meters in front as well as 30 meters behind. This effectively doubles the range of the laser scanner and therefor the algorithm can obtain a more accurate result by avoiding the majority of degenerate situations.

Figure 4.31 shows an illustration of rotating the laser scanner to avoid degenerate configurations. Figure 4.31(a) depicts the system as the operator approaches the end of a long hallway. Shown in green, the visible portions of the geometry do not span $\mathbb{R}^3$ and thus the system will begin to drift at an accelerated rate. Figure 4.31(b) shows that by rotating the null of the laser by 90°, the laser can see both in front and behind which minimizes the chances of degeneracy.

To demonstrate the effect of rotating the laser scanner on the accuracy of the reconstructed trajectory, we rescanned Basement 1 of Large-Scale Dataset 1 using both the normal and rotated laser configurations shown in Fig. 4.31. Since we were unable to physically rotate the scanner on the backpack system, data was collected for the rotated laser config-



(a) Normal Laser Configuration - Ambulatory (b) Rotated Laser Configuration - Ambulatory

(c) Normal Laser Configuration - Cart (d) Rotated Laser Configuration - Cart

Figure 4.32: Wall thicknesses comparisons. (a): The wall thickness is an estimated 8 cm for the normal configuration when carried by a human operator. (b): The wall thickness is an estimated 5 cm for the rotated configuration when the system is carried by a human operator. (c): The wall thickness is an estimated 5 cm for the normal laser configuration when the system mounted on a cart. (d): The wall thickness is estimated as 3.5 cm for the rotated laser configuration when the system is mounted on a cart.

uration by walking sideways to mimic the rotated laser configuration. In addition, we also mounted the system on a cart and collected data using both laser configurations in order to explore the interaction between the operator's gate and the laser scanner configuration. We examined the wall thicknesses and amount of double surfacing in the reconstructed point cloud to evaluate the performance of the system.

Figure 4.32 shows the resulting wall thicknesses from the subsection of the point cloud depicted in Fig. 4.30(a) for the normal and rotated laser configuration when the system is

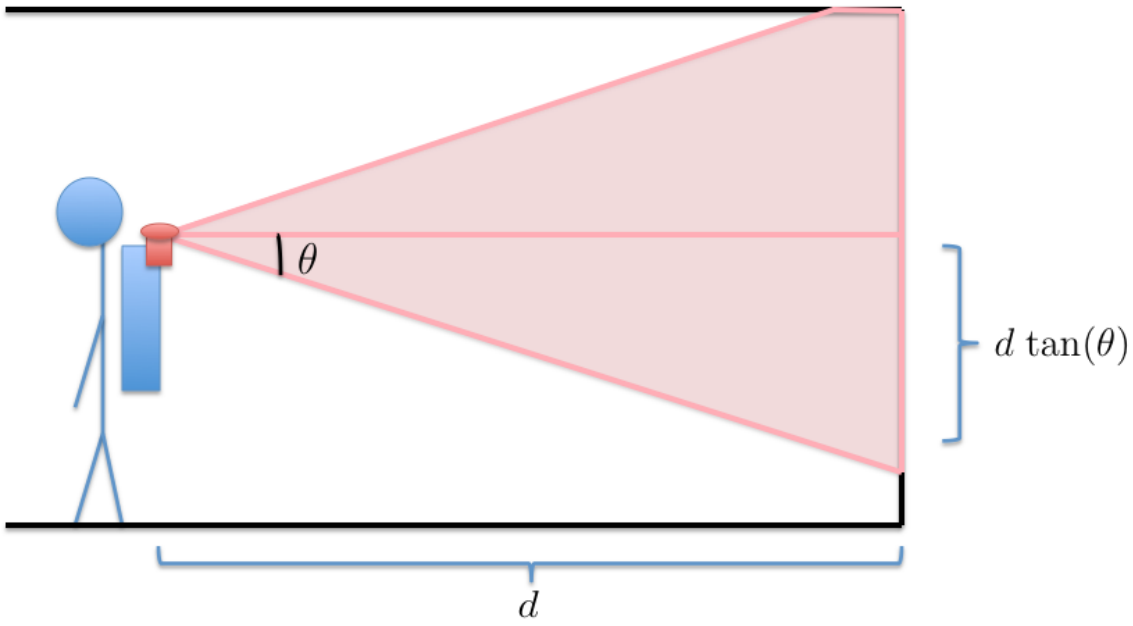
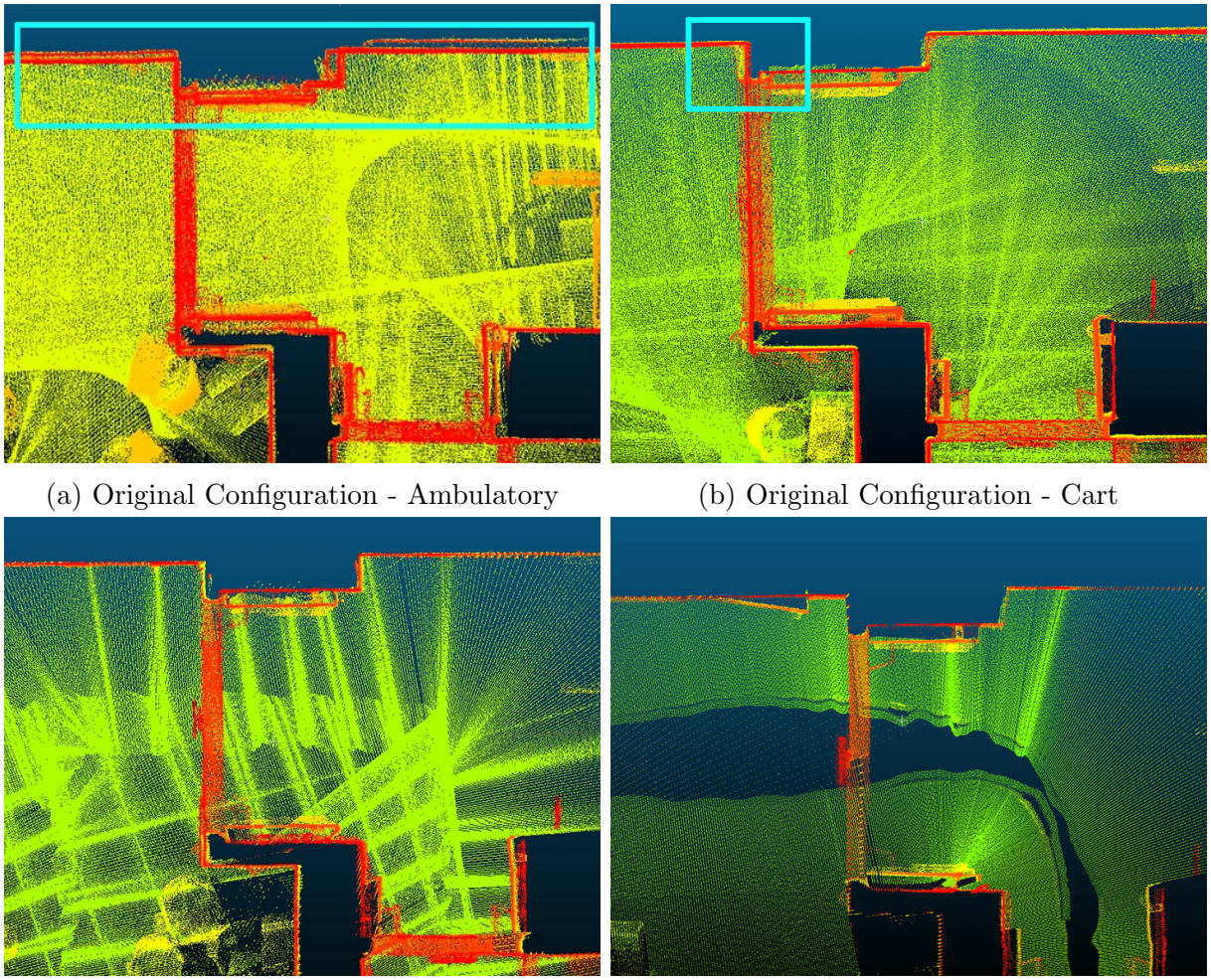


Figure 4.33: An illustration of pitch and roll motion causing the horizontally mounted laser scanner to scan the ceiling or floor. As the scanner undergoes pitch of magnitude θ, the vertical displacement of the scan plane travels a distance of $d\tan(\theta)$. For a sufficiently large room or short operator, the scanner will begin to scan the ceiling or floor.

both carried by a human operator and mounted to a pushcart. In the normal configuration the wall thicknesses are estimated to be around 6-15 cm when carried by a human operator and 5-10 cm when mounted in the rotated configuration. By mounting the system to a cart and orienting the laser scanner in the original configuration, the measured wall thicknesses are reduced to 5-7 cm. After rotating the laser scanner, the range of wall thickness was further reduced to an estimated 3.5-5.5 cm. It is important to note that all three results shown in Fig. 4.32 are better than the 4-12 cm reported in Fig. 4.30. This indicates that both the smooth motion of the cart and the rotated laser configuration contribute to the decreased wall thicknesses present in the reconstructed point cloud. In particular, we can conclude that the smooth motion profile of the cart has a more dramatic effect on reconstructed wall thickness than the orientation of the scanner.

Mounting the system on a cart provides a number of advantages that increase the accuracy of the system. First, by mounting the system on a cart, the system does not undergo the same magnitude of pitch and roll as when carried by a human. By stabilizing the laser scanner, the horizontally mounted laser scanner always scans in the global xy plane instead of periodically scanning the ceiling or floor. Figure 4.33 shows an illustration of this process. As the laser scanner undergoes a pitch of magnitude θ, the horizontal scan plane travels a vertical distance $d\tan(\theta)$. For sufficiently large room or short/tall operator, the laser will scan the ceiling/floor instead of the vertical geometry. Secondly, when the system is carried by a human operator, the accelerometer reads a large spike every time a step is taken. This



(a) Original Configuration - Ambulatory

(b) Original Configuration - Cart

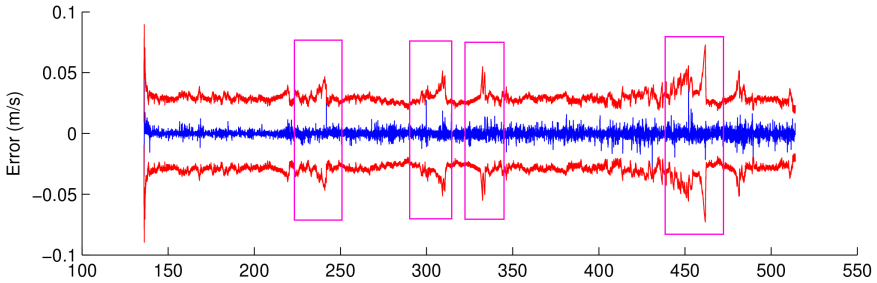
(c) Rotated Laser Configuration - Ambulatory

(d) Rotated Laser Configuration - Cart

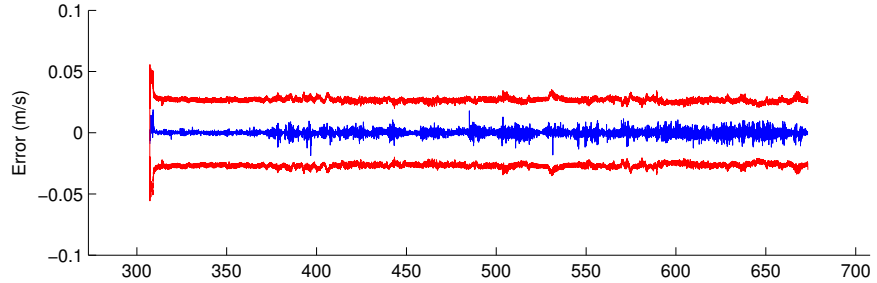
Figure 4.34: Double surfacing results for both the original and rotated laser configurations. Regions of double surfacing are indicated with cyan squares. (a): High double surfacing is present around the selected door frame in the ambulatory version of the original configuration. (b): By mounting the system on a cart, the double surfacing was reduced. (c): Using the rotated laser configuration eliminates all instances of double surfacing in the reconstructed point cloud. (d): Using the rotated laser configuration and mounting the system on a cart eliminates the double surfacing and provides the most accurate results.

process introduces extra noise into the acceleration readings which drives the EKF further away from its linearization point and degrades the accuracy of the reconstructed trajectory. Both of these factors help to explain the increase in accuracy even when the laser scanner configuration remains unchanged.

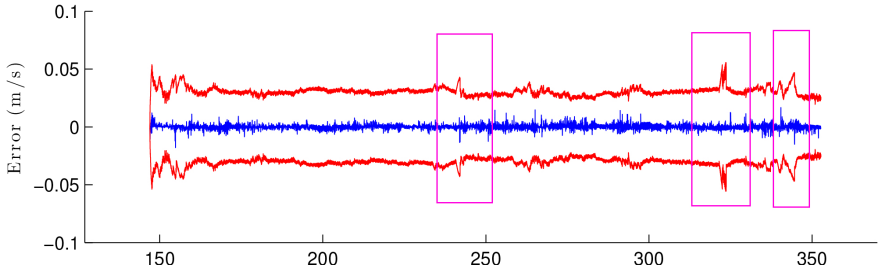
In addition, we also visually examined the point cloud for any regions of double surfacing. Whenever the system experiences degenerate geometry, the drift rate of the algorithm increases which leads to misalignments in the reconstructed point cloud. Figure 4.34 shows a subsection of the point cloud reconstructed using both laser configurations. Figure 4.34(a) corresponds to the case of normal laser configuration when carried by a human operator



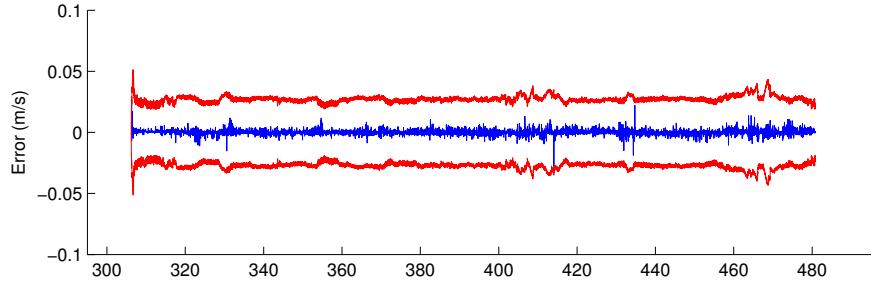
(a) Original Configuration - Ambulatory



(b) Rotated Configuration - Ambulatory



(c) Original Configuration - Cart



(d) Rotated Configuration - Cart

Figure 4.35: Comparison of the variance of the forward direction of the system's velocity for the different laser configurations. The blue lines indicate the estimates of the error state while the red lines denote the 3σ bounds on the error estimate. Pink boxes indicate degenerate locations. (a): The variance plot for the original configuration when human mounted. Notice the regions of the dataset where degeneracies caused rapid increase in the velocity's variance. (b): The variance plot using the rotated laser configuration when human mounted. (c): The variance plot after mounting the system on a cart using the normal configuration. (d): The variance plots for the rotated laser configuration and mounting the system on a cart.

and shows significant double surfacing in the top region of the indicated subsection of the point cloud. Figure 4.34(b) corresponds to the same configuration but after mounting the system on a pushcart. The amount of double surfacing has been reduced, but minor double surfacing still exists. Figure 4.34(c) corresponds to the case of a rotated laser carried by a human operator and shows that by rotating the laser scanner, all double surfacing has been eliminated. Furthermore, Figure 4.34(d) shows that rotating the laser scanner and mounting the system on a pushcart produces the most visually appealing results.

