CONFIDENTIAL

Multi-Sensor Fusion of IMU, LIDAR and Wheel Encoders

Towards Tightly-Coupled Odometry

Koushik Kumaran

Master of Science Thesis

Delft Center for Systems and Control

Multi-Sensor Fusion of IMU, LIDAR and Wheel Encoders

Towards Tightly-Coupled Odometry

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Koushik Kumaran

August 16, 2023

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by Loop Robots BV. Their cooperation is hereby gratefully acknowledged.

Abstract

Loop Robots develops and operates the next generation of fully autonomous disinfection robots in hospitals and healthcare settings. Accurate localization is essential in order to navigate reliably and effectively disinfect the tight hallways and corners of a patient room, operating room, or intensive care unit in a hospital. Odometry, or dead-reckoning, is the process of estimating the robots pose with respect to a known initial pose. It forms the backbone of the robots localization strategy, as well as being critical to many other processes that run on the robot, such as control and mapping.

The current strategy of generating odometry using the wheel encoders is prone to drift due to the integration of errors into the estimate. Moreover, the estimation takes place on the horizontal plane due to the nature of the wheel encoders. To improve the odometry and extend it to the full 3D space, a solution that makes use of all of the onboard sensors in a tightly-coupled manner is required.

In this Thesis, we make the first steps towards this larger goal. The contributions of this thesis include: (1) an Extended Kalman Filter (EKF) algorithm to fuse data from the Inertial Measurement Unit (IMU) and wheel encoders to estimate position and orientation of the robot in 6-DoF; (2) a line feature extraction and tracking methodology to extract primitives from LIDAR data and (3) A Moving Horizon Estimation (MHE) scheme based on a factor-graph formulation to perform pose estimation on the horizontal plane using LIDAR data and wheel encoders.

We test the three modules individually using a combination of simulations and real-world data wherever possible. We found that the MHE scheme was able to reduce drift over the long term, but is sensitive to the effects of outliers in feature matching, motion distortion of LIDAR scans, and wheel slip. The EKF scheme is able to reduce the overall drift and correct for wheel slips.

Based on these results, promising avenues for the improvement of all the proposed modules are given, along with recommendations on how to combine them all in a tightly-coupled fashion.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor Per Slycke for his assistance during the writing of this thesis, along with the entire team at Loop Robots for their support. I also would like to thank my academic supervisors, Dr. Manuel Mazo and Dr. Manon Kok for their invaluable advice and guidance.

Last but not least I would like to thank my friends, family and colleagues, without whom this would not have been possible.

Delft, University of Technology
August 16, 2023

Koushik Kumaran

Chapter 1

Introduction

1-1 Motivation

Loop Robots develops the next generation of fully autonomous robots for UV based disinfection of rooms in hospital and healthcare settings, named SAM-UVC(See Figure 1-1). These robots operate in unknown environments such as patient rooms, operating rooms, or intensive care unit in hospitals, autonomously disinfecting the entire space using UV-C light. These spaces are often cluttered and have very narrow pathways and tight corners for the robot to map, navigate and disinfect.

For the disinfection of such rooms to be completely successful, it is critical that SAM-UVC covers the entire area to be disinfected and keep track of all the areas that have been disinfected. Accurate and reliable localization and mapping is a key ingredient for the successful operation of the SAM-UVC robot in the field.

Simultaneous Localization and Mapping (SLAM) is the process which involves the joint estimation of a map of the environment and the pose of the robot with respect to said map. For a robot to operate autonomously in an unfamiliar environment, precise and reliable SLAM is essential. Over the past decade, both the academic and industrial research communities have directed significant efforts to advance this field, striving for improved accuracy and robustness[1].

A closely related problem to SLAM is the *odometry*, or *dead-reckoning* problem, where the pose of the robot is estimated with respect to a known initial pose rather than the map. Most SLAM algorithms use a source of odometry as input, or front-end [1]. The accuracy and robustness of the odometry onboard is critical for the performance of the SLAM system and by extension, the entire disinfection process.

SAM-UVC relies on odometry generated at a rate of 50 Hz as the front-end of its SLAM system. Moreover, even when the SLAM system is disabled, the motion estimates from the onboard odometry are critical to a number of other processes, such as control, planning, collision detection, etc. Consequently, it is of utmost importance that odometry is generated

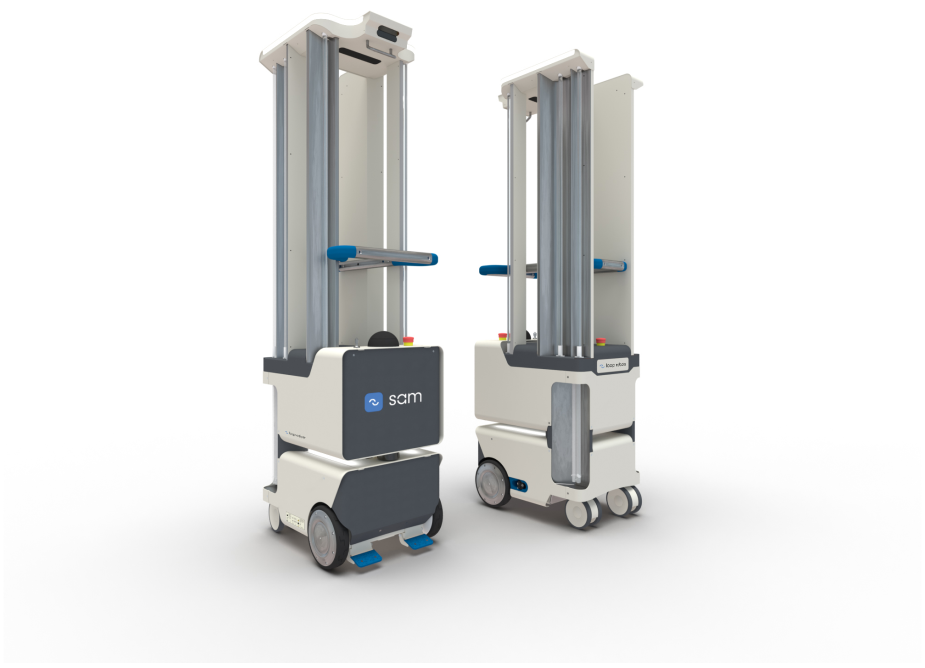


Figure 1-1: SAM-UVC Disinfection Robot

in real-time at a high frequency to ensure accurate and timely feedback for these critical functionalities.

Presently, SAM-UVC estimates odometry by integrating measurements from the wheel encoders, which measure the angular position of each wheel. This estimation method is susceptible to drift because it involves integrating errors that can arise from various sources, such as wheel slip, discretization, linearization errors, and more. As a result, the odometry estimate may deviate from the true position of the robot over time, compromising the accuracy of the robot's localization. Moreover, odometry generated using only the wheel encoders does not provide information about motion outside the plane that occurs when the robot drives over small bumps, ramps etc.

The wider research community has made many strides in improving the odometry estimation process, by incorporating data from other sensors, such as Inertial Measurement Unit (IMU)s, Laser Imaging Detection and Ranging (LIDAR) scanners, cameras, and so on [2] [3]. These methods have come a long way in the last decade or so, resulting in very robust and accurate systems that have succeeded in reducing the overall drift over long time periods by several orders of magnitude.

These methods can be roughly classified into *loosely-coupled* and *tightly-coupled*. In a loosely-coupled scheme, data from each sensor is processed separately and fused together to produce odometry. In practice, loosely-coupled estimation schemes are easier to develop and iterate upon, and are widely used.

However, tightly-coupled approaches have the advantage of being able to utilize the complementary nature of multiple sensors. These approaches have been gaining more traction in recent years, and show promising results with respect to their robustness and accuracy[4] [5]

[6] [7] [8] [9]. By nature, these are not easily generalizable, and must be meticulously designed for each robot and its sensor suite.

The ultimate objective of this thesis is to design a tightly-coupled odometry solution, a goal that lies beyond the scope of this study; nevertheless, we have taken multiple significant steps toward its realization.

1-2 Problem Statement and Key Challenges

The overall design goal of this project is summarized as:

How can we fuse information from LIDAR, IMU and wheel encoders to generate odometry with minimal drift while still being feasible for real-time computation?

This thesis was preceded by a literature study and internship by the author at Loop Robots, where some of the key challenges and questions to be addressed were identified, along with approaches to solve. In this work, we make a step towards developing a complete odometry solution by answering the following sub-questions.

How does one exploit the LIDAR scanner to reduce drift?

Of the three sensory inputs that we consider in this work, only the LIDAR provides measurements relating the external environment to the state. A common trend that was identified in the literature was the exploitation of the structure of indoor environments to track line or plane features in the environment [6] [10], thus minimizing drift. We take inspiration from these works to develop a feature extraction and tracking scheme that extracts line primitives from the LIDAR scans. We then implement a Moving Horizon Estimation (MHE) scheme to fuse the extracted line measurements with the wheel encoder data to provide 2D pose estimates using a factor graph formulation [11].

How does one reconcile the 2D and 3D nature of the different sensors?

Only the IMU provides information about the 6-DoF motion of the robot. The LIDAR and the wheel encoders only provide information regarding movement in the instantaneous plane of motion. To solve this problem, we design an Extended Kalman Filter (EKF) that is capable of fusing velocity measurements from the wheel encoders to estimate the pose and twist of the robot along with IMU biases.

How can the discussed solutions be implemented efficiently in real-time?

To reduce the scope of this thesis, we elected to partially ignore this problem. Only the EKF was implemented on the onboard computer of the robot. However, the 2D pose estimation is also expected to be feasible for a real-time implementation. This expectation stems from successful implementations of similar work in real-time systems, and the existence of efficient solvers for the type of optimization problem we solve to generate 2D pose estimates.

1-3 Summary of Contributions

In this work, we draw on the existing body of literature to design sensor fusion schemes for the SAM-UVC robot. This thesis project follows from a prior internship and literature survey by the author in collaboration with Loop Robots, where some of these ideas started to take shape. The contributions of this thesis include:

1. A line feature extraction and tracking methodology adapted from [12], [13] and [2].
2. A method to fuse the line features and Wheel Encoder measurements to generate 2D pose estimates at the LIDAR rate by optimizing a factor graph within a moving window.
3. An Extended Kalman Filter (EKF) for fusing Inertial Measurements with 2D velocity measurements to provide estimates at the IMU rate.
4. Experimental analysis of the proposed designs with a mix of simulated and real-world experiments.

1-3-1 Thesis Outline

With the key design questions and contributions defined, the remainder of this report is structured as follows.