The above results can be explained by examining the variance of the system's velocity along the direction of motion. Whenever the system experiences geometric degeneracy, the variance in the degenerate dimension increases rapidly. Figure 4.35(a) shows the velocity variance for the original laser configuration along the dominant direction of the building. Pink boxes indicate the four regions of the dataset where the laser undergoes geometric degeneracy. These sections roughly correspond to regions of the point cloud that contain significant double surfacing. Figure 4.35(b) shows the variance plot using the rotated configuration when the system is human mounted. Notice that by using the rotated laser configuration, there are no degenerate regions of the dataset. Figure 4.35(c) shows the variance plot after mounting the system on a cart. Notice that the stabilization of the pitch and roll and helps to mitigate some degenerate regions because the range of scanner is not artificially limited by ceiling or floor data as indicated in Figure 4.33. When the range of the scanner is artificially limited in this manner, some otherwise observable planes may no longer be visible and the support of scanned planes can drop below $\mathbb{R}^3$. Ceiling and floor planes are often large with respect to wall planes and thus are less important for spanning $\mathbb{R}^3$. Ceiling and floor planes are generally visible to at least one laser scanner, namely the Floor and Up Geometry scanners shown in Fig. 2.5(c), at all times while a wall plane may only be visible for a limited amount of time. Although stabilization is beneficial, it does not mitigate all degenerate regions. Finally, Fig. 4.35(d) shows that using the rotated laser configuration also mitigates all degenerate locations when the system is mounted to a cart.

4.9.6 Timing Results

Although the data collection system stores all data for offline processing, it is important to highlight the run-time performance of the algorithms. In this section we detail the basic processing time needed to run the end-to-end system on the large-scale datasets presented in Section 4.9.3. Timing experiments were conducted using an unoptimized, single threaded, C++ implementation of the proposed algorithms on a 2.4GHz Intel i7 processor. The timing results are presented in Table 4.7.

As seen in Table 4.7, the slowest portions of the end-to-end system are the laser data fusion and 3D ICP stages of the algorithm. Determining loop closure constraints requires performing 3D ICP between dense submaps containing hundreds of thousands of points. Since ICP algorithms requires knowledge of a point's nearest neighbor to evaluate its objective function, nearest neighbor lookups dominate the majority of ICP's processing time. While preprocessing the submaps using techniques such as k-d trees [136] can significantly reduce search time, the dense nature of 3D submaps makes the ICP algorithm a processing bottleneck.

Although data fusion for each laser is only 12.5s, our ambulatory backpack system con-

Subtask	Time (min) Dataset 1	Dataset 2	Dataset 3	Mean Per Min. of Collection Time (sec)
Walking Time	20.0	23.0	27.4	-
IMU Integration	0.17	0.24	0.31	0.6
Per Camera Data Fusion	7.70	8.87	10.48	23.1
Per Laser Data Fusion	4.17	5.17	5.31	12.5
3D ICP	18.77	21.37	29.72	59.5
Graph Optimization	2.31	2.67	3.22	7.0
Image Preprocessing	23.56	30.08	32.25	73.2
Submap Creation	5.49	8.06	11.16	21.07
Processing Time (4 Lasers and 1 Camera)	74.68	91.97	108.4	234.4

Table 4.7: Timing results for the end-to-end 3D localization algorithms. Note that there are 4 laser scanners and 1 camera present in our backpack data collection system.

tains 4 laser scanners and thus the total time required for laser data fusion is almost 60s per minute of data collection. The laser data fusion algorithm has two compute intensive steps that require the majority of its processing time. Numerical optimization of the line parameters and their covariances is required for every line segment detected. While this process is inexpensive for a single line segment, the system must process hundreds of line segments per second. If additional speed is needed, this process can be distributed across many CPUs or a GPU. Additionally, since the laser data fusion algorithm operates on a dense $N \times N$ covariance matrix, computing the Kalman gain and covariance update via Eqs. 4.84 and 4.85 can become computationally expensive if the number of planes tracked by the filter grows too large.

Also included in Table 4.7 are timing results for both the image preprocessing and submap creation tasks. Image preprocessing is required mainly for two reasons. First, instead of capturing standard RGB images, the camera sensors mounted on our backpack system record raw Bayer coded images. Images must first be converted into a single 8-bit intensity image before further processing can be applied. Secondly, the camera data fusion algorithms assume that the images are subjected to perspective distortion instead of spherical fisheye distortion and thus must be dewarped before being passed to the camera data fusion algorithm. Submap creation involves assembling small point clouds and is mainly an I/O bound process as a separate point cloud file must be created and stored on disk for each submap. While these steps are not strictly part of the proposed algorithm, they are required and thus effect the overall run time of the system. Since these tasks are primarily dominated by I/O operations, their runtime is ultimately limited by the speed of the hard drive that stores the data.

As shown in Table 4.7, the total time required per minute of data collection is around 235 seconds. In order to reduce the required processing time and apply the presented end-to-end system under real-time constraints, some work would be required to speed up the ICP and laser data fusion sections. Fortunately, these sections are able to be parallelized across

Subtask	Complexity
IMU Integration	$\mathcal{O}(n)$ with walking time.
Camera Data Fusion	$\mathcal{O}(n)$ with walking time. $\mathcal{O}(n^2)$ with size of sliding window.
Laser Data Fusion	$\mathcal{O}(n)$ with walking time. $\mathcal{O}(n^3)$ with the number of plane states.
3D ICP	$\mathcal{O}(n)$ worst case with walking time.
Graph Optimization	$\mathcal{O}(n^3)$ worst case with walking time. $\mathcal{O}(n^{1.5})$ expected with walking time.
Image Preprocessing	$\mathcal{O}(n)$ with walking time.
Submap Creation	$\mathcal{O}(n)$ with walking time.

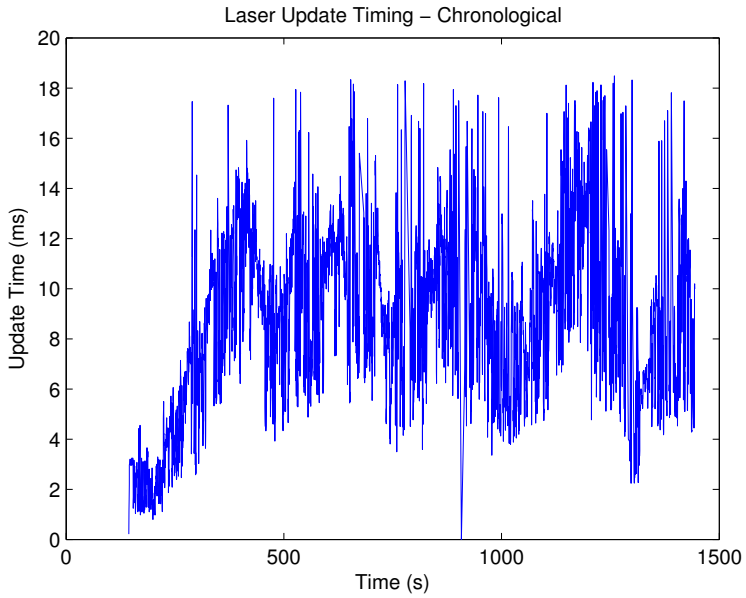
Table 4.8: Summary of the complexity for different stages of the 3D localization algorithm.

many threads. Since each line feature measurement is independent of all others, the work is trivially parallelized across many threads. Additionally, in the 3D ICP algorithm nearest neighbor search consumes the most time and can be directly parallelized as each points nearest neighbor can be calculated independently. Furthermore, in a strict real-time system, many I/O operators required for the image preprocessing and submap creation would no longer be necessary.

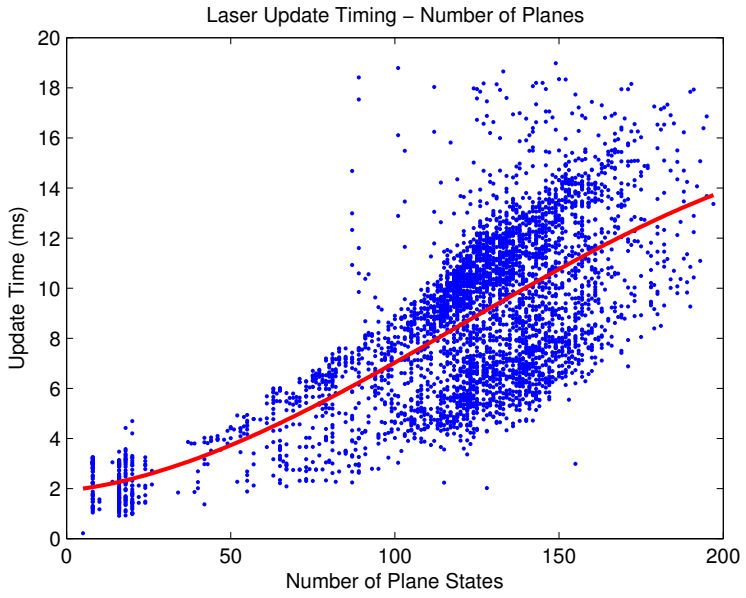
In the event of a missed or erroneously detected loop closure, manual loop closure can take a significant amount of human intervention. Although rare, a human must first diagnose and locate the region which either requires a manually added or removed loop closure. This process is time and labor intensive and can easily dominate the processing time if an extensive number are needed. A trained technician may spend 1-2 minutes per manual loop closure required. Once the manual loop closures have been specified, the saved state of the incremental graph optimization and manually defined loop closure are used to restart the graph optimization algorithm from the location of the loop closure as described in Section 4.9.3. After incorporating the manual loop closure information into the trajectory, the point cloud is regenerated and visually inspected. This process is then repeated until the operator is satisfied with reconstructed trajectory.

The scalability of the 3D localization algorithms are affected by a number of different factors. Table 4.8 summarizes the scalability for the different steps of the algorithm. Many of the stages of the algorithm, including IMU integration, image preprocessing, and submap creation, are trivially shown to be linear in the operator's walking time. Other stages, such as camera data fusion and graph optimization, have had their complexities previously explored in the literature. For more details about the run-time complexities of these stages, see [115] and [33] receptively. Since by design we only a submap to contribute a single loop closure constraint, the number of loop closures is upper bounded by the number of submaps. In doing so, the process for detecting of loop closure locations and transform estimation via 3D ICP detailed in Section 4.8.2 is at worst linear in the operator's walking time.

The laser data fusion algorithm avoids unbounded computational complexity by marginal-



(a) Laser Update Timing — Chronological



(b) Laser Update Timing — Environment Complexity

Figure 4.36: Timing results for a laser based filter update as shown in Eq. 4.85 on Large-Scale Dataset 2. Notice that the time per update is bounded with respect to walking time and super-linear with respect to the number of plane states. (a) The time required per update shown chronologically. (b) The time per update shown with respect to the number of planar filter states. The red line shown the best fit cubic polynomial to the data.

izing out planes that have not been used after a fixed period of time. In this way, it implements a sliding window over a small subsection of the environment's geometry. This gives the EKF update portion of the laser data fusion algorithm, Eq. 4.85, a linear complexity with respect to the operator's walking time. The complexity of the EKF update portion of the laser data fusion algorithm is primarily dominated by updating the filter's dense covariance matrix which scales as $\mathcal{O}(n^3)$ with the number of planes tracked in the filter's planar environment map.

Figure 4.36 demonstrates the linear complexity with respect to walking time and environmental complexity on Large-Scale Dataset 2. Figure 4.36(a) shows the time required for each laser based filter update in chronological order. After a brief period of "warming up," the time per update plateaus and remains roughly bounded. Figure 4.36(b) shows the per update time with respect to the number of plane states tracked by the filter. Based on the data, we observe an obvious super-linear trend in the data. A best fit cubic polynomial is overlaid on the data in order to show that the observed timing data fits reasonably well to the $\mathcal{O}(n^3)$ complexity of the EKF update.

4.10 Discussions

This section describes the limitations and best practices for using the 3D localization algorithms presented in this Chapter. In addition comparison are made against the strengths and weaknesses of the 2.5D methods presented in Chapter 3.

4.10.1 Limitations and Best Practices

Although the end-to-end 3D mapping system presented in this chapter was shown to accurately map a wide variety of environments, it does have a number of limitations that prevent it from being applicable to all environment types. In this section, we will discuss the limitations for each of the data fusion methodologies and provide a list of best practices to obtain the best results when utilizing the presented algorithms.

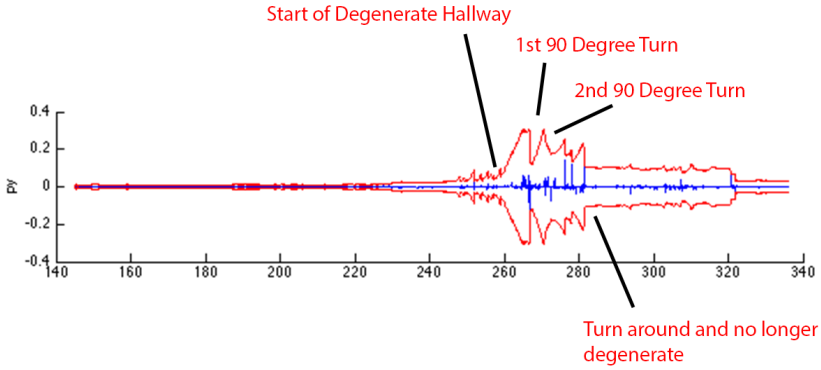
The largest limitation of the laser data fusion algorithm is that the normal vectors of the observed planes must span $\mathbb{R}^3$ for full IMU state observability. Since the individual laser scanners only scan in a single 2D plane, it is unlikely that any single laser scanner will observe geometry that meets this condition. However, since the laser scanners all operate nearly simultaneously, it is sufficient for the plane normals from the union of *all* laser scanners over a sufficiently small temporal window to span $\mathbb{R}^3$. In the event that geometry is degenerate and the plane normals fail to span $\mathbb{R}^3$, one or more dimensions of the IMU's accelerometer bias $\mathbf{b}_a$ will not be observable by the EKF. Without proper feedback, the variance of the system's velocity and position estimates will increase rapidly and the trajectory will start to accumulate drift.