Chapter 2 discusses the notations and conventions used in this text, along with the relevant models of the sensors and robot motion. It also briefly discusses some related work from the literature. In Chapter 3, we elaborate on the line feature tracking and methodology. We then formulate a Maximum A-Posteriori (MAP) estimation problem that allows us to estimate the 2D pose of the robot at the LIDAR rate. In Chapter 4, we discuss the Extended Kalman Filter (EKF) that we use to estimate the pose and twist of the robot at the IMU rate Chapter 3. In Chapter 5 we analyze the performance of our algorithms by applying them on simulated and real datasets, highlighting the strengths and weaknesses of the proposed approaches. Finally, in Chapter 6, we summarize the findings of this thesis and propose ideas for future extensions and implementations of this work.

Chapter 2

Preliminaries

Before delving into the relevant estimation methodology, we first discuss the problem setting in more detail. This chapter familiarizes the reader with the notations and conventions used in this text, as well as the relevant models for the robot and its sensors

General Notation: These conventions apply throughout this document, unless specified otherwise. Small letters represent scalars, bold letters represent vectors. Capital letters represent matrices, unless stated otherwise. The superscript on the right denotes the time stamp of a time-varying quantity. The superscript on the top left denotes the coordinate frame in which a variable is expressed. Sets are denoted by calligraphic uppercase letters. The bottom right subscript is reserved for any relevant indices, or other descriptors. $\dot{x}$ denotes the time derivative of the variable x. $\hat{x}$ denotes the estimate of the variable x. $||x||_P^2$ denotes the weighted two-norm of the vector x, such that $||x||_P^2 = x^T P^{-1} x$.

2-1 Coordinate Frames

We define the following frames of reference that are relevant to the problem:

1. World Frame(W): This is the fixed frame in which the robot pose is estimated, and has its origin at the starting point of the the robot, with it's x-axis along the robots heading.
2. Body/Bot Frame(B): This frame has its origin at the center of rotation of the robot and points its x-axis in the direction that the robot is heading.
3. IMU(I): This frame is aligned exactly with the IMUs sensing axes.

For this thesis, we assume that the extrinsics related to the sensor positioning on the robot are all perfectly calibrated. This means that we know the exact rigid-body transformations between the Body, IMU and LIDAR, and we estimate the transformation from the World to Body Frame.

2-2 Pose and Twist Representations

The position of the robot in the world frame can be expressed as ${}^{\mathrm{W}}\mathbf{p} = \begin{pmatrix} x & y & z \end{pmatrix}^T$.

The orientation of a body in 3D space can be expressed with different parameterizations, such as euler angles, axis-angle representations, quaternions, rotation matrices, etc. Each come with their own advantages and disadvantages. In this work, we primarily employ euler angles. This choice is sometimes discouraged when estimating orientation, owing to the fact that euler angles have issues such as gimbal lock, etc [14]. However, as we expect to operate close to zero roll and pitch at all times, these issues are not relevant in this work. In this work, we use the euler angle parameterization as we expect to be quite close to zero roll and pitch angles at all times.

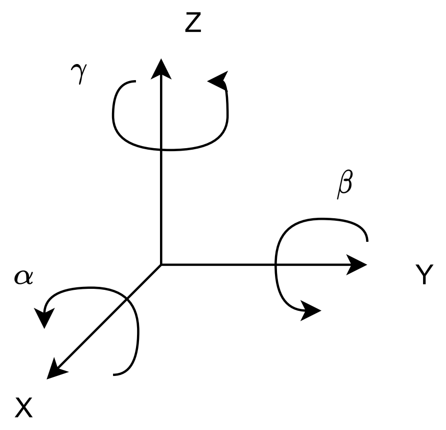


Figure 2-1: Visualization of the Roll, Pitch and Yaw around the Body Frame

The orientation of the robot body in the world frame is expressed by

$$ {}^{\mathrm{WB}}\boldsymbol{\theta} = \begin{pmatrix} \alpha & \beta & \gamma \end{pmatrix}^T \tag{2-1}$$

where α, β and γ are the roll, pitch and yaw angles respectively. The orientation of the robot can be obtained by the three subsequent rotations around body axes, in the order Z-Y-X. The orientation can be converted to a rotation matrix ${}^{WB}R$ [15] using

$$ {}^{\mathrm{WB}}R = \begin{bmatrix} \cos\gamma\cos\beta & \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos r & \cos\gamma\sin\beta\cos\alpha + \sin\gamma\sin\alpha \\ \sin\gamma\cos\beta & \sin\gamma\sin\beta\sin\alpha + \cos\gamma\cos r & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha \end{bmatrix} \tag{2-2}$$

This matrix can be used to convert a vector expressed in the body frame(${}^{\mathrm{B}}\mathbf{v}$) to the world frame(${}^{\mathrm{W}}\mathbf{v}$) using

$$ {}^{\mathrm{W}}\mathbf{v} = {}^{\mathrm{WB}}R \cdot {}^{\mathrm{B}}\mathbf{v} \tag{2-3}$$

We denote the linear velocity of the robot in the world frame as ${}^{W}\mathbf{v} = {}^{W}\dot{\mathbf{p}} = \begin{pmatrix} \dot{x} & \dot{y} & \dot{z} \end{pmatrix}^T$, and the angular velocity of the robot as seen in the body frame as ${}^{B}\boldsymbol{\omega}$. The velocity of the robot in the body frame can be obtained using (2-3).

Finally, we define the pose, linear and angular velocities of the robot on the horizontal plane as

$$\begin{aligned} {}^{\mathrm{W}}\boldsymbol{\xi} &= \begin{pmatrix} x & y & \gamma \end{pmatrix}^T \\ {}^{\mathrm{W}}\boldsymbol{\zeta} &= \begin{pmatrix} \dot{x} & \dot{y} \end{pmatrix}^T \\ {}^{\mathrm{B}}\omega &= \dot{\gamma} \end{aligned} \tag{2-4}$$

On the horizontal plane, we can use the same equation (2-3) to transform vectors between coordinate frames, with a different definition for the rotation matrix. With a slight abuse of notation, this is given by

$${}^{\mathrm{WB}}R = \begin{bmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{bmatrix} \tag{2-5}$$

Through the rest of this document, it should be clear from context which rotation matrix definition is used. Both of the rotation matrix definitions satisfy the properties,

$$\det({}^{\mathrm{WB}}R) = 1, \quad {}^{\mathrm{WB}}R^T \cdot {}^{\mathrm{WB}}R = {}^{\mathrm{WB}}R \cdot {}^{\mathrm{WB}}R^T = I, \quad {}^{\mathrm{WB}}R^{-1} = {}^{\mathrm{WB}}R^T$$

2-3 Motion Models

In this work, we use multiple different motion models to describe the motion of the robot. These models are tailored to specific use cases, such as 2D pose estimation and the implementation of the EKF. In this section, we describe these models.

The motion of SAM-UVC in 6-DoF can therefore be represented in continuous time as

$$\begin{aligned} {}^{\mathrm{WB}}\dot{R} &= {}^{\mathrm{WB}}R \cdot {}^{\mathrm{B}}\boldsymbol{\omega} \\ {}^{\mathrm{W}}\dot{\mathbf{v}} &= {}^{\mathrm{W}}\mathbf{a} \\ {}^{\mathrm{W}}\dot{\mathbf{p}} &= {}^{\mathrm{W}}\mathbf{v} \end{aligned} \tag{2-6}$$

This model is *unconstrained*, i.e, it ignores the holonomic constraints of the robots motion. This means that since SAM is a differential drive robot, it should ideally always have zero lateral velocity. This is addressed in the differential drive model

2-3-1 Differential Drive Model

A simpler model that makes use of the holonomic constraints is the differential drive model, which assumes that the robot operates on a plane, as shown in Figure 2-2. The corresponding motion model is given by

$$\begin{aligned} {}^{\mathrm{B}}\boldsymbol{\zeta} &= \begin{pmatrix} {}^{\mathrm{B}}v_x & 0 \end{pmatrix} \\ {}^{\mathrm{W}}\boldsymbol{\zeta} &= \begin{pmatrix} {}^{\mathrm{B}}v_x \cos\gamma & {}^{\mathrm{B}}v_x \sin\gamma \end{pmatrix}^T \\ \dot{\gamma} &= {}^{\mathrm{B}}\omega \end{aligned} \tag{2-7}$$

where ${}^{\mathrm{B}}v_x$ is the linear velocity. ${}^{\mathrm{B}}\omega$ is the angular velocity of the robot around its center of rotation.

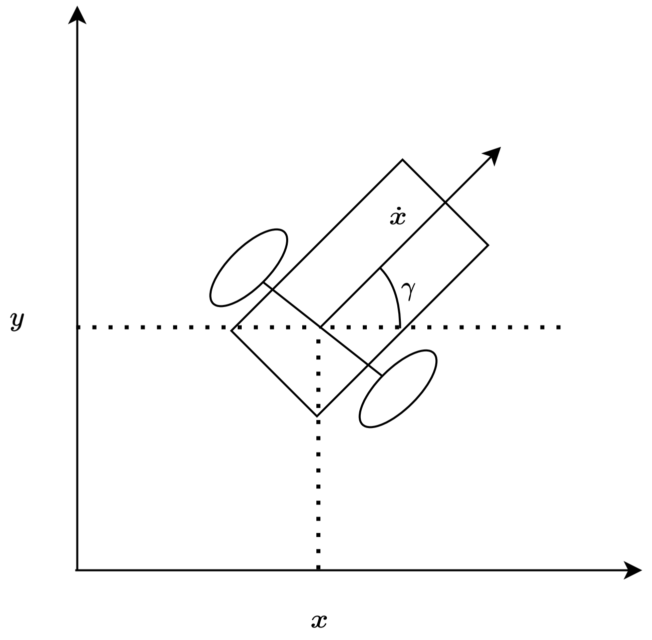


Figure 2-2: The Standard Differential Drive Robot Model

2-4 Sensors

SAM-UVC is equipped with four sensors, each measuring different quantities with respect to its motion and the environment. They can be classified as *proprioceptive* and *exteroceptive*. Proprioceptive sensors provide information about the motion, while exteroceptive sensors provide information about the environment and indirectly, motions.

Another way to classify them is by the type of information that is provided, 2D or 3D. The wheel encoders and LIDAR provide information in the current plane of operation of the robot, while the IMU and Camera provide information in all directions of motion. Table 2-1 presents the different sensors, and their differences. In this work, we disregard the presence of the camera.

Sensor	Measurements	Information Type	Rate
IMU	Linear Acceleration Angular Velocity	3D Proprioceptive	10-1000 Hz
Wheel Encoders	Linear Velocity Angular Velocity	2D Proprioceptive	50 Hz
LIDAR	2D Scan	2D Exteroceptive	10 Hz
RGB-D Camera	RGB-D Image	2D Exteroceptive	30 Hz

Table 2-1: A Comparison of the different sensors present on SAM-UVC

2-4-1 Wheel Encoders

Each of the robots wheels is equipped with a wheel encoder that measures the angular position of the wheel. Traditionally, odometry was calculated using only the wheel encoders by calculating the global position of the robot using the angular positions of the wheels. However, errors accumulate over time, causing large drifts in the estimate. It is therefore advisable to work with *relative* measurements, i.e, linear and angular velocities instead. These may be very slightly biased due to modeling inaccuracies but are not subject to drift.

If we denote the measured wheel rotations over an interval Δt by $\Delta\theta_l$ $\Delta\theta_r$, we can then compute the linear and angular velocity of SAM-UVC in the Body frame using

$$\begin{aligned} {}^{\mathrm{B}}v_x &= \frac{r(\omega_r + \omega_l)}{2} \\ {}^{\mathrm{B}}\omega &= \frac{r(\omega_r - \omega_l)}{2b} \end{aligned} \tag{2-8}$$

where r and b denote the known wheel radius and base length, respectively, and $\omega_l = \frac{\Delta\theta_l}{\Delta t}$, $\omega_r = \frac{\Delta\theta_r}{\Delta t}$.

The noise that occurs when using encoder measurements is often(also in this work) modelled as Gaussian noise, but in reality arises primarily from four sources:

1. Wheel slip: Integrating these measurements by using them in conjunction with (2-7) makes the implicit assumption that the wheels undergo pure rotation on the grounds, disregarding slip and wheel deformation. This is the main source of noise whenever wheel encoders are used.

2. Linearization: When using these measurements, a linearised and discretised version of (2-7) is typically employed. This introduces some error into any scheme that uses the measurements.

3. Modelling inaccuracies: The wheel radius and base length are subject to some error due to mechanical tolerances.

4. Quantization noise: The wheels absolute angle is measured with a accuracy of 8192 ticks per complete rotation. This translates to an angular resolution of 0.03° on $\Delta\theta_l$ and $\Delta\theta_r$

2-4-2 Inertial Measurement Unit (IMU)

SAM-UVC is equipped with an IIM-462340 Inertial Measurement Unit (IMU) from TDK Invensense. This is a 6-DoF sensor that measures and reports the linear accelerations and angular velocities of the sensor in its own frame of reference. These measurements are affected by a slowly varying bias that should be taken into account to mitigate drift in any estimation scheme. Slowly varying bias can be modeled as a random walk. The IMU measurements and bias evolution can be modeled as

$$
\begin{aligned}
{}^{\mathrm{B}}\mathbf{a}^t_{meas} &= \left({}^{\mathrm{WB}}R^t\right)^T \left({}^{\mathrm{W}}\mathbf{a}^t - {}^{\mathrm{W}}\mathbf{g}\right) + \mathbf{b}^t_{acc} + \boldsymbol{\eta}^t_{acc} && \text{(2-9a)}\\
{}^{B}\boldsymbol{\omega}^t_{meas} &= {}^{B}\boldsymbol{\omega}^t + \mathbf{b}^t_{gyro} + \boldsymbol{\eta}^t_{gyro} && \text{(2-9b)}\\
\dot{\mathbf{b}}^t_{acc} &= \boldsymbol{\epsilon}^t_{acc} && \text{(2-9c)}\\
\dot{\mathbf{b}}^t_{gyro} &= \boldsymbol{\epsilon}^t_{gyro} && \text{(2-9d)}
\end{aligned}
$$

,where ${}^{\mathrm{B}}\mathbf{a}^t_{meas} \in \mathbb{R}^3$ and ${}^{B}\boldsymbol{\omega}^t_{meas} \in \mathbb{R}^3$ denote the accelerometer and gyro measurements and $\boldsymbol{\eta}^t_{acc} \sim \mathcal{N}(0, Q_{\boldsymbol{\eta},acc})$, $\boldsymbol{\eta}^t_{gyro} \sim \mathcal{N}(0, Q_{\boldsymbol{\eta},gyro})$, $\boldsymbol{\epsilon}^t_{acc} \sim \mathcal{N}(0, Q_{\boldsymbol{\epsilon},acc})$ and $\boldsymbol{\epsilon}^t_{gyro} \sim \mathcal{N}(0, Q_{\boldsymbol{\epsilon},gyro})$ represent the noise terms on the measurements, and the bias random walks respectively.

Although the sensor measurements are very accurate, they cannot solely be relied upon due to the biases. Since these measurements need to be integrated to derive position and orientation, relying solely on inertial sensors for extended periods leads to unreliable estimates.

The sensor in question can provide measurements at a configurable Output Data Rate (ODR) from 10-1000 Hz. The measurements undergo a trapezoidal integration on board the device to provide data at the requested ODR.

2-4-3 LIDAR

SAM-UVC is equipped with a LS-LIDAR N401 2D-LIDAR scanner. The scanner operates at a rate of 10 Hz. At each sampling instant, the LIDAR scanner scans the environment in 2D. The scan at time t is given by a set of points, $\mathcal{S}^t := \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$, with each point expressed in polar coordinates as $\mathbf{p}_i = \begin{pmatrix} d_i & \phi_i \end{pmatrix}^T$.

These points can alternatively be expressed in cartesian coordinates as $\mathbf{p}_{i,c} = \begin{pmatrix} x_i & y_i \end{pmatrix}^T = \begin{pmatrix} d_i \cos\phi_i & d_i \sin\phi_i \end{pmatrix}^T$. We omit the subscript t for brevity henceforth, except in places where the notation may be ambiguous.

We follow the approach of [12], and model that both the range and the angle measurement are perturbed by Gaussian noise, distributed as $\epsilon_d \sim \mathcal{N}\left(0, \sigma_d^2\right)$ and $\epsilon_\phi \sim \mathcal{N}\left(0, \sigma_\phi^2\right)$

$$\begin{aligned} d_i &= D_i + \epsilon_d \\ \phi_i &= \Phi_i + \epsilon_\phi \end{aligned} \tag{2-10}$$

where (D_i, Φ_i) is true location of the point in polar coordinaes.

We can then express the measured point $\mathbf{p}_i$ in cartesian coordinates as $\mathbf{p}_{i,c}$ using the relation

$$\mathbf{p}_{i,c} = (D_i + \epsilon_d) \begin{bmatrix} \cos(\phi_i + \epsilon_\phi) \\ \sin(\phi_i + \epsilon_\phi) \end{bmatrix} \tag{2-11}$$

Since ϵ_ϕ is very close to zero in practice, we use the approximation $\epsilon_\phi \to 0$, to express the measurement of $\mathbf{p}_{i,c}$ in terms of the true point $\mathbf{P}_{i,c}$.

$${}^c p_i \approx {}^c P_i + \boldsymbol{\epsilon}_p \tag{2-12}$$

where the noise covariance $Q_{i,p}$ of the noise term $\boldsymbol{\epsilon}_p \sim \mathcal{N}(0, Q_{i,p})$, is given by

$$Q_{i,p} = \frac{D_i^2 \sigma_\phi^2}{2} \begin{bmatrix} 2\sin^2\Phi_i & -\sin 2\Phi_i \\ -\sin 2\Phi_i & 2\cos^2\Phi_i \end{bmatrix} + \frac{\sigma_d^2}{2} \begin{bmatrix} 2\cos^2\Phi_i & \sin 2\Phi_i \\ \sin 2\Phi_i & 2\sin^2\Phi_i \end{bmatrix} \tag{2-13}$$

In practice, we approximate the $Q_{i,p}$ matrix by using the measured range and angle.

2-4-4 RGB-D Camera

SAM is also equipped with a Realsens depth camera. This camera provides Red-Green-Blue-Depth (RGB-D) images at a rate of 30 Hz. This is similar to a standard 3-channel RGB image, with an extra channel that corresponds to the *depth* of the pixel, i.e, the distance of the point from the camera. The device can be configured to provide this information in a colored point-cloud form. In this sense, it acts like a 3D-LIDAR with a smaller field of view. For this thesis, we we disregard the information from the camera to contain the scope.

2-5 Related Work

For 3D-LIDAR scanners, the prevailing method of feature extraction is derived from the LOAM algorithm [2]. This algorithm begins by computing a smoothness metric for each individual measured point. The points exhibiting the lowest and highest smoothness values are categorized as plane and edge features, respectively.

The fitting of lines to these points can be done in many different ways. [12] provided a stochastic representation of line features that models the noise of the scanner in its natural polar coordinates. They used a Hough Transform [16] to simultaneously identify line feature points and group them into lines. In our work, we chose to adapt this algorithm due to its ability to provide uncertainty estimates along with the line estimates. Alternatively, there are some algorithms such as RANdom SAmple Consensus (RANSAC) [17], which are by design more robust.

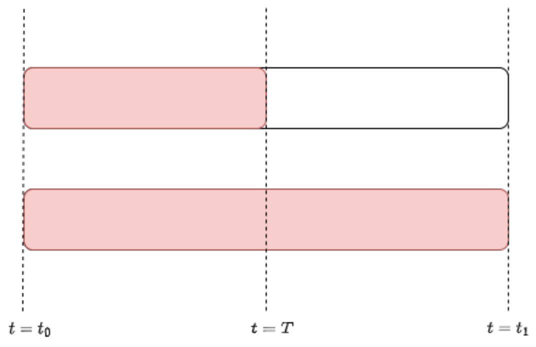


Figure 2-3: The Difference between Filtering(top) and Smoothing(bottom). The state is estimated at time T. The filtering scheme, uses the past measurements within the range $t \in [t_0, T]$ while the smoothing scheme(bottom), uses the past and future measurements within the range $t \in [t_0, t_1]$

For estimating odometry, there are two main approaches are popular in the literature. The first is the filter-based approach which takes into account only the previous estimate of the state when estimating the next state. Examples of this approach include most strategies based on Kalman filters. A popular methodology for Visual-Inertial Odometry (VIO) is based on the Multi State Constraint-Kalman Filter (MSC-KF). The original formulations of the SLAM problem were based on non-linear filtering techniques such as an EKF [18]. In [19], the authors improve the odometry of a wheeled rover, making use of zero-type updates to maintain the filter states from diverging. We have adopted a similar approach in our work.

Another approach is to use a *smoother*, which processes all the data together and provides an estimate of the entire trajectory. While these methods are more accurate, they come with the drawback of being computationally intensive. In the literature regarding computer vision a optimization based smoother is often used to refine trajectories after an initial estimate is generated using a filter, in a process known as *bundle-adjustment* [20]

A middle ground between filtering and smoothing is *fixed-lag smoothing*, in which the trajectory is optimized over a moving window. Figure 2-3 illustrates the difference between these methods. Fixed-lag smoothing is also known as Moving Horizon Estimation (MHE) in the sensor fusion literature. In fact, filtering schemes can be viewed as a special case of fixed-lag smoothers with a window size of 1. In [21], the authors design MHE schemes centered around the use of an IMU.