The most common scenario for this to occur is when the operator is traveling down a hallway that is longer than the range of the laser scanner. Figure 4.37 shows a diagram of horizontally mounted laser scanner in a long hallway environment in a number of configurations. Figure 4.37(a) depicts the laser scanner observing a well-conditioned set of planar features. Due to the blind spot in the laser scanner's field of view, rotating 180° from the

(a) Non-degenerate configuration.

(b) Degenerate configuration.

(c) Performing a turn to mitigate drift.



(d) Positional variance from a long hallway dataset.

Figure 4.37: A diagram of degenerate geometry for laser aided inertial navigation. The blue circle represents the center of the laser scanner and the red arc indicate the laser scanner's field of view. The orange lines indicate portion of the geometry visible to the laser scanner. (a): A non-degenerate configuration for the laser data fusion algorithm. (b): Due to the laser scanner's blind spot, rotating 180° from the orientation shown in (a) causes the geometry to become degenerate. (c): Drift can be mitigated by performing a turn midway down the hallway. (d): The positional variance plot from a long hallway environment.

position shown in Fig. 4.37(a) causes the geometric configuration to become degenerate for laser aided inertial navigation as shown in Fig. 4.37(b). As a best practice, the operator should avoid degenerate geometric configurations at all cost.

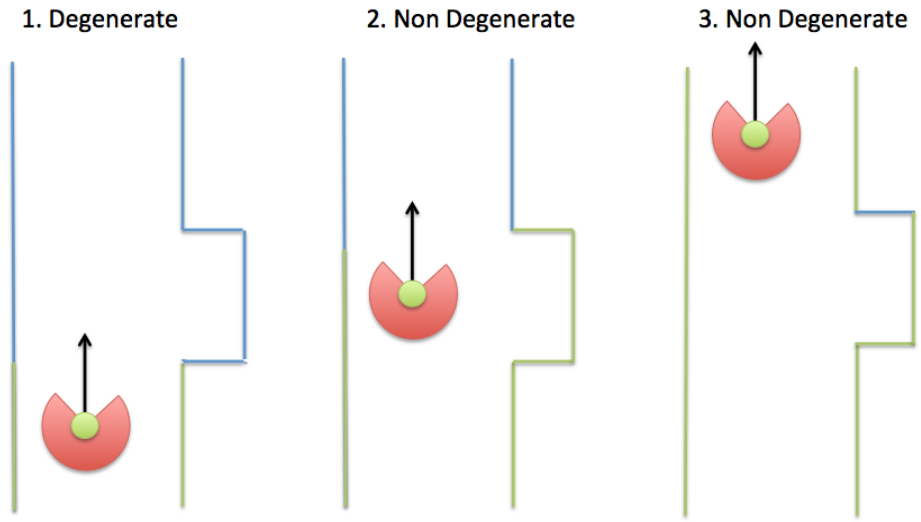
If it is impossible to avoid degenerate geometric configurations, then there are two strategies to mitigate drift. First, by ZUPTing the system, the EKF would be able to reduce the variance in the dimensions of the accelerometer bias that are perpendicular to gravity. Secondly, by turning 90° midway down the hallway, the operator can temporarily calibrate the accelerometer bias and reduce the rate at which drift accumulates. This process is highlighted in Figs. 4.37(c) and (d). Figure 4.37(c) depicts how turning midway down the hallway can temporarily put the laser scanner back into a well-conditioned geometric configuration.

Figure 4.37(d) depicts the positional variance along the hallway direction from a real-world dataset. When the operator begins to travel down the long hallway, the geometry resembles Fig. 4.37(b) and thus the positional variance begins to grow rapidly. By executing two 90° turns as depicted in Fig. 4.37(c), the operator is able to temporarily reduce the drift. Once the operator reaches the end of the hallway and turns around, the geometry resembles Fig. 4.37(a) and thus the positional variance is bounded. The relevant portions of the dataset are labeled in Fig. 4.37(d).

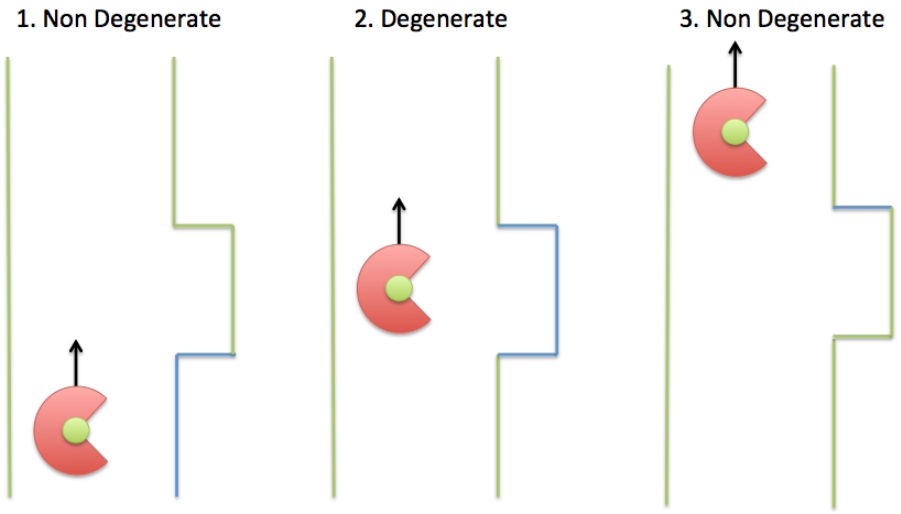
Although the frequency of geometric degeneracies can be reduced by using the rotated laser configuration as shown in Section 4.9.5, some geometric configurations require a full 360° field of view. Figure 4.38 shows one configuration of geometry that causes degeneracies regardless of the orientation of the null with respect to the direction of travel. As the operator walks upward through the hallway, the scanner will undergo a short period of degeneracy regardless of the laser configuration. In order to minimize the frequency of degenerate geometric configurations, the best laser configuration would contain no gaps in the horizontal field of view of the system.

Another common reason for poor performance of the EKF estimator is the presence of moving objects in the environment. Although both the camera and laser data fusion algorithms use Chi-squared outlier testing to remove bad observations, a scene with too many moving objects can corrupt the filter estimates. Large planar objects, such as doors, also present a problem for the data fusion algorithm. Doors are large enough to be modeled in the EKF's plane map, but often move during the course of data acquisition. Extreme care must be taken when walking through doors. It is generally best practice to avoid pointing the laser scanners at a door while it is in motion. In order to avoid problems related to moving doors, an operator should ensure that the null of the XY laser scanner is pointed at the door until the door has closed completely. For the XY scanner configuration used in Fig. 2.5, null is pointed forward and thus the operator should open the door normally, walk through and turn around towards the door, and then let it close completely before continuing to explore the environment.

Motion blur also limits the effectiveness of the MSCKF algorithm in dimly lit environments. When there is poor ambient lighting in the environment, the camera must increase its exposure time to avoid images that are too dark. As the exposure time increases, the effect of motion blur also increases. Motion blur causes smearing in the image which makes feature detection and tracking more noisy. Noisy feature estimates cause the camera data fusion algorithm to track fewer features and thus provide less feedback to the EKF. To that end, it is best practice to limit the rotation velocity of the operator and ensure that areas



(a) Original Laser Configuration



(b) Rotated Laser Configuration

Figure 4.38: An illustration of a geometric configuration that results in geometric degeneracies regardless of the laser configuration. The scan plane of the sensor is shown in red while the visible portions of the environment are shown in green. As the operator walks upward through the hallway, both configurations will pass through a region of degenerate geometry. (a): Original laser configuration. (b): Rotated laser configuration.

are well lit before scanning.

4.10.2 Comparison Against 2.5D Methods

Although the 3D localization algorithms presented in this chapter provide a number of strengths with respect to the 2.5D methods of Chapter 3, it is important to highlight the pros and cons of using the two methodologies. Since the 2.5D algorithm splits the localization task into a 2D localization problem followed by a height estimation problem, sensor data is combined in an ad hoc fashion. Since data is not combined in a principled fashion, the 2.5D algorithms do not easily scale to an arbitrarily large number of 2D laser scanners. Furthermore, since the gravity vector is taken from the IMU directly with no filtering, any errors attitude errors will degrade the accuracy of the reconstructed trajectory.

In contrast, the 3D localization algorithms directly fuse data from all available laser scanners, a camera, and an IMU into a single optimized trajectory. Since each sensor's contributions are formulated using a separate EKF update, it is trivial to extend the algorithm to an arbitrarily large number of laser scanners. Additionally, because the IMU data is processed by a tightly coupled EKF, data from other sensors can be used to correct errors in the data reported by the IMU.

The 3D localization algorithms also have the advantage that they simultaneously calibrate all relevant calibration parameters while building the optimized trajectory. Each sensor has its own independent coordinate frame and clock and thus accurate calibration between sensors is crucial to obtain accurate localization. Since the 2.5D methods cannot account for any temporal or extrinsic calibration parameters, errors such as misaligned timestamps or manufacturing deviations from the CAD model lead to errors in the reconstructed trajectory.

While the 3D localization algorithms provide a number of advantages over the 2.5D methods, the 2.5D methods have two advantages. First, the laser portions of the 3D algorithms require some large structural planes in the environment for proper operation. Environments, such as those with vertical non-planar objects or columns, are difficult for the laser data fusion portion of the 3D localization algorithms. In contrast, since the 2.5D method utilizes ICP variants to process the laser data, any vertical structure can be utilized. Furthermore, since the 3D algorithms use camera imagery for localization, time intensive image preprocessing must be performed before the 3D algorithms can be applied. Based on the timing results in Table 3.4 and Table 4.7, I/O bound tasks such as image preprocessing and submap creation make the 3D algorithms slower per minute of data acquisition.

Chapter 5

Conclusions and Future Work

In this dissertation we have presented two algorithmic pipelines for robust localization and mapping of an ambulatory backpack system in indoor GPS-denied environments. The methods presented in this thesis are capable of allowing a human operator to map the interior of a building with no a priori knowledge of the building. Furthermore, the presented methods do not require any external infrastructure or database. Through experimental results, we demonstrate a number of novel contributions used for large-scale indoor mobile mapping.

We first discussed the algorithmic pipeline for mapping building interiors in 2.5D environments. By decomposing the SLAM problem into a 2D localization problem followed by a height estimation problem, we were able to scale our 2.5D localization scheme to buildings containing an arbitrary number of discrete floors. In doing so, we demonstrated that a low cost MEMS barometer can be used to automatically segment the data collection into individual floors and detect floor transitions. In addition to the multi-story 2.5D localization algorithm, we also presented an algorithm to compute an validate loop closure constraints for 2D laser based SLAM systems. By combining a robust, outlier resistant objective function and a genetic search, we demonstrated improved performance over state-of-the-art scan matching techniques. Following this, we proposed and validated two metrics based on scan overlap and complexity to vet loop closure constraints using both manually and automatically detected loop closure constraints.

To address the shortcomings of 2.5D approaches, we then presented a modular 3D localization algorithm based on a tightly coupled EKF estimator. The objective of this approach was threefold. First, by formulating the mapping problem as an EKF estimation problem, we were able to naturally fuse data from many sensors mounted on the backpack system into a single, optimal estimate of the trajectory of the system. In contrast the 2.5D method presented in Chapter 3 in which we fused the data the IMU and laser scanners in a ad hoc fashion, our 3D EKF based localization algorithm fused data from all laser scanners, the IMU, and a single backwards facing camera in a principled and optimal fashion. Secondly, using a tightly coupled EKF estimator allowed us to explicitly model the various sensors' calibration parameters during the mapping process. We demonstrated that by including the relevant calibration parameters in the EKF's state vector, we were able to not only overcome timestamping latency but also calibrate out any deviations from the CAD model of the relative positions and orientations of the scanners. We then demonstrated the calibration method reduced parameter variance on both simulated and real-world datasets. Lastly, we

demonstrated the scalability of the presented 3D algorithms using a few large scale datasets.

There are a number of avenues for extending the presented algorithms to both new systems and new sensors. First, low-cost depth cameras provide an interesting avenue of future research. As the cost and size of these sensors decreases, they will become more commonplace in consumer electronics. By adapting the plane mapping algorithm presented in this thesis, a small consumer device equipped with an IMU, camera, and depth sensor could be used to quickly map many locations ill-suited for a backpack system. Additionally, depth sensors provide a high-rate, dense, method of observing the environment and could be used to augment the laser scanners and cameras already present on our ambulatory backpack system. Similar to low-cost depth sensors, high-quality 3D laser scanners could be used to improve the accuracy of the 3D localization algorithms. True 3D scanners, such as the Velodyne VLP-16 [137], can directly measure 3D planar segments instead of 2D line segments and thus would reduce the required number of laser scanners and eliminate the majority degenerate cases in the laser data fusion algorithm. In order to apply true 3D scanners, the algorithm would need to extract planar features from the scan data instead of line features and redefine the residuals from Section 4.6.2 appropriately.

Another possible avenue of future research could be to extend the camera data fusion algorithms to incorporate data from multiple cameras. Although we only utilize data from a single backward facing camera, the backpack system also has two additional cameras used for texture mapping and virtual reality applications. Since the additional cameras have partially overlapping fields of view, sparse depth reconstruction is possible. This signal would provide yet another source of information to fuse in the EKF estimator for an even more accurate trajectory. The major hurdle to incorporating more cameras is that every camera requires around 100 states to track its sliding window of poses. Without proper precautions, the EKF would quickly become computationally intractable.

Finally, manually placed targets in the environment can also be utilized to further increase the accuracy of the system. By placing targets of a predefined size in the environment, loop closure detection and transform estimation could be a more robust process. Computer vision techniques such as the PnP algorithm can be used to find the relative position between two cameras viewing the same calibrated target. These constraints can be naturally fed into the incremental smoothing algorithm to increase the accuracy of the reconstructed trajectory. Furthermore, the targets could also be used in a manner similar to control points by defining them as landmarks similar to the process described in Section 3.4.3. The only modification that would need to be made for this scenario would be that the initial variance of the landmarks would need to be set sufficiently high to model the uncertainty in initializing their locations.

Bibliography

[1] C. Holenstein, R. Zlot, and M. Bosse, "Watertight surface reconstruction of caves from 3d laser data," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 3830–3837.

[2] Trimble.com, "Trimble - indoor mobile mapping solution - indoor mapping," 2015. [Online]. Available: http://www.trimble.com/Indoor-Mobile-Mapping-Solution/Indoor-Mapping.aspx?dtID=overview&

[3] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, "The vSLAM algorithm for robust localization and mapping," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 24–29.

[4] J. Hesch, S. Roumeliotis *et al.*, "An indoor localization aid for the visually impaired," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3545–3551.

[5] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping," *Robotics, IEEE Transactions on*, vol. 28, no. 5, pp. 1104–1119, 2012.

[6] P. Moghadam, M. Bosse, and R. Zlot, "Line-based extrinsic calibration of range and image sensors," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3685–3691.

[7] M. F. Fallon, H. Johannsson, J. Brookshire, S. Teller, and J. J. Leonard, "Sensor fusion for flexible human-portable building-scale mapping," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4405–4412.

[8] N. Corso and A. Zakhor, "Indoor localization algorithms for an ambulatory human operated 3d mobile mapping system," *Remote Sensing*, vol. 5, no. 12, pp. 6611–6646, 2013.

[9] A. Bachrach, S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1320–1343, 2012.

[10] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," in *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE, 2011, pp. 20–25.

[11] ——, "Autonomous indoor 3d exploration with a micro-aerial vehicle," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 9–15.

[12] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 2011, pp. 127–136.

[13] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended kinectfusion," 2012.

[14] J. Fossel, D. Hennes, D. Claes, S. Alers, and K. Tuyls, "OctoSLAM: A 3d mapping approach to situational awareness of unmanned aerial vehicles," in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*. IEEE, 2013, pp. 179–188.

[15] H. A. Karimi, A. J. Khattak, and J. E. Hummer, "Evaluation of mobile mapping systems for roadway data collection," *Journal of computing in civil engineering*, vol. 14, no. 3, pp. 168–173, 2000.

[16] C. Ellum and N. El-Sheimy, "Land-based mobile mapping systems," *Photogrammetric engineering and remote sensing*, vol. 68, no. 1, pp. 13–17, 2002.

[17] R. Van Der Merwe, E. A. Wan, S. Julier *et al.*, "Sigma-point Kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation," in *Proceedings of the AIAA Guidance, Navigation & Control Conference*, 2004, pp. 16–19.

[18] J. L. Crassidis, "Sigma-point Kalman filtering for integrated GPS and inertial navigation," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 42, no. 2, pp. 750–756, 2006.

[19] A. Doucet, N. De Freitas, K. Murphy, and S. Russell, "Rao-Blackwellised particle filtering for dynamic Bayesian networks," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000, pp. 176–183.

[20] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit *et al.*, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *AAAI/IAAI*, 2002, pp. 593–598.

[21] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, 2006.

[22] B. Ferris, D. Fox, and N. D. Lawrence, "WiFi-SLAM using Gaussian process latent variable models." in *IJCAI*, vol. 7, 2007, pp. 2480–2485.

[23] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *Robotics, IEEE Transactions on*, vol. 24, no. 6, pp. 1365–1378, 2008.

[24] Y. Chen, D. Lymberopoulos, J. Liu, and B. Priyantha, "FM-based indoor localization," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 169–182.

[25] V. Otsason, A. Varshavsky, A. LaMarca, and E. De Lara, "Accurate GSM indoor localization," in *UbiComp 2005: Ubiquitous Computing*. Springer, 2005, pp. 141–158.

[26] G. Sibley, L. Matthies, and G. Sukhatme, "A sliding window filter for incremental SLAM," in *Unifying perspectives in computational and robot vision*. Springer, 2008, pp. 103–112.

[27] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *AeroSense'97*. International Society for Optics and Photonics, 1997, pp. 182–193.

[28] E. Wan, R. Van Der Merwe *et al.*, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. IEEE, 2000, pp. 153–158.

[29] E. Lefferts, F. Markley, and M. Shuster, "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, Sept.-Oct. 1982.

[30] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1, pp. 99–141, 2001.

[31] B. Ristic, S. Arulampalam, and N. J. Gordon, *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2004.

[32] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 2432–2437.

[33] 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*, p. 0278364911430419, 2011.

[34] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3607–3613.

[35] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," GT RIM, Tech. Rep. GT-RIM-CP&R-2012-002, Sept 2012. [Online]. Available: https://research.cc.gatech.edu/borg/sites/edu.borg/files/downloads/gtsam.pdf

[36] G. Grisetti, C. Stachniss, and W. Burgard, "Nonlinear constraint network optimization for efficient map learning," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 3, pp. 428–439, 2009.

[37] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4531–4537.

[38] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation for autonomous rotorcraft MAVs in complex environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1758–1764.

[39] G. Huang, M. Kaess, and J. J. Leonard, "Towards consistent visual-inertial navigation," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4926–4933.

[40] M. Li, H. Yu, X. Zheng, A. Mourikis *et al.*, "High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 409–416.

[41] A. T. Erdem and A. O. Ercan, "Fusing inertial sensor data in an Extended Kalman filter for 3d camera tracking," *Image Processing, IEEE Transactions on*, vol. 24, no. 2, pp. 538–548, 2015.

[42] K. J. Wu, A. M. Ahmed, G. A. Georgiou, and S. I. Roumeliotis, "A square root inverse filter for efficient vision-aided inertial navigation on mobile devices," *Robotics: Science and Systems,(Rome, Italy)*, 2015.

[43] A. Tamjidi, C. Ye, and S. Hong, "6-DOF pose estimation of a portable navigation aid for the visually impaired," in *Robotic and Sensors Environments (ROSE), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 178–183.

[44] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4973–4980.

[45] M. Kaess, "Simultaneous localization and mapping with infinite planes."

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

[47] M. Bosse and R. Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based SLAM," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 667–691, 2008.

[48] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1. IEEE, 1999, pp. 666–673.

[49] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2301–2306.

[50] Y. Bok, Y. Jeong, D.-G. Choi, and I. S. Kweon, "Capturing village-level heritages with a hand-held camera-laser fusion sensor," *International Journal of Computer Vision*, vol. 94, no. 1, pp. 36–53, 2011.

[51] F. M. Mirzaei, S. Roumeliotis *et al.*, "A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1143–1156, 2008.

[52] M. Li, B. H. Kim, A. Mourikis *et al.*, "Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4712–4719.

[53] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers." in *Robotics: Science and Systems*, 2013.

[54] J. Rehder, P. Beardsley, R. Siegwart, and P. Furgale, "Spatio-temporal laser to visual/inertial calibration with applications to hand-held, large scale scanning," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 459–465.

[55] M. He, H. Zhao, J. Cui, and H. Zha, "Calibration method for multiple 2d LIDARs system," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3034–3041.

[56] D. J. Aldous, *Exchangeability and related topics*. Springer, 1985.

[57] J. Pitman *et al.*, "Combinatorial stochastic processes," Technical Report 621, Dept. Statistics, UC Berkeley, 2002. Lecture notes for St. Flour course, Tech. Rep., 2002.

[58] M. A. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 3, pp. 381–396, 2002.

[59] K. Lenac, E. Mumolo, and M. Nolich, "Robust and accurate genetic scan matching algorithm for robotic navigation," in *Intelligent Robotics and Applications*. Springer, 2011, pp. 584–593.

[60] J. M. Phillips, R. Liu, and C. Tomasi, "Outlier robust ICP for minimizing fractional RMSD," in *3-D Digital Imaging and Modeling, 2007. 3DIM'07. Sixth International Conference on*. IEEE, 2007, pp. 427–434.

[61] J. Hesch, F. M. Mirzaei, G. L. Mariottini, S. Roumeliotis *et al.*, "A laser-aided inertial navigation system (L-INS) for human localization in unknown indoor environments," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5376–5382.

[62] S. Zhao, J. Farrell *et al.*, "2d LIDAR aided INS for vehicle positioning in urban environments," in *Control Applications (CCA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 376–381.

[63] S. Zhao, "Realtime, decimeter accuracy navigation using sliding window estimator and autonomous vehicle trajectory tracking control," Ph.D. dissertation, University of California, Riverside, 2014.

[64] L. Wei, C. Cappelle, and Y. Ruichek, "Camera/laser/GPS fusion method for vehicle positioning under extended NIS-based sensor validation," *Instrumentation and Measurement, IEEE Transactions on*, vol. 62, no. 11, pp. 3110–3122, 2013.

[65] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3923–3929.

[66] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4974–4981.

[67] J. G. Fryer and D. C. Brown, "Lens distortion for close-range photogrammetry," *Photogrammetric engineering and remote sensing*, no. 52, pp. 51–58, 1986.

[68] G. Bradski *et al.*, "The opencv library," *Doctor Dobbs Journal*, vol. 25, no. 11, pp. 120–126, 2000.

[69] J.-Y. Bouguet, "Camera calibration toolbox for matlab," 2004.

[70] C. B. Duane, "Close-range camera calibration," *Photogram. Eng. Remote Sens*, vol. 37, pp. 855–866, 1971.

[71] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A flexible technique for accurate omnidirectional camera calibration and structure from motion," in *Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on*. IEEE, 2006, pp. 45–45.

[72] ——, "A toolbox for easily calibrating omnidirectional cameras," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 5695–5701.

[73] J. Mallon and P. F. Whelan, "Precise radial un-distortion of images," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 1. IEEE, 2004, pp. 18–21.

[74] Hokuyo, "Scanning range finder UTM-30LX," 2015. [Online]. Available: https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html

[75] F. Pomerleau, A. Breitenmoser, M. Liu, F. Colas, and R. Siegwart, "Noise characterization of depth sensors for surface inspections," in *Applied Robotics for the Power Industry (CARPI), 2012 2nd International Conference on*. IEEE, 2012, pp. 16–21.

[76] P. Demski, M. Mikulski, and R. Koteras, "Characterization of Hokuyo UTM-30LX laser range finder for an autonomous mobile robot," in *Advanced Technologies for Intelligent Systems of National Border Security*. Springer, 2013, pp. 143–153.

[77] G. Chen, J. Kua, S. Shum, N. Naikal, M. Carlberg, and A. Zakhor, "Indoor localization algorithms for a human-operated backpack system," in *3D Data Processing, Visualization, and Transmission*. Citeseer, 2010.

[78] T. Liu, M. Carlberg, G. Chen, J. Chen, J. Kua, and A. Zakhor, "Indoor localization and visualization using a human-operated backpack system," in *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*. IEEE, 2010, pp. 1–10.

[79] J. Kua, N. Corso, and A. Zakhor, "Automatic loop closure detection using multiple cameras for 3d indoor localization," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2012, pp. 82 960V–82 960V.

[80] M. Bosse and R. Zlot, "Keypoint design and evaluation for place recognition in 2d lidar maps," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1211–1224, 2009.

[81] K. Granström, J. Callmer, F. Ramos, and J. Nieto, "Learning to detect loop closure from range data," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 15–22.

[82] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Robotics-DL tentative*. International Society for Optics and Photonics, 1992, pp. 586–606.

[83] M. Bosse and R. Zlot, "Continuous 3d scan-matching with a spinning 2d laser," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 4312–4319.

[84] A. W. Fitzgibbon, "Robust registration of 2d and 3d point sets," *Image and Vision Computing*, vol. 21, no. 13, pp. 1145–1153, 2003.

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

[86] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 3. IEEE, 2002, pp. 545–548.

[87] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[88] K. P. Murphy *et al.*, "Bayesian map learning in dynamic environments." in *NIPS*, 1999, pp. 1015–1021.

[89] G. J. McLachlan and K. E. Basford, "Mixture models: Inference and applications to clustering," *Applied Statistics*, 1988.

[90] T. K. Moon, "The expectation-maximization algorithm," *Signal processing magazine, IEEE*, vol. 13, no. 6, pp. 47–60, 1996.

[91] M. Jordan and Y. W. Teh, "A gentle introduction to the dirichlet process, the beta process, and bayesian nonparametrics," Dept. Statistics, UC Berkeley, 2014. Lecture notes, Tech. Rep., 2014.

[92] R. M. Neal, "Markov chain sampling methods for Dirichlet process mixture models," *Journal of computational and graphical statistics*, vol. 9, no. 2, pp. 249–265, 2000.

[93] C. K. Carter and R. Kohn, "On gibbs sampling for state space models," *Biometrika*, vol. 81, no. 3, pp. 541–553, 1994.

[94] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[95] K.-F. Man, K. S. TANG, and S. Kwong, *Genetic algorithms: Concepts and designs*. Springer Science & Business Media, 2012.

[96] J. L. Martínez, J. González, J. Morales, A. Mandow, and A. J. García-Cerezo, "Mobile robot motion estimation by 2d scan matching with genetic and iterative closest point algorithms," *Journal of Field Robotics*, vol. 23, no. 1, pp. 21–34, 2006.

[97] A. Barla, R. Odone, and A. Verr, "Histogram intersection kernel for image classification," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 3. IEEE, 2003, pp. III–513.

[98] W. H. Press, *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[99] L. Geosystems, "Leica ScanStation C10," http://hds.leica-geosystems.com/en/Leica-ScanStation-C10_79411.htm.

[100] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA., 1967, pp. 281–297.

[101] J. J. Oliver, R. A. Baxter, and C. S. Wallace, "Unsupervised learning using MML," in *ICML*, 1996, pp. 364–372.

[102] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[103] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[104] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.

[105] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual–inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[106] N. Tarawny and S. I. Roumeliotis, "Indirect Kalman filter for 3d pose estimation," MARS Lab, Dept. of Computer Science & Engineering, University of Minnesota, Minneapolis, MN., Tech. Rep., March 2005.

[107] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using Allan variance," *Instrumentation and Measurement, IEEE Transactions on*, vol. 57, no. 1, pp. 140–149, 2008.

[108] I. of Electrical and E. Engineers, *IEEE Standard Specification Format Guide and Test Procedure for Linear, Single-axis, Nongyroscopic Accelerometers*. IEEE, 1999.

[109] W. Flenniken, "Modeling inertial measurement units and anlyzing the effect of their errors in navigation applications," Ph.D. dissertation, 2005.

[110] R. J. Vaccaro and A. S. Zaki, "Statistical modeling of rate gyros," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 3, pp. 673–684, March 2012.

[111] M. Li and A. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[112] J. C. Butcher, *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*. Wiley-Interscience, 1987.

[113] G. Huang, "Toward consistent visual-inertial navigation," Computer Science and Arificial Intelligence Laboratory, Massachusetts Insitute of Technology, Tech. Rep., 2013.

[114] M. Shelley, "Monocular visual inertial odometry on a mobile device," Master's thesis, Technishen Universitat Munchen, 2014.

[115] A. Mourikis, S. Roumeliotis *et al.*, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3565–3572.

[116] M. Li and A. Mourikis, "Optimization-based estimator design for vision-aided inertial navigation," 2013.

[117] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision." in *IJCAI*, vol. 81, 1981, pp. 674–679.

[118] R. W. Hamming, "Error detecting and error correcting codes," *Bell System technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[119] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *Computer Vision–ECCV 2010*, pp. 778–792, 2010.

[120] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[121] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 430–443.

[122] J. Montiel, J. Civera, and A. J. Davison, "Unified inverse depth parametrization for monocular SLAM," *analysis*, vol. 9, p. 1, 2006.

[123] F. Kenney and E. Keeping, "Mathematics of statistics-part two," 1951.

[124] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[125] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936.

[126] C. L. Lawson and R. J. Hanson, *Solving least squares problems*. SIAM, 1974, vol. 161.

[127] T. Pavlidis and S. L. Horowitz, "Segmentation of plane curves," *IEEE transactions on Computers*, no. 8, pp. 860–870, 1974.