In [22], the authors investigate the differences between each strategy for VIO problems, and experimentally show that, using a smoother results in an estimator that is more robust and only slightly more accurate. In [8] [23] and [6], the authors propose a tightly-coupled odometry system for a legged robot using factor graphs, with the usage of plane and line features in an MHE scheme, further demonstrating the desirable properties of such systems. We hope that the work in this project can be extended to such a scheme in the future, see Chapter 6 for a more in-depth discussion.

Chapter 3

2D Pose Estimation

This Chapter is dedicated to discusssing the working of the 2D Pose Estimation based on the LIDAR and the wheel encoders. We begin by familiarizing the reader with the methodology used to extract and track line primitives from the scan data. We then formulate a Maximum A-Posteriori (MAP) problem that allows us to generate pose estimates from the LIDAR and wheel encoders in 2D at the LIDAR rate.

Figure 3-1 visualizes the different steps involved in processing the LIDAR scans to extract line primitives. We first remove all points in the scan that are not part of a line feature. The remaining points are then grouped together, and an initial line estimate is generated using a least-squares method. The initial estimate is then optimized using the weighted line fitting method, providing us with line features and their associated uncertainties. Finally, we match the features across multiple scans using a chi-squared test.

The method we implement to extract and track line primitives is adapted from [12] and [2]. We reproduce and explain the relevant results and equations from these texts in this chapter. The interested reader is directed to the original texts for more detailed derivations and proofs.

Simplifiying Assumption 1: In this chapter, we ignore the roll and pitch angles of the robot, i.e, $\alpha, \beta \approx 0$. This assumption is expected to introduce some error into the estimation prcoess. Proper modelling of the effect of these angles on the line feature estimation process is outside the scope of this work.

Simplifiying Assumption 2: We also ignore the effects of motion distortion on the LIDAR scans. The points in a single scan $\mathcal{S}^{t_k}$ are actually measured at different times between (t_{k-1}, t_k) due to the rotating nature of the scanner. The scanner accumulates these points over a single rotation and timestamps the entire set of points at the end of a rotation before transmitting them over the network. If the robot is in motion during a scan, it can cause the scan to be distorted, introducing errors into the estimation process if left unmodelled. Due to the slow moving nature of the robot, we chose to disregard this effect.

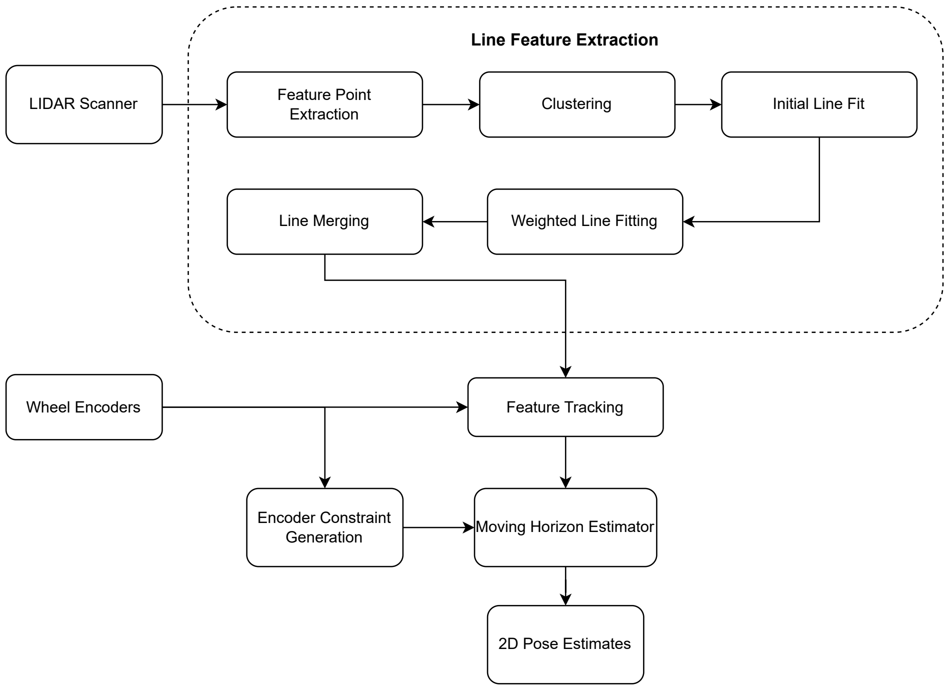


Figure 3-1: The proposed pipeline for pose estimation on the horizontal plane. It includes the feature extraction and feature tracking modules, followed by the MHE

3-1 Line Feature Extraction

3-1-1 Line Feature Model

We define a line feature as

$$^{\mathrm{B}}\mathbf{l} = \begin{pmatrix} r & \alpha & s \end{pmatrix}^T \tag{3-1}$$

with r as the normal distance to the line, α the angle of the normal to the line with respect to the robot frame. The s parameter denotes the distance weighted mean of the points from the origin, along the line. The superscript B denotes the reference frame of the measurement. The superscript is dropped wherever the coordinate frame is evident from context for brevity.

Therefore, the r and α values parameterize the infinite line, while s provides information about the measured *segment.*

As shown in Figure 3-2, we can define a reference frame L for each line, which is centered with the body frame, rotated with an angle α. The pair (r, s) denotes the coordinates of the center of the line segment in the L frame.

3-1-2 Identifying Line Feature Points

In this work, we adapt the feature point extraction method from [2] for use with a 2D scanner. We use this extraction method to extract candidate *line feature points* from each scan. By

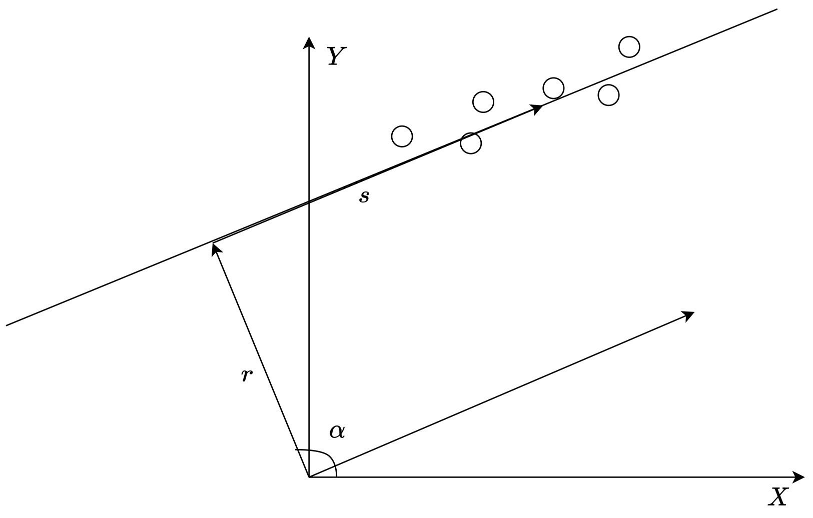


Figure 3-2: Geometry of a single line feature and its associated L frame

applying this method, we remove any points in the observed scan that are not part of a line feature.

For each point (d_i, ϕ_i), we calculate a smoothness coefficient c_i using,

$$c_i = \frac{1}{|\mathcal{S}_i| \cdot d_i} \left\| \sum_{j \in \mathcal{S}_i, j \neq i} \begin{pmatrix} d_i \cos\phi_i - d_j \cos\phi_j \\ d_i \sin\phi_i - d_j \sin\phi_j \end{pmatrix} \right\|_2 \tag{3-2}$$

where $\mathcal{S}_i := \{p_j | p_j \in \mathcal{S}\}, |\theta_j - \theta_i| \leq \theta_{max}\}$ and $|\mathcal{S}_i|$ denotes the cardinality of the set $\mathcal{S}_i$.

The set of candidate line feature points is given by the points a smoothness less than a given threshold, or $\mathcal{P}_t := \{p_i | c_i \leq c_{\text{thresh}}, p_i \in \mathcal{S}_t\}$.

Using this method to extract only the salient points in a scan reduces the total number of points to be clustered and fit by roughly 20% in our experiments, see Chapter 5.

3-1-3 Clustering

After extracting the line feature points, we separate them into clusters. We iterate over the points in the scan and classify points together based on their distance to the previous point as elaborated in Algorithm 1. At the end of this process, we are left with a set of clusters $\mathcal{G}$.

Finally, we discard any cluster with a small number of points, as these are likely to be outliers in the measurements.

3-1-4 Unweighted Initial Line Fitting

The weighted line fitting method that we use benefits from a good initial estimate of the line parameters. To obtain such an estimate, we convert the set of points $\mathcal{C} := \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, ..., \mathbf{p}_N\}$, to cartesian coordinates as $x_i = r_i \cos\theta_i$, $y_i = r_i \sin\theta_i$.

Algorithm 1 Grouping Points

```
Input: Set of Line Feature Points $\mathcal{P}$
Output: A Set of $M$ Clusters $\mathcal{G} := \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \dots \mathcal{C}_M\}$
    $j \leftarrow 1$
    $\mathcal{C}_1 \leftarrow \{p_1\}$
    $\mathcal{G} \leftarrow \{\mathcal{C}_1\}$
    for $p_i \in \mathcal{P}$ do
        if $||{}^c p_i - {}^c p_{i-1}||_2 > d_{min}$ then
            $j \leftarrow j + 1$
            Add new cluster $\mathcal{C}_j$ to $\mathcal{G}$
        end if
        Add ${}^c p_i$ to $\mathcal{C}_j$
    end for
```

We then fit these coordinates to a simplified slope-intercept line model given by

$$y_i = m x_i + c + \epsilon_l$$

where $\epsilon_l \sim \mathcal{N}(0, \sigma_l)$. Now, all the measurements can be grouped together in the form,

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} m \\ c \end{bmatrix}}_{\Lambda} + \epsilon_L \tag{3-3}$$

where $\epsilon_L \sim \mathcal{N}(0, \sigma_l I_N)$. This model is a crude approximation that gives us acceptable initial estimates for the line parameters in practice.

Assuming that each point measurement is independent, we can estimate the parameter vector Λ as $\hat{\Lambda}$ by minimizing the least-square error, giving us the estimator

$$\hat{\Lambda} = \left(X^T X\right)^{-1} X^T Y \tag{3-4}$$

This gives an initial estimate of the slope-intercept parameters of the infinite line, given by $\hat{m}, \hat{c}$ which can be used to arrive at an initial estimate $\hat{r}, \hat{\alpha}$ using the transformation

$$\begin{aligned} \hat{r} &= \frac{\hat{c}}{\sqrt{1 + \hat{m}^2}} \\ \hat{\alpha} &= \arctan \hat{m} + \frac{\pi}{2} \end{aligned} \tag{3-5}$$

We can now also arrive at an initial estimate for the s parameter by using,

$$\hat{s} = \sqrt{\bar{x}^2 + \bar{y}^2} \sin\left(\hat{\alpha} - \arctan\left(\frac{\bar{y}}{\bar{x}}\right)\right) \tag{3-6}$$

where $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$, $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$

3-1-5 Weighted Line Fitting

For each point $\mathbf{p}_i$ expressed in polar coordinates, we calculate the distance from the line as

$$\begin{aligned} \delta_{i,r} &= u_{i,r} - r = d_i \cos(\alpha - \phi_i) - r \\ \delta_{i,s} &= u_{i,s} - s = d_i \sin(\alpha - \phi_i) - s \end{aligned} \tag{3-7}$$

where $\mathbf{u}_i = \begin{pmatrix} u_{i,r} & u_{i,s} \end{pmatrix}^T$ are the coordinates of $\mathbf{p}_i$ expressed in the L frame. These errors can be considered uncorrelated due to the choice of reference frame. Therefore, only δ_r needs to be minimized to fit the infinite line parameters r and α, and only δ_s needs to be minimized to find the center, s.

The covariance of the scalar errors in the RS reference frame are given by

$$\begin{aligned} P_{i,r} &= Q_{11} \cos^2 \alpha + 2Q_{12} \sin\alpha \cos\alpha + Q_{22} \sin^2 \alpha \\ P_{i,s} &= Q_{11} \sin^2 \alpha - 2Q_{12} \sin\alpha \cos\alpha + Q_{22} \cos^2 \alpha \end{aligned} \tag{3-8}$$

where $Q \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}$ is the noise covariance as defined in (2-13).

We use a maximum likelihood formulation to estimate the line parameters. Assuming the measurements are all independent, the error distribution is given by

$$\begin{aligned} p(\delta|l) &= \prod_{i=1}^{N} p(\delta_i|l) \\ p(\delta|l) &= \prod_{i=1}^{N} \frac{e^{-\frac{1}{2}(\delta_i)^T (P_i)^{-1} \delta_i}}{2\pi\sqrt{\det P_i}} \end{aligned} \tag{3-9}$$

where we omit the subscripts r and s, as the same equation applies to both error functions. To find the optimal estimates for the line parameters, we minimize the negative log-likelihood function

$$\begin{aligned} l &= \arg\min_l - \log p(\delta|l) \\ l &= \arg\min_l \left[\sum_{i=1}^{N} \frac{1}{2} (\delta_i)^T (P_i)^{-1} \delta_i + \sum_{i=1}^{N} \ln \left(2\pi \sqrt{\det P_i} \right) \right] \end{aligned} \tag{3-10}$$

Since the P_i is a function of α, there is no exact closed form solution to this problem. Exploiting the fact if α is fixed, then r and s have a closed-form solution, we use an iterative scheme to optimize the line parameters. First, using the initial guess for the line orientation from Section 3-1-4 , we calculate the optimal r, s parameters using (3-11). We then incrementally update α using (3-13), and recalculate r, s. This procedure is repeated until convergence. Finally, we calculate the uncertainty of the estimates using (3-14)

The optimal estimate for r and s is given by

$$\begin{bmatrix} r \\ s \end{bmatrix} = \begin{bmatrix} P_{rr} & 0 \\ 0 & P_{ss} \end{bmatrix} \sum_{i=1}^{N} \left(\begin{bmatrix} P_{i,r} & 0 \\ 0 & P^{i,s} \end{bmatrix}^{-1} \begin{bmatrix} u_{i,r} \\ u_{i,s} \end{bmatrix} \right) \tag{3-11}$$

where P_{rr} and P_{ss} are given by

$$\begin{aligned} P_{rr} &= \frac{1}{\sum_{i=1}^{N} \frac{1}{P_{i,r}}} \\ P_{ss} &= \frac{1}{\sum_{i=1}^{N} \frac{1}{P_{i,s}}} \end{aligned} \tag{3-12}$$

The estimate α is updated as $\alpha \leftarrow \alpha + \delta\alpha$, where $\delta\alpha$ is given by

$$\delta\alpha = -\frac{\sum_{i=1}^{N} \left(\frac{\delta^{i,r}\delta^{i,s}}{P_{i,r}} \right)}{\sum_{i=1}^{N} \left(\frac{\delta_{i,s}\delta_{i,s}}{P_{i,r}} \right)} \tag{3-13}$$

The covariance of the line estimates are given by

$$P_{\mathbf{l}} = \begin{bmatrix} P_{rr} & P_{r\alpha} & 0 \\ P_{r\alpha} & P_{\alpha\alpha} & 0 \\ 0 & 0 & P_{ss} \end{bmatrix} \tag{3-14}$$

where

$$\begin{aligned} P_{\alpha\alpha} &= \frac{1}{\sum_{i=1}^{N} \left(\frac{\delta_{i,s}\delta_{i,s}}{P_{i,r}} \right)} \\ P_{r\alpha} &= -P_{rr}P_{\alpha\alpha} \sum_{i=1}^{N} \left(\frac{\delta_{i,s}}{P_{i,r}} \right) \end{aligned} \tag{3-15}$$

3-2 Line Merging and Tracking

In various scenarios, having a metric to assess the similarity between two distinct line measurements taken from the same line proves advantageous. This metric serves multiple purposes, such as facilitating feature tracking over time and enabling the consolidation of redundant measurements of a single line within a scan. Redundancy might arise due to obstacles obstructing the line, leading to its division into separate line segments.

To avoid redundant measurements, we merge multiple line segments that are pieces of the same infinite line together into a single line feature by comparing each pair of lines based on their squared Mahalanobis distance, and merge them if they are within a threshold determined from a one sided χ^2 distribution. Note that we compare only the r, α parameters that parameterize the infinite line

The squared mahalanobis distance between two line measurements $\mathbf{l}_1$ and $\mathbf{l}_2$, with covariances $P_{\mathbf{l}_1}$ and $P_{\mathbf{l}_2}$ respectively is given as

$$D^2 = \boldsymbol{\delta}_{\mathbf{l}}^T \left(P_{\mathbf{l}_1} + P_{\mathbf{l}_2} \right)^{-1} \boldsymbol{\delta}_{\mathbf{l}} \tag{3-16}$$

where

$$\boldsymbol{\delta}_{\mathbf{l}} = \mathbf{l}_1 \boxminus \mathbf{l}_2$$

and we define the $\boxminus$ operator between two lines as $\mathbf{l}_1 \boxminus \mathbf{l}_2 = \begin{pmatrix} r_1 - r_2 \\ \alpha_1 - \alpha_2 \\ 0 \end{pmatrix}$

If the distance is small enough, we merge the two lines to give $\mathbf{l}_m$ as

$$\begin{aligned} P_{\mathbf{l}_m} &= \left(P_{\mathbf{l}_1}^{-1} + P_{\mathbf{l}_2}^{-1}\right)^{-1} \\ \mathbf{l}_m &= P_{\mathbf{l}_m}^{-1}\left(P_{\mathbf{l}_1}^{-1}\mathbf{l}_1 + P_{\mathbf{l}_2}^{-1}\mathbf{l}_2\right) \end{aligned} \tag{3-17}$$

3-2-1 Feature Tracking

To use the measurements of the line features in a sensor fusion framework, we are required to track the line features across multiple scans. If the robot undergoes a motion between two time instants t_1 and t_2, and we have a transformation between them $\boldsymbol{\xi}^{t_1,t_2} = \begin{pmatrix}\delta x & \delta y & \delta\gamma\end{pmatrix}^T$, the line parameters ${}^{\mathrm{B}}\mathbf{l}^{t_1}$ and ${}^{\mathrm{B}}\mathbf{l}^{t_2}$ are related by

$$\begin{bmatrix} r^{t_1} \\ \alpha^{t_1} \\ s^{t_1} \end{bmatrix} = f_{\text{forward}}({}^{\mathrm{B}}\mathbf{l}^{t_2}, \boldsymbol{\xi}^{t_1,t_2}) = \begin{bmatrix} r^{t_2} + \delta x\cos(\alpha^{t_2}+\delta\gamma) + \delta y\sin(\alpha^{t_2}+\delta\gamma) \\ \alpha^{t_2} + \delta\gamma \\ s^{t_2} - \delta x\sin(\alpha^{t_2}+\delta\gamma) + \delta y\cos(\alpha^{t_2}+\delta\gamma) \end{bmatrix} \tag{3-18}$$

or alternatively,

$$\begin{bmatrix} r^{t_2} \\ \alpha^{t_2} \\ s^{t_2} \end{bmatrix} = f_{\text{backward}}({}^{\mathrm{B}}\mathbf{l}^{t_1}, \boldsymbol{\xi}^{t_1,t_2}) = \begin{bmatrix} r^{t_1} - \delta x\cos(\alpha^{t_1}) - \delta y\sin(\alpha^{t_1}) \\ \alpha^{t_1} - \delta\gamma \\ s^{t_1} + \delta x\sin(\alpha^{t_1}) - \delta y\cos(\alpha^{t_1}) \end{bmatrix} \tag{3-19}$$

The uncertainty of the line measurement can be propagated as

$$P^{t_2} = BP^{t_1}B^T + KQ^{t_1,t_2}K^T \tag{3-20}$$

where B and K are obtained by linearising (3-18) and Q^{t_1,t_2} is the uncertainty associated with the transformation $\boldsymbol{\xi}^{t_1,t_2}$.

$$\begin{aligned} B &= \begin{bmatrix} 1 & \delta x\sin\alpha_t - \delta y\cos\alpha_t & 0 \\ 0 & 1 & 0 \\ 0 & \delta x\cos\alpha_t + \delta y\sin\alpha_t & 1 \end{bmatrix} \\ K &= \begin{bmatrix} -\cos\alpha_t & -\sin\alpha_t & 0 \\ 0 & 0 & 1 \\ \sin\alpha_t & -\cos\alpha_t & 0 \end{bmatrix} \end{aligned} \tag{3-21}$$

We then match the lines using the same criterion as (3-16)

3-3 Fusing Line Features with Wheel Encoder Measurements

Our goal is now to fuse the obtained line measurements with the wheel encoder measurements. To do this, we formulate an optimization problem for a window of past states at every instant when a LIDAR measurement is received, which can then be solved by the use a non-linear least squares solver to estimate the robots 2D pose.

Non-linear least squares problems such as the one we formulate are common in robotic control and sensor fusion setups. Typically, these problems have a special sparse structure due to the nature of the problem. Factor graphs are a probabilistic model that take advantage of this sparse structure to represent the problem succinctly, opening the door to efficient incremental solvers [24] [25] and a host of other techniques to efficiently solve these problems.

A factor graph is formally defined as a bipartite graph $F = (\mathcal{U}, \mathcal{V}, \mathcal{E})$.

1. $\phi_i \in \mathcal{U}$ are nodes that represent *factors.*
2. $x_i \in \mathcal{V}$ are nodes that represent *variables.*
3. e_{ij} represent edges that connect factor nodes and variable nodes.