[128] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial & Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

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

[130] B. Taylor, *Methodus incrementorum directa & inversa*. Inny, 1717.

[131] T. Schoen and F. Lindsten, "Manipulating the multivariate Gaussian density." Linkoeping University, http://www.rt.isy.liu.se/ schon/Publications/SchonL2011.pdf., Tech. Rep., 2011.

[132] I. Skog, P. Händel, J.-O. Nilsson, and J. Rantakokko, "Zero-velocity detection—an algorithm evaluation," *Biomedical Engineering, IEEE Transactions on*, vol. 57, no. 11, pp. 2657–2666, 2010.

[133] K.-L. Low, "Linear least-squares optimization for point-to-plane ICP surface registration," *Chapel Hill, University of North Carolina*, 2004.

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

[135] C. Z. Mooney, *Monte carlo simulation*. Sage Publications, 1997, vol. 116.

[136] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[137] Velodyne, "Velodyne lidar VLP-16," 2016. [Online]. Available: http://velodynelidar.com/vlp-16.html

Appendix A

Laser Residual Jacobian Derivations

In this Appendix we will derive the Jacobians needed for performing a laser based EKF update as described in Section 4.6.2, namely Equations 4.63 and 4.75. Recall that a line measurement observation, $\mathbf{u}_j = [\rho_j, \phi_j]^T$, defines two geometric constraints for the EKF. Denoted $\mathbf{z}_1$ and $\mathbf{z}_2$, they constrain the orientation and distance of the system to the observed plane respectively. We will derive the Jacobians for each individually.

A.1 Angular Constraint Jacobians

In this section we derive the Jacobians needed to evaluate Equation 4.63. The orientation of the observed line measurement ϕ_j is first used to define a constraint between the orientation of the system and the observed plane $\mathbf{\Pi}_i$.

$$\mathbf{z}_{l_{1j}} = \boldsymbol{\pi}_i^{T\,B}_{\;\;G}\mathbf{R}^T(t+t_n)^G_L\mathbf{R}\,\boldsymbol{l}_{j\|} = 0 \tag{A.1}$$

Here the line $\boldsymbol{l}_{j\|}$ is the 3D line that the laser measures that is in the plane and $\boldsymbol{\pi}_i$ is the normal vector of the plane $\mathbf{\Pi}_i$. Since the state vector and line observations are noisy, the estimated observation is found by evaluating Eq. A.1 using the best current estimates.

$$\hat{\mathbf{z}}_{l_{1j}} = \hat{\boldsymbol{\pi}}_i^{T\,B}_{\;\;G}\hat{\mathbf{R}}^T(t+\hat{t}_n)^G_L\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\|} \tag{A.2}$$

The residual vector is then the difference between the expected measurement and the observed measurement.

$$\begin{aligned}\mathbf{r}_{l_{1j}} &= \mathbf{z}_{l_{1j}} - \hat{\mathbf{z}}_{l_{1j}} \\ &\approx \mathbf{H}_{l_{1j}B}\tilde{\mathbf{x}}_B + \mathbf{H}_{l_{1j}L}\tilde{\mathbf{x}}_L + \mathbf{H}_{l_{1j}P_i}\tilde{\mathbf{x}}_{P_i} + \mathbf{\Gamma}_{l_{1j}}\tilde{\mathbf{u}}_j\end{aligned} \tag{A.3}$$

To derive the linearized version of the residual and the required Jacobians, we begin first by substituting the expression for $\mathbf{z}_1$ into Eq. A.3. We note that the derivation is identical for the case of vertical and horizontal planes with the exception that the contributions for the normal angle θ_i are omitted for horizontal planes.

$$\begin{aligned}\mathbf{r}_{l_{1j}} &= \boldsymbol{\pi}_i^T {}^B_G\mathbf{R}^T(t+t_n){}^G_L\mathbf{R}\,\boldsymbol{l}_{j\|} \\ &\quad - \hat{\mathbf{z}}_1\end{aligned} \tag{A.4}$$

By applying first order Taylor expansion to the normal vector, we note that the normal vector can be approximated as a function of the error in the normal angle.

$$\boldsymbol{\pi}_i \approx \hat{\boldsymbol{\pi}}_i - \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor \mathbf{e}_z \tilde{\theta}_i \tag{A.5}$$

The above approximation is used to expand the $\boldsymbol{\pi}_i$ term.

$$\begin{aligned}\mathbf{r}_{l_{1j}} &= \hat{\boldsymbol{\pi}}_i^T {}^B_G\mathbf{R}^T(t+t_n){}^G_L\mathbf{R}\,\boldsymbol{l}_{j\|} \\ &\quad + \mathbf{e}_z^T \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor {}^B_G\mathbf{R}^T(t+t_n){}^G_L\mathbf{R}\,\boldsymbol{l}_{j\|}\tilde{\theta}_i \\ &\quad - \hat{\mathbf{z}}_1\end{aligned} \tag{A.6}$$

Next we approximate the true rotation ${}^B_G\mathbf{R}(t+t_n)$ by successively applying Taylor approximation for both the rotation component and the temporal component.

$${}^B_G\mathbf{R}(t+t_n) \approx {}^B_G\hat{\mathbf{R}}(t+\hat{t}_n)\big(\mathbf{I}_{3\times 3} - \lfloor \tilde{\boldsymbol{\theta}}^G(t+\hat{t}_n)\times \rfloor\big) - \lfloor {}^B\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor {}^B_G\hat{\mathbf{R}}(t+\hat{t}_n)\tilde{t}_n \tag{A.7}$$

The above expression is then substituted into the residual expression. We note that any 2^{nd} order error terms generated from here on are omitted to maintain the linear requirements of the EKF.

$$\begin{aligned}\mathbf{r}_{l_{1j}} &= \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n){}^G_L\mathbf{R}\,\boldsymbol{l}_{j\|} \\ &\quad + \hat{\boldsymbol{\pi}}_i^T \lfloor \tilde{\boldsymbol{\theta}}^G(t+\hat{t}_n)\times \rfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n){}^G_L\mathbf{R}\,\boldsymbol{l}_{j\|} \\ &\quad + \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)\lfloor {}^B\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor {}^G_L\mathbf{R}\,\boldsymbol{l}_{j\|}\tilde{t}_n \\ &\quad + \mathbf{e}_z^T \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n){}^G_L\mathbf{R}\,\boldsymbol{l}_{j\|}\tilde{\theta}_i \\ &\quad - \hat{\mathbf{z}}_1\end{aligned} \tag{A.8}$$

In a similar fashion the rotation between the IMU and the laser frames of reference can be estimated using the small angle approximation.

$${}^B_L\mathbf{R} \approx {}^B_L\hat{\mathbf{R}}\big(\mathbf{I}_{3\times 3} - \lfloor \tilde{\boldsymbol{\theta}}^L\times \rfloor\big) \tag{A.9}$$

Which allows for the expansion of the extrinsic rotation term ${}^B_L\mathbf{R}$ in Eq. A.8.

$$
\begin{aligned}
\mathbf{r}_{l_{1j}} &= \hat{\boldsymbol{\pi}}_i^T {}_G^B\hat{\mathbf{R}}^T(t+\hat{t}_n){}_L^G\hat{\mathbf{R}}\, \boldsymbol{l}_{j\|} \\
&- \hat{\boldsymbol{\pi}}_i^T {}_G^B\hat{\mathbf{R}}^T(t+\hat{t}_n){}_L^G\hat{\mathbf{R}}\lfloor\tilde{\boldsymbol{\theta}}^L\times\rfloor\, \boldsymbol{l}_{j\|} \\
&+ \hat{\boldsymbol{\pi}}_i^T\lfloor\tilde{\boldsymbol{\theta}}^G(t+\hat{t}_n)\times\rfloor {}_G^B\hat{\mathbf{R}}^T(t+\hat{t}_n){}_L^G\hat{\mathbf{R}}\, \boldsymbol{l}_{j\|} \\
&+ \hat{\boldsymbol{\pi}}_i^T {}_G^B\hat{\mathbf{R}}^T(t+\hat{t}_n)\lfloor{}^B\hat{\boldsymbol{\omega}}(t+\hat{t}_n)\times\rfloor {}_L^G\hat{\mathbf{R}}\, \boldsymbol{l}_{j\|}\tilde{t}_n \\
&+ \mathbf{e}_z^T\lfloor\hat{\boldsymbol{\pi}}_i\times\rfloor {}_G^B\hat{\mathbf{R}}^T(t+\hat{t}_n){}_L^G\hat{\mathbf{R}}\, \boldsymbol{l}_{j\|}\tilde{\theta}_i \\
&- \hat{\mathbf{z}}_1
\end{aligned}
\tag{A.10}
$$

The final term that must be expanded is the in-plane line $\boldsymbol{l}_{j\|}$. First order Taylor expansion is used to find the best linear estimate of $\boldsymbol{l}_{j\|}$.

$$
\boldsymbol{l}_{j\|} \approx \hat{\boldsymbol{l}}_{j\|} - \lfloor\, \hat{\boldsymbol{l}}_{j\|}\times\rfloor \mathbf{e}_z\tilde{\phi}_j \tag{A.11}
$$

The last substitution that must be made is for the line measurement observation $\boldsymbol{l}_{j\|}$.

$$
\begin{aligned}
\mathbf{r}_{l_{1j}} &= \hat{\boldsymbol{\pi}}_i^T {}_G^B\hat{\mathbf{R}}^T(t+\hat{t}_n){}_L^G\hat{\mathbf{R}}\, \hat{\boldsymbol{l}_{j\|}} \\
&+ \hat{\boldsymbol{\pi}}_i^T {}_G^B\hat{\mathbf{R}}^T(t+\hat{t}_n){}_L^G\hat{\mathbf{R}}\lfloor\, \hat{\boldsymbol{l}_{j\|}}\times\rfloor \mathbf{e}_z\tilde{\phi}_j \\
&- \hat{\boldsymbol{\pi}}_i^T {}_G^B\hat{\mathbf{R}}^T(t+\hat{t}_n){}_L^G\hat{\mathbf{R}}\lfloor\tilde{\boldsymbol{\theta}}^L\times\rfloor\, \hat{\boldsymbol{l}_{j\|}} \\
&+ \hat{\boldsymbol{\pi}}_i^T\lfloor\tilde{\boldsymbol{\theta}}^G(t+\hat{t}_n)\times\rfloor {}_G^B\hat{\mathbf{R}}^T(t+\hat{t}_n){}_L^G\hat{\mathbf{R}}\, \hat{\boldsymbol{l}_{j\|}} \\
&+ \hat{\boldsymbol{\pi}}_i^T {}_G^B\hat{\mathbf{R}}^T(t+\hat{t}_n)\lfloor{}^B\hat{\boldsymbol{\omega}}(t+\hat{t}_n)\times\rfloor {}_L^G\hat{\mathbf{R}}\, \hat{\boldsymbol{l}_{j\|}}\tilde{t}_n \\
&- \mathbf{e}_z^T\lfloor\hat{\boldsymbol{\pi}}_i\times\rfloor {}_G^B\hat{\mathbf{R}}^T(t+\hat{t}_n){}_L^G\hat{\mathbf{R}}\, \hat{\boldsymbol{l}_{j\|}}\tilde{\theta}_i \\
&- \hat{\mathbf{z}}_1
\end{aligned}
\tag{A.12}
$$

Examining Eq. A.12 we notice that the first line is exactly the definition of $\hat{\mathbf{z}}_1$ and thus we are able to cancel that term from the expression. Furthermore, we note that the above expression contains the IMU body state's angular error at time $t+\hat{t}_n$ instead of at time t. Applying Taylor expansion we get an expression for $\tilde{\boldsymbol{\theta}}^G$ at time t.

$$
\begin{aligned}
\tilde{\boldsymbol{\theta}}^G(t+\hat{t}_n) &\approx \tilde{\boldsymbol{\theta}}^G(t) + \hat{t}_n{}^B\tilde{\boldsymbol{\omega}}(t) \\
&\approx \tilde{\boldsymbol{\theta}}^G(t) - \hat{t}_n\tilde{\mathbf{b}}_g
\end{aligned}
\tag{A.13}
$$

In practice, neglecting the reliance on the error in the gyroscope bias has minimal effect and can be omitted. To get Eq. A.12 in a the correct form, we use the cross product vector identity $\mathbf{a}^T\lfloor\mathbf{b}\times\rfloor = -\mathbf{b}^T\lfloor\mathbf{a}\times\rfloor$ to rearrange all terms.

$$\begin{aligned}\mathbf{r}_{l_{1j}} &= -\hat{\boldsymbol{\pi}}_i^T \lfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^G_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\|} \times \rfloor \tilde{\boldsymbol{\theta}}^G(t) \\ &+ \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^G_L\hat{\mathbf{R}} \lfloor \hat{\boldsymbol{l}}_{j\|} \times \rfloor \tilde{\boldsymbol{\theta}}^L \\ &+ \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) \lfloor {}^B\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor {}^G_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\|} \tilde{t}_n \\ &+ \mathbf{e}_z^T \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^G_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\|} \tilde{\theta}_i \\ &- \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^G_L\hat{\mathbf{R}} \lfloor \hat{\boldsymbol{l}}_{j\|} \times \rfloor \mathbf{e}_z \tilde{\phi}_j \end{aligned} \tag{A.14}$$

The timing error $\tilde{t}_n$ is easily expressed as a function of the time delay error and the rolling shutter timing error.

$$\begin{aligned}\tilde{t}_n &= t_n - \hat{t}_n \\ &= \tilde{t}_d + \frac{k}{N}\tilde{t}_r \end{aligned} \tag{A.15}$$

Plugging the expanded form of $\tilde{t}_n$ into Eq. A.14 we obtain the final expression for $\mathbf{r}_{ij}$.

$$\begin{aligned}\mathbf{r}_{l_{1j}} &= -\hat{\boldsymbol{\pi}}_i^T \lfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^G_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\|} \times \rfloor \tilde{\boldsymbol{\theta}}^G(t) \\ &+ \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^G_L\hat{\mathbf{R}} \lfloor \hat{\boldsymbol{l}}_{j\|} \times \rfloor \tilde{\boldsymbol{\theta}}^L \\ &+ \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) \lfloor {}^B\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor {}^G_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\|} \tilde{t}_d \\ &+ \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) \lfloor {}^B\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor {}^G_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\|} \frac{k}{N} \tilde{t}_r \\ &+ \mathbf{e}_z^T \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^G_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\|} \tilde{\theta}_i \\ &- \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^G_L\hat{\mathbf{R}} \lfloor \hat{\boldsymbol{l}}_{j\|} \times \rfloor \mathbf{e}_z \tilde{\phi}_j \end{aligned} \tag{A.16}$$

Examining Eq. A.16, the elements of the Jacobians from Eq. A.3 are readily available. The Jacobian of the residual with respect to the IMU body state is denoted by $\mathbf{H}_{l_{1j}B}$

$$\mathbf{H}_{l_{1j}B} = \begin{bmatrix} -\hat{\boldsymbol{\pi}}_i^T \lfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^G_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\|} \times \rfloor & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \end{bmatrix} \tag{A.17}$$