In our case, the nodes within the factor graph correspond to the 2D pose of the robot at each LIDAR time instance, while the factors represent the line and wheel encoder measurements. Although we do not extensively explore the theory behind factor graphs and the associated optimization in this work, it is worth highlighting that by structuring our problem in this manner, we pave the way for potential future implementations using established solvers and streamlined incremental techniques.

Note on Line Measurements: We drop the s measurements from the line estimates at this point, and redefine a line as ${}^{\mathrm{B}}\mathbf{l} = \begin{pmatrix} r & \alpha \end{pmatrix}^T$. Since the errors in the s measurement are uncorrelated with the other errors, as shown in (3-14), we can also redefine the covariance matrix by excluding the last row and column. Since the LIDAR scanner is affected by occlusion, it is possible that over time the true value of the s parameter for an observed line segment can change as different segments of the same infinite line are observed. For this reason, we remove these measurements from the estimation process as we are only interested in improving the pose estimates. Effectively, we only estimate the parameters of the *infinite* line.

3-3-1 Line Factor

A line in the world frame can be transformed to the body frame using

$$ {}^{\mathrm{B}}\mathbf{l} = h({}^{\mathrm{W}}\mathbf{l}, \boldsymbol{\xi}) = \begin{bmatrix} {}^{\mathrm{W}}r - x\cos({}^{\mathrm{W}}\alpha) - y\sin({}^{\mathrm{W}}\alpha) \\ {}^{\mathrm{W}}\alpha - \gamma \end{bmatrix} \tag{3-22} $$

The measurement model for the j^{th} line feature at time t is given by

$$ \mathbf{z}_t^j = h({}^{\mathrm{W}}\mathbf{l}, \boldsymbol{\xi}^t) + \boldsymbol{\epsilon}_l \tag{3-23} $$

where $\boldsymbol{\epsilon}_l \sim \mathcal{N}\left(0, P_j^t\right)$ is the measurement uncertainty as given by Equation 3-14.

3-3-2 Generating Encoder Factors

It is not computationally feasible to initialise a new node in the graph at each measurement of the wheel encoders as they arrive at a higher rate than the LIDAR measurements. We summarize the encoder measurements between the LIDAR nodes so that we do not add any extra nodes.

More formally, we summarize N encoder measurements $\{\mathbf{z}^{t_1}, \mathbf{z}^{t_2}, \mathbf{z}^{t_3}, \ldots, \mathbf{z}^{t_N}\}$between two time instants t_0, t_N, given by $\mathbf{z}^{t_i} = \begin{pmatrix}\delta x^{t_i} & \delta\gamma^{t_i}\end{pmatrix}$ as a single pose constraint denoted by $\boldsymbol{\xi}^{t_0,t_N}$.

If the initial pose is $\boldsymbol{\xi}^{t_0}$, we can recursively apply the following equation to obtain a single pose transformation.

$$\boldsymbol{\xi}^{t_0,t_{i+1}} = f(\boldsymbol{\xi}^{t_0,t_i}, \mathbf{z}^{t_{i+1}}) = \begin{bmatrix} x^{t_0,t_i} + \delta x^{t_{i+1}} \cos\left(\gamma^{t_0,t_i} + \frac{\delta\gamma^{t_{i+1}}}{2}\right) \\ y^{t_0,t_i} + \delta x^{t_{i+1}} \sin\left(\gamma^{t_0,t_i} + \frac{\delta\gamma^{t_{i+1}}}{2}\right) \\ \gamma^{t_0,t_i} + \delta\gamma^{t_{i+1}} \end{bmatrix} \tag{3-24}$$

Using, Equation 3-24, we can summarize the encoder measurements between the LIDAR measurement time stamps, thus preventing the addition of extra nodes.

3-3-3 Encoder Factor

The encoder, after the process described in Section 3-3-2 measures the transform between two poses, given by $\boldsymbol{\xi}^{t_1,t_2} = \begin{pmatrix}\delta x^{t_1,t_2} & \delta y^{t_1,t_2} & \delta\gamma^{t_1,t_2}\end{pmatrix}^T$

$$\boldsymbol{\xi}^{t_1,t_2} = f(\boldsymbol{\xi}^{t_1}, \boldsymbol{\xi}^{t_2}) = \begin{bmatrix} \cos\gamma^{t_1} & \sin\gamma^{t_1} & 0 \\ -\sin\gamma^{t_1} & \cos\gamma^{t_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^{t_2} - x^{t_1} \\ y^{t_2} - y^{t_1} \\ \gamma^{t_2} - \gamma^{t_1} \end{bmatrix} \tag{3-25}$$

The measurement model is therefore given by:

$$\mathbf{z}^{t_1,t_2} = f(\boldsymbol{\xi}^{t_1}, \boldsymbol{\xi}^{t_2}) + \boldsymbol{\epsilon}_w \tag{3-26}$$

where $\boldsymbol{\epsilon_w} \sim \mathcal{N}(0, Q_w)$ is the noise on the pose transformation, with

$$Q_w = \begin{pmatrix} \sigma_{\delta x} & 0 & 0 \\ 0 & \sigma_{\delta y} & 0 \\ 0 & 0 & \sigma_{\delta\gamma} \end{pmatrix} \tag{3-27}$$

3-3-4 Maximum A-Posteriori Estimation

At every time instant t_k that a LIDAR scan is received, we formulate a MAP estimation problem to estimate the trajectory within a window.

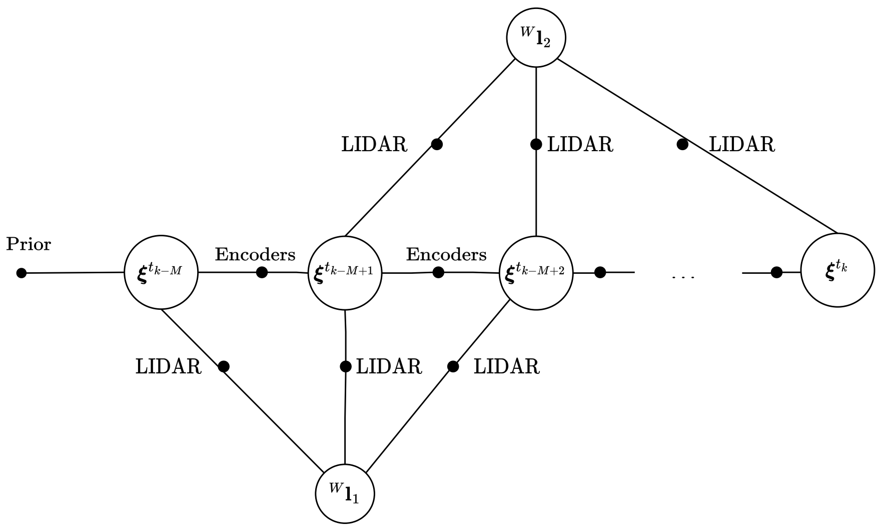


Figure 3-3: Factor Graph Representation corresponding to the optimization problem in (3-32)

For the k^{th} optimization problem, we estimate the trajectory from t_{k-M} to t_k. The value of M is the length of the horizon, and can be chosen so as to keep the $t_k - t_{k-M}$ as close to constant as possible, given that the sampling rate of the sensors are not constant.

We define $\mathcal{K}_k$ as the set of LIDAR keyframe time instances for this k^{th} optimization problem and $\mathcal{L}_k$ as the set of the line landmark indices for the kth optimization problem. The set $\mathcal{X}_k$ defines the set of poses and lines that are to be estimated.

$$\mathcal{X}_k = \left[\bigcup_{\forall t \in \mathcal{K}_k} \boldsymbol{\xi}^t \bigcup_{\forall j \in \mathcal{L}_k} {}^{\mathrm{W}}\mathbf{l}_j \right] \tag{3-28}$$

The set of prior, line and encoder measurements $\mathcal{Z}_K$ is given by

$$\mathcal{Z}_k = \left[\mathbf{z}^{t_{k-M}} \bigcup_{\forall t \in \mathcal{K}_k, j \in \mathcal{L}_k} \mathbf{z}_j^t \bigcup_{\forall i = k-M}^{k-1} \mathbf{z}^{t_i, t_{i+1}} \right] \tag{3-29}$$

We also have a prior given by $\mathbf{z}^{t_{k-M}} \sim \mathcal{N}\left(\boldsymbol{\xi}^{t_{k-M}}, \Sigma^{t_{k-M}}\right)$. We maximize the conditional distribution to find the optimal trajectory.

$$\mathcal{X}_k^* \triangleq \arg\max_{\mathcal{X}_k} p(\mathcal{X}_k | \mathcal{Z}_k) \tag{3-30}$$

We can alternatively minimize the negative log likelihood of the posterior as follows,

$$\mathcal{X}_k^* = \arg\max_{\mathcal{X}_k} \frac{p(\mathcal{Z}_k | \mathcal{X}_k) p(\mathcal{X}_k)}{p(\mathcal{Z}_k)} = \arg\min_{\mathcal{X}_k} \left[-\log p(\mathcal{X}_k ; \mathcal{Z}_k) - \log p(\mathcal{X}_k) \right] \tag{3-31}$$

Using the models from (3-22) and (3-25), this becomes a non-linear least squares problem:

$$\begin{aligned}\mathcal{X}_k^* = \arg\min_{\mathcal{X}_k} \sum_{t \in \mathcal{K}_k} \sum_{j \in \mathcal{L}_k} \left(\left|\left| h({}^{\mathrm{W}}\mathbf{l}_j, \boldsymbol{\xi}^t) - \mathbf{z}_j^t \right|\right|^2_{P_j^t} \right) \\ + \sum_{t_1, t_2 \in \mathcal{K}_k, t_1 \neq t_2} \left(\left|\left| f(\boldsymbol{\xi}^{t_1}, \boldsymbol{\xi}^{t_2}) - \mathbf{z}^{t_i, t_{i+1}} \right|\right|^2_{P_w^{t_i, t_{i+1}}} \right) \\ + \left|\left| \boldsymbol{\xi}^{t_{k-M}} - \mathbf{z}^{t_{k-M}} \right|\right|^2_{\Sigma^{t_{k-m}}}\end{aligned} \tag{3-32}$$

We solve this non-linear least squares solver at every t_k through the Levenberg-Marquardt method to get a smoothed trajectory and its estimate. More details on how this is implemented are provided in Appendix A.

The last state in the estimated trajectory $\boldsymbol{\xi}^{t_k}$ is therefore the resultant optimized state at each LIDAR instant, and is available as odometry at the LIDAR rate.

Chapter 4

Full 3D Motion Estimation

In this Chapter, we discuss our Extended Kalman Filter (EKF) and the associated models we use to fuse wheel encoders with the measurements from the IMU to produce 6-DoF odometry. We begin by defining the state vector to be estimated, along with relevant process and measurement models. We then present the standard EKF, and a square-root formulation of the algorithm that we use to alleviate the effect of numerical issues.