The Jacobian of the residual with respect to the laser calibration parameters is $\mathbf{H}_{l_{1j}L}$.

$$\mathbf{H}_{l_{1j}L} = \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) \begin{bmatrix} {}^B_L\hat{\mathbf{R}} \lfloor \hat{\boldsymbol{l}}_{j\|} \times \rfloor & \mathbf{0}_{3\times 3} & \mathbf{M}_j & \frac{n}{N}\mathbf{M}_j \end{bmatrix} \tag{A.18}$$

Where the matrix $\mathbf{M}_j$ is a compact notation for the cross product of the rotational velocity and observed line.

$$\mathbf{M}_j = \lfloor \hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\|} \tag{A.19}$$

The Jacobian with respect to the observed plane's parameters is denoted as $\mathbf{H}_{l_{1j}P_i}$. For horizontal planes this matrix is the 1×1 zero matrix, but for vertical planes there is a reliance on the normal angle of the plane θ_i.

$$\mathbf{H}_{l_{1j}P_i} = \begin{bmatrix} \mathbf{0}_{1\times 1} & \mathbf{e}_z^T \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor {}^B_G\hat{\mathbf{R}}^T (t + \hat{t}_n) {}^B_L\hat{\mathbf{R}} \, \hat{\boldsymbol{l}}_{j\parallel} \end{bmatrix} \tag{A.20}$$

The final Jacobian, $\boldsymbol{\Gamma}_{l_{1j}}$, relates the line observations to the residual.

$$\boldsymbol{\Gamma}_{l_{1j}} = \begin{bmatrix} \mathbf{0}_{1\times 1} & -\hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T (t + \hat{t}_n) {}^B_L\hat{\mathbf{R}} \lfloor \hat{\boldsymbol{l}}_{j\parallel} \times \rfloor \mathbf{e}_z \end{bmatrix} \tag{A.21}$$

The Jacobians $\mathbf{H}_{l_{1j}B}$, $\mathbf{H}_{l_{1j}L}$, $\mathbf{H}_{l_{1j}P_i}$, and $\boldsymbol{\Gamma}_{l_{1j}}$ are the exact expressions recovered from Equations 4.64, 4.65, 4.69, and 4.70 respectively.

A.2 Distance Constraint Jacobians

In this section we will derive the Jacobians needed for evaluating Equation 4.75. The second constraint derived from the line observation measurement $\mathbf{u}_j$ relates the observed and expected orthogonal distances to the plane $\boldsymbol{\Pi}_i$. As defined in Sec. 4.6.2, the expected distance to plane d_i is computed by projecting the distance the laser scanner measures to the plane along the plane's normal vector.

$$\mathbf{z}_{l_{2j}} = \boldsymbol{\pi}_i^T \big({}^G\mathbf{p}_B(t + t_n) + {}^B_G\mathbf{R}^T (t + t_n) ({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R} \, \boldsymbol{l}_{j\perp}) \big) - d_i = 0 \tag{A.22}$$

The line $\boldsymbol{l}_{j\perp}$ is the direction of shortest distance to the laser scanner and orthogonal to the in-plane line $\boldsymbol{l}_{j\parallel}$. Again, we form the estimated measurement by evaluating Eq. A.22 using the current state estimates.

$$\hat{\mathbf{z}}_{l_{2j}} = \hat{\boldsymbol{\pi}}_i^T \big({}^G\hat{\mathbf{p}}_B(t + \hat{t}_n) + {}^B_G\hat{\mathbf{R}}^T (t + \hat{t}_n) ({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}} \, \hat{\boldsymbol{l}}_{j\perp}) \big) - \hat{d}_i \tag{A.23}$$

Again, we then define the residual vector as the difference in the expected observation and the measured observation.

$$\begin{aligned} \mathbf{r}_{l_{2j}} &= \mathbf{z}_{l_{2j}} - \hat{\mathbf{z}}_{l_{2j}} \\ &= \mathbf{H}_{l_{2j}B} \tilde{\mathbf{x}}_B + \mathbf{H}_{l_{2j}L} \tilde{\mathbf{x}}_L + \mathbf{H}_{l_{2j}P_i} \tilde{\mathbf{x}}_{P_i} + \boldsymbol{\Gamma}_{l_{2j}} \tilde{\mathbf{u}}_j \end{aligned} \tag{A.24}$$

We again begin the Jacobian derivation by substituting the definition of $\mathbf{z}_{l_{2j}}$ from Eq. A.22 into the residual expression.

$$\begin{aligned} \mathbf{r}_{l_{2j}} = {} & \boldsymbol{\pi}_i^T \big({}^G\mathbf{p}_B(t + t_n) + {}^B_G\mathbf{R}^T (t + t_n) ({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R} \, \boldsymbol{l}_{j\perp}) \big) - d_i \\ & - \hat{\mathbf{z}}_{l_{2j}} \end{aligned} \tag{A.25}$$

Using the best linear estimate of $\boldsymbol{\pi}_i$, we substitute its linearized expression from Eq. A.5 into the residual expression above.

$$\begin{aligned}\mathbf{r}_{l_{2j}} &= \hat{\boldsymbol{\pi}}_i^T \big({}^G\mathbf{p}_B(t+t_n) + {}^B_G\mathbf{R}^T(t+t_n)({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\, \boldsymbol{l}_{j\perp}) \big) - d_i \\ &\quad + \mathbf{e}_z \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor \big({}^G\mathbf{p}_B(t+t_n) + {}^B_G\mathbf{R}^T(t+t_n)({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\, \boldsymbol{l}_{j\perp}) \big) \tilde{\theta}_i \\ &\quad - \hat{\mathbf{z}}_{l_{2j}}\end{aligned} \tag{A.26}$$

Next, the ${}^G\mathbf{p}_B(t+t_n)$ term must be rewritten in terms of elements in the state vector. By applying first order Taylor expansion, we eliminate t_n and replace it with its estimate.

$${}^G\mathbf{p}_B(t+t_n) \approx {}^G\mathbf{p}_B(t+\hat{t}_n) + {}^G\mathbf{v}_B(t+\hat{t}_n)\tilde{t}_n \tag{A.27}$$

The above expression contains the position and velocity at time $t+\hat{t}_n$ but the error state has those quantities at time t. We obtain an expression that is in the correct form by applying linear approximation again and ignoring any higher order terms.

$$\begin{aligned}{}^G\mathbf{p}_B(t+t_n) &\approx {}^G\mathbf{p}_B(t+\hat{t}_n) + {}^G\mathbf{v}_B(t+\hat{t}_n)\tilde{t}_n \\ &\approx {}^G\hat{\mathbf{p}}_B(t+\hat{t}_n) + {}^G\tilde{\mathbf{p}}_B(t) + \hat{t}_n {}^G\tilde{\mathbf{v}}_B(t) + {}^G\hat{\mathbf{v}}_B(t+\hat{t}_n)\tilde{t}_n\end{aligned} \tag{A.28}$$

Eq. A.28 now only contains errors at time t and state estimates at time $\hat{t}_n$ as needed by the EKF. Since the filter estimates both sets of quantities we now can continue by substituting the previous expression into the residual equation.

$$\begin{aligned}\mathbf{r}_{l_{2j}} &= \hat{\boldsymbol{\pi}}_i^T \big({}^G\hat{\mathbf{p}}_B(t+t_n) + {}^B_G\mathbf{R}^T(t+t_n)({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\, \boldsymbol{l}_{j\perp}) \big) - d_i \\ &\quad + \hat{\boldsymbol{\pi}}_i^T {}^G\tilde{\mathbf{p}}_B(t) + \hat{\boldsymbol{\pi}}_i^T \hat{t}_n {}^G\tilde{\mathbf{v}}_B(t) \\ &\quad + \hat{\boldsymbol{\pi}}_i^T {}^G\hat{\mathbf{v}}_B(t+\hat{t}_n)\tilde{t}_n \\ &\quad + \mathbf{e}_z \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor \big({}^G\hat{\mathbf{p}}_B(t+t_n) + {}^B_G\mathbf{R}^T(t+t_n)({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\, \boldsymbol{l}_{j\perp}) \big) \tilde{\theta}_i \\ &\quad - \hat{\mathbf{z}}_{l_{2j}}\end{aligned} \tag{A.29}$$

Next we apply the approximation from Eq. A.7 to expand the IMU body rotation ${}^B_G\mathbf{R}(t+t_n)$.

$$\begin{aligned}\mathbf{r}_{l_{2j}} &= \hat{\boldsymbol{\pi}}_i^T \big({}^G\hat{\mathbf{p}}_B(t+t_n) + {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\, \boldsymbol{l}_{j\perp}) \big) - d_i \\ &\quad + \hat{\boldsymbol{\pi}}_i^T \lfloor \tilde{\boldsymbol{\theta}}^G(t) \times \rfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\, \boldsymbol{l}_{j\perp}) \\ &\quad + \hat{\boldsymbol{\pi}}_i^T \big({}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) \lfloor {}^B\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor ({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\, \boldsymbol{l}_{j\perp}) + {}^G\hat{\mathbf{v}}_B(t+\hat{t}_n) \big) \tilde{t}_n \\ &\quad + \hat{\boldsymbol{\pi}}_i^T {}^G\tilde{\mathbf{p}}_B(t) + \hat{\boldsymbol{\pi}}_i^T \hat{t}_n {}^G\tilde{\mathbf{v}}_B(t) \\ &\quad + \mathbf{e}_z \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor \big({}^G\hat{\mathbf{p}}_B(t+t_n) + {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\, \boldsymbol{l}_{j\perp}) \big) \tilde{\theta}_i \\ &\quad - \hat{\mathbf{z}}_{l_{2j}}\end{aligned} \tag{A.30}$$

The translation between the IMU and the laser can be simply expanded using the definition

of the error state ${}^{B}\mathbf{p}_L = {}^{B}\hat{\mathbf{p}}_L + {}^{B}\tilde{\mathbf{p}}_L$.

$$\begin{aligned}
\mathbf{r}_{l_{2j}} &= \hat{\boldsymbol{\pi}}_i^T \big({}^{G}\hat{\mathbf{p}}_B(t+t_n) + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \rho_j {}^{B}_{L}\mathbf{R}\, \boldsymbol{l}_{j\perp}) \big) - d_i \\
&+ \hat{\boldsymbol{\pi}}_i^T {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}\tilde{\mathbf{p}}_L \\
&+ \hat{\boldsymbol{\pi}}_i^T \lfloor \tilde{\boldsymbol{\theta}}^{G}(t) \times \rfloor {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \rho_j {}^{B}_{L}\mathbf{R}\, \boldsymbol{l}_{j\perp}) \\
&+ \hat{\boldsymbol{\pi}}_i^T \big({}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) \lfloor {}^{B}\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor ({}^{B}\hat{\mathbf{p}}_L + \rho_j {}^{B}_{L}\mathbf{R}\, \boldsymbol{l}_{j\perp}) + {}^{G}\hat{\mathbf{v}}_B(t+\hat{t}_n) \big) \tilde{t}_n \\
&+ \hat{\boldsymbol{\pi}}_i^T {}^{G}\tilde{\mathbf{p}}_B(t) + \hat{\boldsymbol{\pi}}_i^T \hat{t}_n {}^{G}\tilde{\mathbf{v}}_B(t) \\
&+ \mathbf{e}_z \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor \big({}^{G}\hat{\mathbf{p}}_B(t+t_n) + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \rho_j {}^{B}_{L}\mathbf{R}\, \boldsymbol{l}_{j\perp}) \big) \tilde{\theta}_i \\
&- \hat{\mathbf{z}}_{l_{2j}}
\end{aligned} \tag{A.31}$$

In a similar fashion, we expand the lasers measured distance ρ_j along the line $\boldsymbol{l}_{i\perp}$ and substitute it into Eq. A.32.

$$\begin{aligned}
\mathbf{r}_{l_{2j}} &= \hat{\boldsymbol{\pi}}_i^T \big({}^{G}\hat{\mathbf{p}}_B(t+t_n) + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\mathbf{R}\, \boldsymbol{l}_{j\perp}) \big) - d_i \\
&+ \hat{\boldsymbol{\pi}}_i^T {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}_{L}\mathbf{R}\, \boldsymbol{l}_{j\perp} \tilde{\rho}_j \\
&+ \hat{\boldsymbol{\pi}}_i^T {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}\tilde{\mathbf{p}}_L \\
&+ \hat{\boldsymbol{\pi}}_i^T \lfloor \tilde{\boldsymbol{\theta}}^{G}(t) \times \rfloor {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\mathbf{R}\, \boldsymbol{l}_{j\perp}) \\
&+ \hat{\boldsymbol{\pi}}_i^T \big({}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) \lfloor {}^{B}\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor ({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\mathbf{R}\, \boldsymbol{l}_{j\perp}) + {}^{G}\hat{\mathbf{v}}_B(t+\hat{t}_n) \big) \tilde{t}_n \\
&+ \hat{\boldsymbol{\pi}}_i^T {}^{G}\tilde{\mathbf{p}}_B(t) + \hat{\boldsymbol{\pi}}_i^T \hat{t}_n {}^{G}\tilde{\mathbf{v}}_B(t) \\
&+ \mathbf{e}_z \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor \big({}^{G}\hat{\mathbf{p}}_B(t+t_n) + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\mathbf{R}\, \boldsymbol{l}_{j\perp}) \big) \tilde{\theta}_i \\
&- \hat{\mathbf{z}}_{l_{2j}}
\end{aligned} \tag{A.32}$$

We then expand ${}^{B}_{L}\mathbf{R}$ by reusing the linearizion from Eq. A.9.

$$\begin{aligned}
\mathbf{r}_{l_{2j}} &= \hat{\boldsymbol{\pi}}_i^T \big({}^{G}\hat{\mathbf{p}}_B(t+t_n) + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\, \boldsymbol{l}_{j\perp}) \big) - d_i \\
&+ \hat{\boldsymbol{\pi}}_i^T {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}} \lfloor \tilde{\boldsymbol{\theta}}^{L} \times \rfloor \, \boldsymbol{l}_{j\perp} \\
&+ \hat{\boldsymbol{\pi}}_i^T {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}_{L}\hat{\mathbf{R}}\, \boldsymbol{l}_{j\perp} \tilde{\rho}_j \\
&+ \hat{\boldsymbol{\pi}}_i^T {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}\tilde{\mathbf{p}}_L \\
&+ \hat{\boldsymbol{\pi}}_i^T \lfloor \tilde{\boldsymbol{\theta}}^{G}(t) \times \rfloor {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\, \boldsymbol{l}_{j\perp}) \\
&+ \hat{\boldsymbol{\pi}}_i^T \big({}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) \lfloor {}^{B}\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor ({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\, \boldsymbol{l}_{j\perp}) + {}^{G}\hat{\mathbf{v}}_B(t+\hat{t}_n) \big) \tilde{t}_n \\
&+ \hat{\boldsymbol{\pi}}_i^T {}^{G}\tilde{\mathbf{p}}_B(t) + \hat{\boldsymbol{\pi}}_i^T \hat{t}_n {}^{G}\tilde{\mathbf{v}}_B(t) \\
&+ \mathbf{e}_z \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor \big({}^{G}\hat{\mathbf{p}}_B(t+t_n) + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\, \boldsymbol{l}_{j\perp}) \big) \tilde{\theta}_i \\
&- \hat{\mathbf{z}}_{l_{2j}}
\end{aligned} \tag{A.33}$$

The final term that needs to be expanded is the line $\boldsymbol{l}_{j\perp}$. The best linear approximation of $\boldsymbol{l}_{j\perp}$ is found by linearizing it around the observed measurement $\hat{\phi}_j$.

$$\boldsymbol{l}_{j\perp} \approx \hat{\boldsymbol{l}}_{j\perp} - \lfloor\, \hat{\boldsymbol{l}}_{j\perp} \times \rfloor \mathbf{e}_z \tilde{\phi}_j \tag{A.34}$$

Substituting Eq. A.34 into the expression of Eq. A.33 and expanding d_i into $\hat{d}_i + \tilde{d}_i$ yields the fully expanded expression for the residual.

$$\begin{aligned}
\mathbf{r}_{l_{2_j}} = & \hat{\boldsymbol{\pi}}_i^T \big({}^G\hat{\mathbf{p}}_B(t+t_n) + {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) \big) \\
& - \tilde{d}_i \\
& + \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)\hat{\rho}_j {}^B_L\hat{\mathbf{R}} \lfloor\, \hat{\boldsymbol{l}}_{j\perp} \times \rfloor \mathbf{e}_z \tilde{\phi}_j \\
& + \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)\hat{\rho}_j {}^B_L\hat{\mathbf{R}} \lfloor \tilde{\boldsymbol{\theta}}^L \times \rfloor\, \hat{\boldsymbol{l}}_{j\perp} \\
& + \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp} \tilde{\rho}_j \\
& + \hat{\boldsymbol{\pi}}_i^T {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^B\tilde{\mathbf{p}}_L \\
& + \hat{\boldsymbol{\pi}}_i^T \lfloor \tilde{\boldsymbol{\theta}}^G(t) \times \rfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) \\
& + \hat{\boldsymbol{\pi}}_i^T \big({}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) \lfloor {}^B\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor ({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) + {}^G\hat{\mathbf{v}}_B(t+\hat{t}_n) \big) \tilde{t}_n \\
& + \hat{\boldsymbol{\pi}}_i^T {}^G\tilde{\mathbf{p}}_B(t) + \hat{\boldsymbol{\pi}}_i^T \hat{t}_n {}^G\tilde{\mathbf{v}}_B(t) \\
& + \mathbf{e}_z \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor \big({}^G\hat{\mathbf{p}}_B(t+t_n) + {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) \big) \tilde{\theta}_i \\
& - \hat{\mathbf{z}}_{l_{2_j}}
\end{aligned} \tag{A.35}$$

Rearranging the above expression into state order and expanding $\tilde{t}_n$ according to Eq. A.15 allows us to easily see the required Jacobian expressions. The Jacobian of $\mathbf{r}_{l_{2_j}}$ is denoted as $\mathbf{H}_{l_{2_j}B}$.

$$\mathbf{H}_{l_{2_j}B} = \hat{\boldsymbol{\pi}}_i^T \begin{bmatrix} -\lfloor {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) \times \rfloor & \mathbf{I}_{3\times3} & (\hat{t}_n)\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{A.36}$$

Similarly the expression for the Jacobian with respect to the laser calibration parameters $\mathbf{H}_{l_{2_j}L}$ is easily found by examining Eq. A.35.

$$\mathbf{H}_{l_{2_j}L} = \hat{\boldsymbol{\pi}}_i^T \begin{bmatrix} {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)\hat{\rho}_j {}^B_L\hat{\mathbf{R}} \lfloor \hat{\boldsymbol{l}}_{j\perp} \times \rfloor & {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n) & \mathbf{S}_j & \frac{n}{N}\mathbf{S}_j \end{bmatrix} \tag{A.37}$$

The matrix $\mathbf{S}_j$ is defined as via Eq. A.38.

$$\mathbf{S}_j = {}^B_G\hat{\mathbf{R}}^T(\hat{t}_n) \lfloor \hat{\boldsymbol{\omega}}(\hat{t}_n) \times \rfloor ({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) + {}^G\hat{\mathbf{v}}_B(\hat{t}_n) \tag{A.38}$$

We note that the Jacobian for the plane states $\mathbf{H}_{l_{2_j}P_i}$ is identical for horizontal and vertical planes with the exception that the horizontal case has reduced dimensionality as there is no normal angle.

$$\mathbf{H}_{l_{2_j}P_i} = \begin{bmatrix} -\mathbf{I}_{1\times1} & \mathbf{e}_z^T \lfloor \hat{\boldsymbol{\pi}}_i \times \rfloor \big({}^G\hat{\mathbf{p}}_B(t+\hat{t}_n) + {}^B_G\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^B\hat{\mathbf{p}}_L + \hat{\rho}_j {}^B_L\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\perp}) \big) \end{bmatrix} \tag{A.39}$$

The final Jacobian needed relates the input error to the residual and is denoted as $\mathbf{\Gamma}_{l_{2j}}$.

$$\mathbf{\Gamma}_{l_{2j}} = \hat{\boldsymbol{\pi}}_i^{T\,B}_{\;\;G}\hat{\mathbf{R}}^T(t+\hat{t}_n)^B_L\hat{\mathbf{R}}\begin{bmatrix}\hat{\boldsymbol{l}}_{j\perp} & -\hat{\rho}_j\lfloor\hat{\boldsymbol{l}}_{j\perp}\times\rfloor\mathbf{e}_z\end{bmatrix} \tag{A.40}$$

The derived expressions for the Jacobians $\mathbf{H}_{l_{2j}B}$, $\mathbf{H}_{l_{2j}L}$, $\mathbf{H}_{l_{2j}P_i}$, and $\mathbf{\Gamma}_{l_{2j}}$ are exactly those described in Equations 4.76, 4.77, 4.80, and 4.81 respectively.

Appendix B

Plane Augmentation Jacobian Derivations

In this section we will derive the Jacobians needed to perform new plane initialization, namely those needed to evaluate Equations 4.92, 4.96, and 4.105. When a laser measurement $\mathbf{u}_j$ is found to originate from a horizontal or vertical plane not stored in the EKF state vector, it is used to create a new plane and is augmented into the plane map $\mathbf{x}_P$. In addition to an estimate of the plane parameters $\mathbf{x}_{P_i}$, the EKF needs an estimate of the error in those parameters to augment into the covariance matrix. Since the plane parameters are a function of both the current state estimate and the laser observation, a function that relates them is needed for proper augmentation of the covariance matrix. We will derive the required Jacobians for the normal angle and offset parameters individually. Note that for the case of a horizontal plane, there is no normal angle and thus the augmentation Jacobians will only contain the contributions for the offset parameter.

B.1 Normal Angle Jacobians

A vertical plane's normal angle is defined as the angle that the normal vector makes in the global xy plane. Mathematically, this is equivalent to saying it is the four-quadrant arctangent of the x and y components of the normal vector.

$$\theta_i = \text{atan2}(v, u) \tag{B.1}$$

Here u and v define the x and y components of the normal vector respectively and are denoted as $\mathbf{y} = [u, v]^T$. We compute the vector y by taking the in-plane line $\boldsymbol{l}_{j\|}$, converting it to global coordinates, taking the cross product of it with the global z vector, and projecting it down into the xy plane.

$$\begin{aligned}\mathbf{y} &= \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix} \lfloor \mathbf{e}_z \times \rfloor {}^{B}_{G}\mathbf{R}^T (t+t_n) {}^{B}_{L}\mathbf{R}\, \boldsymbol{l}_{j\parallel} \\ &= \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix} \lfloor \mathbf{e}_z \times \rfloor \mathbf{f}(\mathbf{x}, \mathbf{u}_j)\end{aligned} \tag{B.2}$$

The derivation begins by first evaluating the expression for θ_i using the observed line measurement feature $\hat{\mathbf{u}}_j$ and state estimate $\hat{\mathbf{x}}$.

$$\hat{\theta}_i = \text{atan2}(\hat{v}, \hat{u}) \tag{B.3}$$

with

$$\hat{\mathbf{y}} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix} \lfloor \mathbf{e}_z \times \rfloor {}^{B}_{G}\hat{\mathbf{R}}^T (t+t_n) {}^{B}_{L}\hat{\mathbf{R}}\, \hat{\boldsymbol{l}}_{j\parallel} \tag{B.4}$$

Since the EKF requires the expression in error state form, we create the error from by subtracting the true and estimated version of θ_i.

$$\begin{aligned}\tilde{\theta}_i &= \theta_i - \hat{\theta}_i \\ &\approx \mathbf{J}_{\mathbf{x}}(\theta_i)\Big|_{\mathbf{x}=\hat{\mathbf{x}}} \tilde{\mathbf{x}} + \mathbf{J}_{\mathbf{u}_j}(\theta_i)\Big|_{\mathbf{u}_j=\hat{\mathbf{u}}_j} \tilde{\mathbf{u}}_j\end{aligned} \tag{B.5}$$

In the above expression $\mathbf{J}_{\mathbf{x}}(\theta_i)$ indicates the Jacobian of θ_i with respect to the variable $\mathbf{x}$. The required Jacobians are found using the chain rule on Eq. B.1.

$$\begin{aligned}\mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\theta_i) &= \mathbf{J}_{\mathbf{y}}(\theta_i)\mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\mathbf{y}) \\ &= \mathbf{J}_{\mathbf{y}}(\theta_i) \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix} \lfloor \mathbf{e}_z \times \rfloor \mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\mathbf{f}(\mathbf{x}, \mathbf{u}_j))\end{aligned} \tag{B.6}$$

The Jacobian of the four-quadrant arctangent function is found by simply taking the partial derivatives with respect to the elements of $\mathbf{y}$.

$$\mathbf{J}_{\mathbf{y}}(\theta_i) = \begin{bmatrix}\dfrac{-v}{u^2+v^2} & \dfrac{u}{u^2+v^2}\end{bmatrix} \tag{B.7}$$

The inner function $\mathbf{f}(\mathbf{x}, \mathbf{u}_j)$ is simply the expression that rotates the observed line $\boldsymbol{l}_{j\parallel}$ into world coordinates. The Jacobian of this function is computed by forming the residual expression for $\mathbf{f}(\mathbf{x}, \mathbf{u}_j)$ and linearizing it around the current estimate.

$$\begin{aligned}\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{u}_j) &= \mathbf{f}(\mathbf{x}, \mathbf{u}_j) - \mathbf{f}(\hat{\mathbf{x}}, \hat{\mathbf{u}}_j) \\ &\approx \mathbf{H}_{\theta_i B}\tilde{\mathbf{x}}_B + \mathbf{H}_{\theta_i L}\tilde{\mathbf{x}}_L + \boldsymbol{\Gamma}_{\theta_i}\tilde{\mathbf{u}}_j\end{aligned} \tag{B.8}$$

We begin the derivation for the above Jacobians by substituting the definition of $\mathbf{f}(\mathbf{x}, \mathbf{u}_j)$ into the Eq. B.8.

$$\begin{aligned}\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{u}_j) = {}^{B}_{G}\mathbf{R}^T(t+t_n){}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\|} \\ - \mathbf{f}(\hat{\mathbf{x}}, \hat{\mathbf{u}}_j)\end{aligned} \tag{B.9}$$

The true rotation ${}^{B}_{G}\mathbf{R}(t+t_n)$ is linearized about the current state estimate using the expression from Eq. A.7 and the resulting linear approximation is substituted into the above expression.

$$\begin{aligned}\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{u}_j) = {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n){}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\|} \\ + \lfloor \tilde{\boldsymbol{\theta}}^G(t+\hat{t}_n)\times \rfloor {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n){}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\|} \\ + {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n)\lfloor {}^{B}\hat{\boldsymbol{\omega}}(t+\hat{t}_n)\times \rfloor {}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\|}\tilde{t}_n \\ - f(\hat{\mathbf{x}}, \hat{\mathbf{u}}_j)\end{aligned} \tag{B.10}$$

Next we expand the true rotation between the laser and IMU frames using the small angle approximation from Eq. A.9 and substituting it into Eq. B.10.

$$\begin{aligned}\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{u}_j) = {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n){}^{B}_{L}\hat{\mathbf{R}}\,\boldsymbol{l}_{j\|} \\ - {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n){}^{B}_{L}\hat{\mathbf{R}}\lfloor \tilde{\boldsymbol{\theta}}^L \times \rfloor\, \boldsymbol{l}_{j\|} \\ + \lfloor \tilde{\boldsymbol{\theta}}^G(t+\hat{t}_n)\times \rfloor {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n){}^{B}_{L}\hat{\mathbf{R}}\,\boldsymbol{l}_{j\|} \\ + {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n)\lfloor {}^{B}\hat{\boldsymbol{\omega}}(t+\hat{t}_n)\times \rfloor {}^{B}_{L}\hat{\mathbf{R}}\,\boldsymbol{l}_{j\|}\tilde{t}_n \\ - f(\hat{\mathbf{x}}, \hat{\mathbf{u}}_j)\end{aligned} \tag{B.11}$$

The final substitution that must be made is to expand the line $\boldsymbol{l}_{j\|}$ using the linear approximation from A.11.

$$\begin{aligned}\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{u}_j) = {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n){}^{B}_{L}\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\|} \\ - {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n){}^{B}_{L}\hat{\mathbf{R}}\lfloor\, \hat{\boldsymbol{l}}_{j\|}\times \rfloor \mathbf{e}_z\tilde{\phi}_j \\ - {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n){}^{B}_{L}\hat{\mathbf{R}}\lfloor \tilde{\boldsymbol{\theta}}^L \times \rfloor\, \hat{\boldsymbol{l}}_{j\|} \\ + \lfloor \tilde{\boldsymbol{\theta}}^G(t+\hat{t}_n)\times \rfloor {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n){}^{B}_{L}\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\|} \\ + {}^{B}_{G}\hat{\mathbf{R}}^T(t+t_n)\lfloor {}^{B}\hat{\boldsymbol{\omega}}(t+\hat{t}_n)\times \rfloor {}^{B}_{L}\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\|}\tilde{t}_n \\ - f(\hat{\mathbf{x}}, \hat{\mathbf{u}}_j)\end{aligned} \tag{B.12}$$

Inspecting the previous expression, we note that the first line is exactly the definition of $\mathbf{f}(\hat{\mathbf{x}}, \hat{\mathbf{u}}_j)$ and thus is canceled from the equation. We are left with an expression that is linear in the error state and only requires the current state estimate to evaluate. The required Jacobians can be trivially deduced by visual inspection.

$$\begin{aligned}\mathbf{H}_{\theta_i B} &= \begin{bmatrix}\lfloor -{}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n){}^{B}_{L}\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\parallel}\times\rfloor & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\end{bmatrix} \\ \mathbf{H}_{\theta_i L} &= \begin{bmatrix}{}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n){}^{B}_{L}\hat{\mathbf{R}}\lfloor\,\hat{\boldsymbol{l}}_{j\parallel}\times\rfloor & \mathbf{0}_{3\times3} & {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)\mathbf{M}_j & \frac{n}{N}{}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)\mathbf{M}_j\end{bmatrix} \\ \boldsymbol{\Gamma}_{\theta_i} &= \begin{bmatrix}\mathbf{0}_{3\times1} & -{}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n){}^{B}_{L}\hat{\mathbf{R}}\lfloor\,\hat{\boldsymbol{l}}_{j\parallel}\times\rfloor\mathbf{e}_z\end{bmatrix}\end{aligned} \tag{B.13}$$

The matrix $\mathbf{M}_j$ in the above expression is identical to that of Eq. A.19. The resulting Jacobians are then combined with Jacobian from Eq. B.7 to assemble the total Jacobian $\mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\theta_i)$ needed for covariance augmentation via the chain rule expression of Eq. B.6.