4-1 Dynamic Model

We define the state vector to be estimated as:

$$\begin{aligned}\mathbf{x}^T &= \begin{bmatrix} {}^{\mathrm{W}}\mathbf{p}^T & {}^{\mathrm{W}}\mathbf{v}^T & {}^{\mathrm{WB}}\boldsymbol{\theta}^T & \mathbf{b}_{acc}^T & \mathbf{b}_{gyro}^T \end{bmatrix} \\ \mathbf{u}^T &= \begin{bmatrix} {}^{\mathrm{B}}F^T & {}^{\mathrm{B}}\boldsymbol{\omega}^T \end{bmatrix}\end{aligned} \tag{4-1}$$

where we use the definitions from Section 2-2. $\mathbf{u}$ is the vector of observed linear accelerations and angular velocities, that is, the IMU measurements. We also use the rotation ${}^{\mathrm{WB}}R$ that transforms a vector from the body to world frame, given by Equation 2-2. Henceforth, we omit the superscript on the rotation matrix for brevity.

Finally, we also have an offset vector $\vec{r} \in \mathbb{R}^3$ which denotes the offset between the IMU and the center of rotation of the robot. We assume this vector to be known very accurately in advance from the robots CAD models. When this offset is non-zero, as it is in our case, centripetal accelerations can cause artefacts on the IMU measurements. This vector is known in advance from the CAD model of the robot.

The IMU measurements can then be related to the true acceleration of the robot using the equation,

$${}^{\mathrm{B}}F = R^T \left({}^{\mathrm{W}}\mathbf{a} - \mathbf{g}\right) + {}^{\mathrm{B}}\boldsymbol{\omega} \times {}^{\mathrm{B}}\boldsymbol{\omega} \times \vec{r} + \mathbf{b}_{acc} \tag{4-2}$$

To derive our dynamic model, we start with the fundamental continuous time relations,

$$\begin{aligned} {}^{\mathrm{W}}\dot{\mathbf{p}} &= {}^{\mathrm{W}}\mathbf{v} \\ {}^{\mathrm{W}}\dot{\mathbf{v}} &= {}^{\mathrm{W}}\mathbf{a} \\ \dot{R} &= R\left[{}^{\mathrm{B}}\boldsymbol{\omega}_{\times}\right] \\ {}^{\mathrm{W}}\mathbf{a} &= R\left({}^{\mathrm{B}}F - \mathbf{b}_{acc} - {}^{\mathrm{B}}\boldsymbol{\omega} \times {}^{\mathrm{B}}\boldsymbol{\omega} \times \vec{r}\right) + \mathbf{g} \end{aligned} \tag{4-3}$$

The IMU measurements are received at roughly 50 Hz. If we assume **u** to be constant over the sampling interval T, from t_k to t_{k+1}, we arrive at the approximate discrete model,

$$\begin{aligned} {}^{\mathrm{W}}\mathbf{p}^{t_{k+1}} &= {}^{\mathrm{W}}\mathbf{p}^{t_k} + T\,{}^{\mathrm{W}}\mathbf{v}^{t_k} + \frac{T^2}{2}{}^{\mathrm{W}}\mathbf{a}^{t_k} + \boldsymbol{\eta}_p \\ {}^{\mathrm{W}}\mathbf{v}^{t_{k+1}} &= {}^{\mathrm{W}}\mathbf{v}^{t_k} + T\,{}^{\mathrm{W}}\mathbf{a} + \boldsymbol{\eta}_v \\ R^{t_{k+1}} &= R^{t_k}\exp\left(({}^{\mathrm{B}}\boldsymbol{\omega} - {}^{\mathrm{B}}\delta_{gyr,k})T\right) + \boldsymbol{\eta}_\gamma \\ \mathbf{b}_{acc}^{t_{k+1}} &= \mathbf{b}_{acc}^{t_k} + \boldsymbol{\epsilon}_{acc} \\ \mathbf{b}_{gyro}^{t_{k+1}} &= \mathbf{b}_{acc}^{t_k} + \boldsymbol{\epsilon}_{gyro} \end{aligned} \tag{4-4}$$

We model that the accelerometer and gyro measurements are affected by a noise $\boldsymbol{\eta}_p \sim \mathcal{N}(0, Q_{\boldsymbol{\eta},acc})$, $\boldsymbol{\eta}_v \sim \mathcal{N}(0, Q_{\boldsymbol{\eta},acc})$, $\boldsymbol{\eta}_\gamma \sim \mathcal{N}(0, Q_{\boldsymbol{\eta},gyro})$. We also model the bias terms as being affected by $\boldsymbol{\epsilon}_{acc} \sim \mathcal{N}(0, Q_{\boldsymbol{\epsilon},acc})$, $\boldsymbol{\epsilon}_{gyro} \sim \mathcal{N}(0, Q_{\boldsymbol{\epsilon},gyro})$.

This gives us the approximate state space model

$$\mathbf{x}^{t_{k+1}} = f_I(\mathbf{x}^{t_k}, \mathbf{u}^{t_k}) + \epsilon^{t_k} = \begin{bmatrix} {}^{\mathrm{W}}\mathbf{p}^{t_k} + T\,{}^{\mathrm{W}}\mathbf{v}^{t_k} + \frac{T^2}{2}{}^{\mathrm{W}}\mathbf{a}^{t_k} \\ {}^{\mathrm{W}}\mathbf{v}^{t_k} + T\,{}^{\mathrm{W}}\mathbf{a} \\ \log\left(R^{t_k}\exp\left(({}^{\mathrm{B}}\omega - {}^{\mathrm{B}}\delta_{gyr,k})T\right)\right) \\ \mathbf{b}_{acc}^{t_k} \\ \mathbf{b}_{gyro}^{t_k} \end{bmatrix} + \epsilon^{t_k} \tag{4-5}$$

where ϵ^{t_k} is the collection of all the noises, and is distributed as $\epsilon^{t_k} \sim \mathcal{N}(0, Q_i)$, with the noise covariance matrix Q_i given by

$$Q_i = \begin{bmatrix} Q_{\boldsymbol{\eta},acc} & 0 & 0 & 0 & 0 \\ 0 & Q_{\boldsymbol{\eta},acc} & 0 & 0 & 0 \\ 0 & 0 & Q_{\boldsymbol{\eta},gyro} & 0 & 0 \\ 0 & 0 & 0 & Q_{\boldsymbol{\epsilon},gyro} & 0 \\ 0 & 0 & 0 & 0 & Q_{\boldsymbol{\epsilon},gyro} \end{bmatrix} \tag{4-6}$$

4-2 Measurement Models

The measurements from the wheel encoder are the angular velocity on the ground plane, and the first component of the linear velocity (see (2-7)). We utilise the holonomic constraints of the robots to augment the measurement model with two zero velocity pseudo-measurements corresponding to lateral and vertical velocities. The addition of pseudo-measurementsis a well-known technique [26] [19] to include some form of existing knowledge about the system dynamics into the filter indirectly.

The velocity measurement is therefore

$$ {}^{\mathrm{B}}\mathbf{v} = \begin{pmatrix} {}^{\mathrm{B}}v_x \\ 0 \\ 0 \end{pmatrix} \tag{4-7} $$

The angular velocity measurement(ω) is expressed as a measurement of the yaw angle using the relation,

$$ {}^{\mathrm{W}}\gamma^{t_k} = {}^{\mathrm{W}}\gamma^{t_{k-1}} + {}^{B}\omega * T \tag{4-8} $$

The encoder measurement model now becomes,

$$ \mathbf{y}^{t_k} = h(\mathbf{x}^{t_k}) = \begin{bmatrix} R^T \cdot {}^{\mathrm{W}}\mathbf{v}^{t_k} \\ \\ {}^{\mathrm{W}}\gamma^{t_k} \end{bmatrix} + \epsilon_h \tag{4-9} $$

where $\epsilon_h \sim \mathcal{N}(0, Q_h)$. The elements of the matrix Q_h can be tuned to reduce or strengthen the effect of the corrections from the wheel encoders and the zero-velocity pseudo measurements.

Note that this measurement model makes the implicit assumption that the robot is on a flat plane in (4-8)

4-3 Extended Kalman Filter

The Extended Kalman Filter is arguably the most popular method for fusing data from multiple sources. It involves a time update step and a correction step at each time step. We use the IMU model to perform the time update, and the velocity measurements from the wheel encoder for the correction. Figure 4-1 shows the standard predict and update cycles for the filter. The notation $\hat{x}_{k|k-1}$ denotes the estimate of x at t_k given all the measurements upto t_{k-1}.

Note: the relevant jacobians that we use in Algorithm 2 and Algorithm 3 were computed numerically, through the use of a forward euler approximation.

A single time update and correction cycle of the Extended Kalman Filter is shown in Algorithm 2

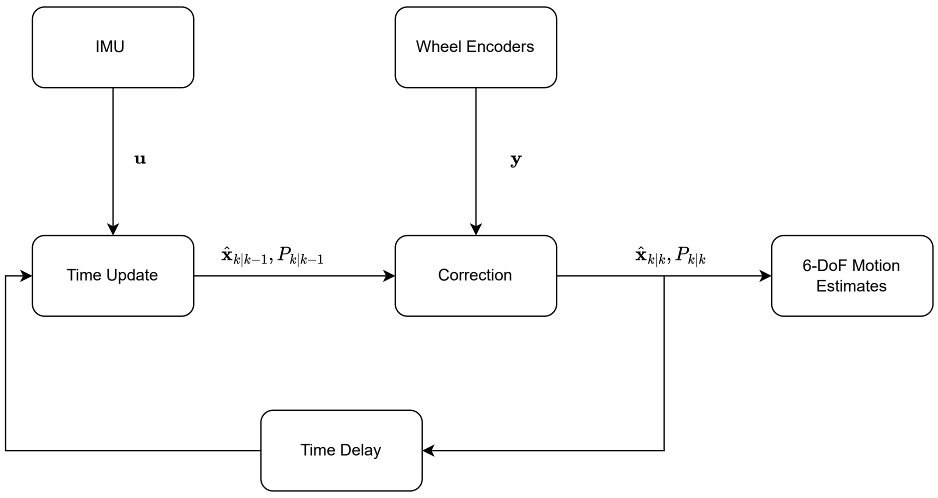


Figure 4-1: Structure of the EKF

Algorithm 2 Extended Kalman Filter

```
Input: Previous state $\mathbf{x}_{k-1|k-1}$, IMU Measurements $\mathbf{u}_{k-1}$, Encoder Measurements $\mathbf{y}^{t_k}$, and
    associated covariances $P_{k-1|k-1}$, $Q_i$, $Q_h$
Output: Next state $\mathbf{x}_{k|k}$ and associated covariance $P_{k|k}$
    $F^{t_k} = \frac{\partial}{\partial x} f(\mathbf{x},\mathbf{u}))\Big|_{\hat{\mathbf{x}}_{\mathbf{k-1|k-1}},\mathbf{u}_{k-1}}$
    $\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{\mathbf{k-1|k-1}}, \mathbf{u}^{t_k})$
    $P_{k|k-1} = F^{t_k} P_{k-1|k-1} {F^{t_k}}^T + Q_i$
    $H^{t_k} = \frac{\partial}{\partial u} h(\mathbf{x})\Big|_{\hat{\mathbf{x}}_{\mathbf{k|k-1}}}$
    $\tilde{\mathbf{y}} = \mathbf{z}^{t_k} - h(\hat{\mathbf{x}}_{k|k-1})$
    $S^{t_k} = H P_{k|k-1} H^T + Q_h^{t_k}$
    $K^{t_k} = P_{k|k-1} H^T S^{-1}$
    $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K^{t_k}\tilde{\mathbf{y}}$
    $P_{k|k} = (I - K^{t_k} H) P_{k|k-1}$
```

4-3-1 Square Root Formulation

In practice, our EKF proved to be unstable due to numerical issues with very small covariance matrices. To alleviate the numerical stability issue, we used a square root formulation of the EKF equations, adapted from the method in [27]. In this formulation, all the covariance matrices are represented with their square roots. This means that we store and update the square root U instead of P, which are related by $P = U^T U$. This technique has the advantage of halving the bits of precision required to accurately store these matrices.