B.2 Plane Offset Jacobians

The offset parameter d_i describes the orthogonal distance between the origin and a plane $\boldsymbol{\Pi}_i$. We compute d_i from the state vector and laser line measurements by rearranging the distance constraint $\mathbf{z}_{l_{2j}}$ from Eq. A.22 and isolate the offset parameter to the left-hand side of the equation.

$$d_i = \boldsymbol{\pi}_i^T\big({}^{G}\mathbf{p}_B(t+t_n) + {}^{B}_{G}\mathbf{R}^T(t+t_n)({}^{B}\mathbf{p}_L + \rho_j {}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\perp})\big) \tag{B.14}$$

Unfortunately the expression for d_i is not only a function of the state vector and line measurement parameters, but also a function of the plane's normal angle θ_i. This means that the Jacobian expressions must take into account the uncertainty in normal angle and thus are slightly more complicated to derive. We begin by noting that Eq. B.14 can be factored into a function of the normal angle multiplied by a function of the state and input parameters.

$$\begin{aligned} d_i &= \boldsymbol{\pi}_i^T\big({}^{G}\mathbf{p}_B(t+t_n) + {}^{B}_{G}\mathbf{R}^T(t+t_n)({}^{B}\mathbf{p}_L + \rho_j {}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\perp})\big) \\ &= \mathbf{f}(\theta_i)^T\mathbf{g}(\mathbf{x},\mathbf{u}_j)\end{aligned} \tag{B.15}$$

Next, we evaluate Eq. B.14 using the current state estimate and observed line features to derive the estimate for the plane offset $\hat{d}_i$.

$$\hat{d}_i = \hat{\boldsymbol{\pi}}_i^T\big({}^{G}\hat{\mathbf{p}}_B(t+\hat{t}_n) + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\perp})\big) \tag{B.16}$$

Then we create the error state form of the offset equation by subtracting the estimate $\hat{d}_i$ from the true value d_i and apply first-order Taylor approximation to obtain the required linearized form.

$$\begin{aligned}\tilde{d}_i &= d_i - \hat{d}_i \\ &= \mathbf{f}(\theta_i)^T\mathbf{g}(\mathbf{x},\mathbf{u}_j) - \mathbf{f}(\hat{\theta}_i)^T\mathbf{g}(\hat{\mathbf{x}},\hat{\mathbf{u}}_j) \\ &\approx \mathbf{J}_{\mathbf{x},\mathbf{u}_j}\big(\mathbf{f}(\theta_i)^T\mathbf{g}(\mathbf{x},\mathbf{u}_j)\big)\big|_{\mathbf{x}=\hat{\mathbf{x}},\mathbf{u}_j=\hat{\mathbf{u}}_j}\begin{bmatrix}\tilde{\mathbf{x}}\\ \tilde{\mathbf{u}}_j\end{bmatrix}\end{aligned} \tag{B.17}$$

The Jacobian $\mathbf{J}_{\mathbf{x},\mathbf{u}_j}\big(\mathbf{f}(\theta_i)^T\mathbf{g}(\mathbf{x},\mathbf{u}_j)\big)$ is computed by applying both the multi-dimensional product and chain rules to the expression from Eq. B.15.

$$\mathbf{J}_{\mathbf{x},\mathbf{u}_j}\big(\mathbf{f}(\theta_i)^T\mathbf{g}(\mathbf{x},\mathbf{u}_j)\big) = \mathbf{g}^T(\mathbf{x},\mathbf{u}_j)\mathbf{J}_{\theta_i}\big(\mathbf{f}(\theta_i)\big)\mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\theta_i) + \mathbf{f}^T(\theta_i)\mathbf{J}_{\mathbf{x},\mathbf{u}_j}\big(\mathbf{g}(\mathbf{x},\mathbf{u}_j)\big) \tag{B.18}$$

Note that for horizontal planes $\mathbf{f}$ is constant and thus the first term evaluates to zero. Equation B.18 requires three separate Jacobians to be computed. The first, $\mathbf{J}_{\mathbf{x},\mathbf{u}_j}(\theta_i)$ is the Jacobian of the normal angle estimate with respect to state vector and input parameters. This is exactly Jacobian that we derived in Sec. B.1. The second Jacobian, $\mathbf{J}_{\theta_i}\big(\mathbf{f}(\theta_i)\big)$, is the derivative of the normal vector with respect to its angle.

$$\mathbf{J}_{\theta_i}\big(\mathbf{f}(\theta_i)\big) = \begin{bmatrix} -\sin(\theta_i) \\ \cos(\theta_i) \\ 0 \end{bmatrix} \tag{B.19}$$

The final Jacobian needed $\mathbf{J}_{\mathbf{x},\mathbf{u}_j}\big(\mathbf{g}(\mathbf{x},\mathbf{u}_j)\big)$ is computed by creating the linearized error form of $\mathbf{g}(\mathbf{x},\mathbf{u}_j)$.

$$\begin{aligned} \tilde{\mathbf{g}}(\mathbf{x},\mathbf{u}_j) &= \mathbf{g}(\mathbf{x},\mathbf{u}_j) - \hat{\mathbf{g}}(\mathbf{x},\mathbf{u}_j) \\ &\approx \mathbf{H}_{\mathbf{g}B}\tilde{\mathbf{x}}_B + \mathbf{H}_{\mathbf{g}L}\tilde{\mathbf{x}}_L + \mathbf{\Gamma}_{\mathbf{g}j}\tilde{\mathbf{u}}_j \end{aligned} \tag{B.20}$$

We begin the derivation begin by substituting the equation for $\mathbf{g}(\mathbf{x},\mathbf{u}_j)$ into the above equation.

$$\begin{aligned} \tilde{\mathbf{g}}(\mathbf{x},\mathbf{u}_j) = {}& {}^G\mathbf{p}_B(t+t_n) + {}^B_G\mathbf{R}^T(t+t_n)({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\,\boldsymbol{l}_{j\perp}) \\ & - \hat{\mathbf{g}}(\mathbf{x},\mathbf{u}_j) \end{aligned} \tag{B.21}$$

Next, we apply the approximation of ${}^G\mathbf{p}_B(t+t_n)$ from Eq. A.27 and Eq. A.28 and substitute it into the above equation.

$$\begin{aligned} \tilde{\mathbf{g}}(\mathbf{x},\mathbf{u}_j) = {}& {}^G\hat{\mathbf{p}}_B(t+\hat{t}_n) + {}^B_G\mathbf{R}^T(t+t_n)({}^B\mathbf{p}_L + \rho_j {}^B_L\mathbf{R}\,\boldsymbol{l}_{j\perp}) \\ & + {}^G\tilde{\mathbf{p}}_B(t) + \hat{t}_n {}^G\tilde{\mathbf{v}}_B(t) \\ & + {}^G\hat{\mathbf{v}}_B(t+\hat{t}_n)\tilde{t}_n \\ & - \hat{\mathbf{g}}(\mathbf{x},\mathbf{u}_j) \end{aligned} \tag{B.22}$$

Then the rotation is expanded using the small angle approximation and first order approximation from Eq. A.7 to eliminate the true rotation from the above equation.

$$\begin{aligned}\tilde{\mathbf{g}}(\mathbf{x},\mathbf{u}_j) = &{}^{G}\hat{\mathbf{p}}_B(t+\hat{t}_n) + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\mathbf{p}_L + \rho_j {}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\perp}) \\ &+ {}^{G}\tilde{\mathbf{p}}_B(t) + \hat{t}_n {}^{G}\tilde{\mathbf{v}}_B(t) \\ &+ \lfloor \tilde{\boldsymbol{\theta}}^{G}(t) \times \rfloor {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\mathbf{p}_L + \rho_j {}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\perp}) \\ &+ \big({}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)\lfloor {}^{B}\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor ({}^{B}\mathbf{p}_L + \rho_j {}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\perp}) + {}^{G}\hat{\mathbf{v}}_B(t+\hat{t}_n)\big)\tilde{t}_n \\ &- \hat{\mathbf{g}}(\mathbf{x},\mathbf{u}_j)\end{aligned} \tag{B.23}$$

The remaining additive parameters ρ_j and ${}^{B}\mathbf{p}_L$ are eliminated using the standard additive error definition and omitting any higher order terms.

$$\begin{aligned}\tilde{\mathbf{g}}(\mathbf{x},\mathbf{u}_j) = &{}^{G}\hat{\mathbf{p}}_B(t+\hat{t}_n) + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\perp}) \\ &+ {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\perp}\tilde{\rho}_j \\ &+ {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}\tilde{\mathbf{p}}_L \\ &+ {}^{G}\tilde{\mathbf{p}}_B(t) + \hat{t}_n {}^{G}\tilde{\mathbf{v}}_B(t) \\ &+ \lfloor \tilde{\boldsymbol{\theta}}^{G}(t) \times \rfloor {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\perp}) \\ &+ \big({}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)\lfloor {}^{B}\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor ({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\mathbf{R}\,\boldsymbol{l}_{j\perp}) + {}^{G}\hat{\mathbf{v}}_B(t+\hat{t}_n)\big)\tilde{t}_n \\ &- \hat{\mathbf{g}}(\mathbf{x},\mathbf{u}_j)\end{aligned} \tag{B.24}$$

We then expand the true rotation between the laser and IMU body coordinate frames using the approximation from Eq. A.9 and substitute it back into Eq. B.24.

$$\begin{aligned}\tilde{\mathbf{g}}(\mathbf{x},\mathbf{u}_j) = &{}^{G}\hat{\mathbf{p}}_B(t+\hat{t}_n) + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\,\boldsymbol{l}_{j\perp}) \\ &- {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)\hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\lfloor \tilde{\boldsymbol{\theta}}^{L} \times \rfloor\, \boldsymbol{l}_{j\perp} \\ &+ {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}_{L}\hat{\mathbf{R}}\,\boldsymbol{l}_{j\perp}\tilde{\rho}_j \\ &+ {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}\tilde{\mathbf{p}}_L \\ &+ {}^{G}\tilde{\mathbf{p}}_B(t) + \hat{t}_n {}^{G}\tilde{\mathbf{v}}_B(t) \\ &+ \lfloor \tilde{\boldsymbol{\theta}}^{G}(t) \times \rfloor {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\,\boldsymbol{l}_{j\perp}) \\ &+ \big({}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)\lfloor {}^{B}\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times \rfloor ({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\,\boldsymbol{l}_{j\perp}) + {}^{G}\hat{\mathbf{v}}_B(t+\hat{t}_n)\big)\tilde{t}_n \\ &- \hat{\mathbf{g}}(\mathbf{x},\mathbf{u}_j)\end{aligned} \tag{B.25}$$

The final substitution that must be made is to eliminate the true line $\boldsymbol{l}_{j\perp}$ and replace it via the Taylor approximation of Eq. A.34.

$$
\begin{aligned}
\tilde{\mathbf{g}}(\mathbf{x}, \mathbf{u}_j) = & {}^{G}\hat{\mathbf{p}}_B(t+\hat{t}_n) + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\perp}) \\
& - {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)\hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\lfloor\, \boldsymbol{l}_{j\perp} \times\rfloor \mathbf{e}_z \tilde{\phi}_j \\
& - {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)\hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\lfloor \tilde{\boldsymbol{\theta}}^{L} \times\rfloor\, \hat{\boldsymbol{l}}_{j\perp} \\
& + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}_{L}\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\perp}\tilde{\rho}_j \\
& + {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}\tilde{\mathbf{p}}_L \\
& + {}^{\tilde{G}}\mathbf{p}_B(t) + \hat{t}_n {}^{G}\tilde{\mathbf{v}}_B(t) \\
& + \lfloor \tilde{\boldsymbol{\theta}}^{G}(t) \times\rfloor {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\perp}) \\
& + \big({}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)\lfloor {}^{B}\hat{\boldsymbol{\omega}}(t+\hat{t}_n) \times\rfloor ({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\perp}) + {}^{G}\hat{\mathbf{v}}_B(t+\hat{t}_n)\big)\tilde{t}_n \\
& - \hat{\mathbf{g}}(\mathbf{x}, \mathbf{u}_j)
\end{aligned}
\tag{B.26}
$$

The first line is now exactly equation to $\hat{\mathbf{g}}(\mathbf{x}, \mathbf{u}_j)$ and is canceled from the expression. The remaining terms are now in the correct form to form the required matrices by visual inspection.

$$
\begin{aligned}
\mathbf{H}_{\mathbf{g}B} &= \begin{bmatrix} -\lfloor {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)({}^{B}\hat{\mathbf{p}}_L + \hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\,\hat{\boldsymbol{l}}_{j\perp}) \times\rfloor & \mathbf{I}_{3\times3} & \hat{t}_n\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \\
\mathbf{H}_{\mathbf{g}L} &= \begin{bmatrix} {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n)\hat{\rho}_j {}^{B}_{L}\hat{\mathbf{R}}\lfloor \hat{\boldsymbol{l}}_{j\perp} \times\rfloor & {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) & \mathbf{S}_j & \frac{n}{N}\mathbf{S}_j \end{bmatrix} \\
\boldsymbol{\Gamma}_{j\mathbf{g}} &= {}^{B}_{G}\hat{\mathbf{R}}^T(t+\hat{t}_n) {}^{B}_{L}\hat{\mathbf{R}} \begin{bmatrix} \hat{\boldsymbol{l}}_{j\perp} & -\hat{\rho}_j\lfloor \hat{\boldsymbol{l}}_{j\perp} \times\rfloor \mathbf{e}_z \end{bmatrix}
\end{aligned}
\tag{B.27}
$$

We note that the above Jacobians are exactly those needed to evaluate Equation 4.107. The overall Jacobians are finally constructed via Eq. B.18.