In order to rewrite our EKF equations to work directly with square-roots of covariances, we must be able to add two matrices with just their square roots, i.e, we need to calculate $\sqrt{A+B}$, without squaring $\sqrt{A}$ and $\sqrt{B}$ and adding them. This can be done as follows

$$A + B = \begin{bmatrix} \sqrt{A}^T & \sqrt{B}^T \end{bmatrix} \begin{bmatrix} \sqrt{A} \\ \sqrt{B} \end{bmatrix}$$

We can take the qr decomposition of the matrix on the right as,

$$Q, R = \mathrm{qr}\left(\begin{bmatrix} \sqrt{A} \\ \sqrt{B} \end{bmatrix} \right)$$

Plugging this into the previous equation, we get

$$A + B = R^T Q^T Q R$$
$$A + B = R^T R$$

Therefore, we get $\sqrt{A+B} = \mathrm{qr}_r \left(\begin{bmatrix} \sqrt{A} \\ \sqrt{B} \end{bmatrix} \right)$

where qr_r is the operation that gives us only the R matrix of a QR decomposition.

Using this technique, the Extended Kalman Filter equations can be rewritten as in Algorithm 3. For a more detailed derivation, see [27].

Algorithm 3 Square-Root Extended Kalman Filter

```
Input: Previous state $\mathbf{x}_{k-1|k-1}$, IMU Measurements $\mathbf{u}_{k-1}$, Encoder Measurements $\mathbf{y}^{t_k}$, and
    associated covariance square roots $\sqrt{P_{k-1|k-1}}$, $\sqrt{Q_i}$, $\sqrt{Q_h}$
Output: Next state $\mathbf{x}_{k|k}$ and associated covariance $P_{k|k}$
  $F^{t_k} = \frac{\partial}{\partial x} f(\mathbf{x}, \mathbf{u}))\Big|_{\hat{\mathbf{x}}_{\mathbf{k-1|k-1}}, \mathbf{u}_{k-1}}$
  $\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{\mathbf{k-1|k-1}}, \mathbf{u}^{t_k})$
  $\sqrt{P}_{k|k-1} = \mathrm{qr}_r \left( \begin{bmatrix} \sqrt{P}_{k-1|k-1} {F^{t_k}}^T \\ \sqrt{Q_i} \end{bmatrix} \right)$
  $H^{t_k} = \frac{\partial}{\partial u} h(\mathbf{x})\Big|_{\hat{\mathbf{x}}_{\mathbf{k|k-1}}}$
  $\tilde{\mathbf{y}} = \mathbf{z}^{t_k} - h(\hat{\mathbf{x}}_{k|k-1})$
  $\tilde{S}^{t_k} = \mathrm{qr}_r \left( \begin{bmatrix} \sqrt{P}_{k|k-1} {H^{t_k}}^T \\ \sqrt{R} \end{bmatrix} \right)$
  $K^{t_k} = \left[ \tilde{S^{t_k}}^{-1} \left( \tilde{S^{t_k}}^{-1} H^{t_k} \right) \sqrt{P}^T_{k|k-1} \sqrt{P}_{k|k-1} \right]^T$
  $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K^{t_k} \tilde{\mathbf{y}}$
  $P_{k|k} = \mathrm{qr}_r \left( \begin{bmatrix} \sqrt{P}_{k|k-1} (I - KH)^T) \\ \sqrt{R} K^T \end{bmatrix} \right)$
```

Chapter 5

Experiments and Results

This chapter describes the simulations and real-world experiments we conducted to analyse the performance of the proposed methods. These experiments were carefully selected to highlight the strengths of each algorithm, while also highlighting their weaknesses and situations under which they fail

Feature Extraction and Tracking: For the feature extraction and tracking, we do not have an easy way to directly quantify performance and accuracy. However, from the results of the 2D Odometry and simulations, we can make some inferences regarding its properties.

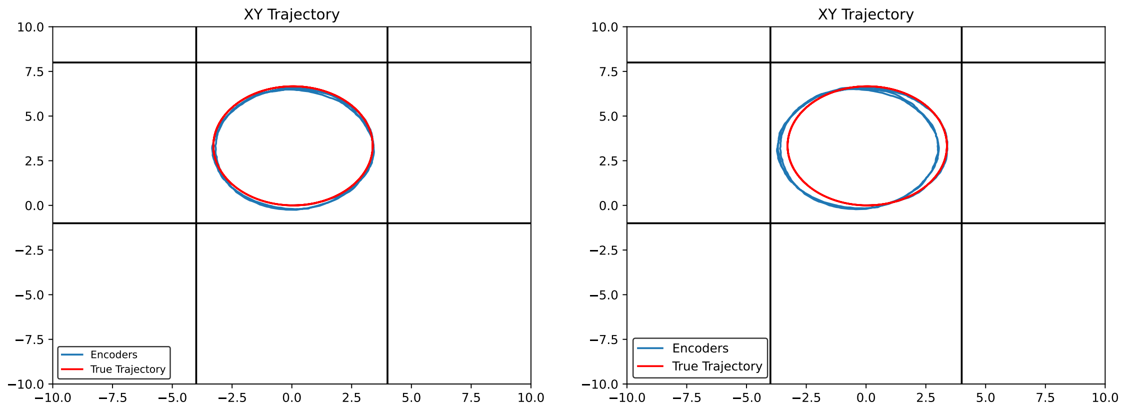
2D Pose Estimation: The 2D pose estimation was tested in isolation with a simulated trajectory in a room with four walls. With this simulation, we analyze the results of the Moving Horizon Estimation (MHE) approach. We pay special attention to the effect of the horizon length and simulate cases with non-Gaussian errors such as wheel slip and outliers in line matching. Next, we present two real-world datasets that allow us to demonstrate the performance of the algorithm under real conditions.

Full 3D Motion Estimation: We tested this algorithm on two other datasets, one in which a wheel slip occurs and another in which the robot was manually rolled and pitched to test the algorithm when the horizontal plane assumption is violated.

Note on real-world experiments: We lack ground-truth data in all our real-world experiments. We attempted to work through this through the use of waypoints, straight trajectories, and closed-loop trajectories, which are sufficient for us to make qualitative inferences. However, due to experimental tolerance, it is difficult to provide metrics for these experiments.

5-1 Simulations

To validate the performance of the 2D pose estimation scheme and analyze it in isolation from the feature extraction and tracking module, we simulated a room the robot moving in a circular trajectory in a room with four walls, for 60 s. The ground truth trajectory and trajectory from pure encoder integration are shown in Figure 5-1a. We assume that we get encoder and line measurements at a constant rate of 10 Hz, with uncorrelated Gaussian noise affecting them both.



(a) No slip **(b)** Added Wheel Slip

Figure 5-1: True trajectory of the robot(red) and the trajectory obtained from pure encoder integration(blue)

We tested our MHE scheme on three different variations of this experiment, which were simulated as follows:

Experiment 1: We add Gaussian noise on the wheel encoder measurements and the line measurements.

Experiment 2: In addition to the Gaussian noise, we simulated a wheel slip. The wheel slip was modelled as a constant error in the ω_l term from (2-8), present between $t = 10$ and $t = 10.5$. The trajectory obtained from purely integrating wheel encoders is shown in Figure 5-1b

Experiment 3: The wheel did not slip in this experiment. We add outliers in the line measurements of one of the lines at $t = 10$, $t = 20$ and $t = 30$.

The noise levels for all the simulations are specified in Table 5-1

We generate results with different horizon lengths of $M = 2$, $M = 10$, $M = 50$, along with the results of a full trajectory smoother($M = \infty$). Note that the full trajectory smoother was run on the complete trajectory with all the measurements, and not incrementally at each time instant.

Table 5-1: Errors and Noise Parameters for the Simulated Datasets

Experiment	P_{rr}	$P_{\alpha\alpha}$	$\sigma_{\delta x}$	$\sigma_{\delta y}$	$\sigma_{\delta\gamma}$	Wheel Slip	Outliers
1	10^{-2}	10^{-3}	10^{-3}	10^{-2}	10^{-3}	None	None
2	10^{-2}	10^{-3}	10^{-3}	10^{-2}	10^{-3}	Yes	None
3	10^{-2}	10^{-3}	10^{-3}	10^{-2}	10^{-3}	None	Yes

5-1-1 Error Analysis

For each test, we select four metrics to compare the estimators. These metrics are given by

$$e_1 = \frac{1}{N}\sqrt{\sum_{i=0}^{N} \left|\left| \begin{bmatrix} \hat{x}^{t_i} \\ \hat{y}^{t_i} \end{bmatrix} - \begin{bmatrix} x^{t_i} \\ y^{t_i} \end{bmatrix} \right|\right|} \tag{5-1}$$

$$e_2 = \frac{1}{N}\sqrt{\sum_{i=0}^{N} ||\hat{\gamma}^{t_i} - \gamma^{t_i}||} \tag{5-2}$$

$$e_3 = \frac{1}{4}\sqrt{\sum_{j=1}^{3} ||\hat{r}_j - r_j||} \tag{5-3}$$

$$e_4 = \frac{1}{4}\sqrt{\sum_{j=1}^{4} ||\hat{\alpha}_j - \alpha_j||} \tag{5-4}$$

which are RMSE errors for the translational and rotational estimates of the estimated poses and line parameters respectively. e_3 and e_4 give us a good measure of the long-term drift of the estimator.

5-1-2 Results and discussion

The resultant error plots for the three experiments are shown in Figure 5-2, Figure 5-3 and Figure 5-4 respectively. The corresponding error metrics are presented in Table 5-2.

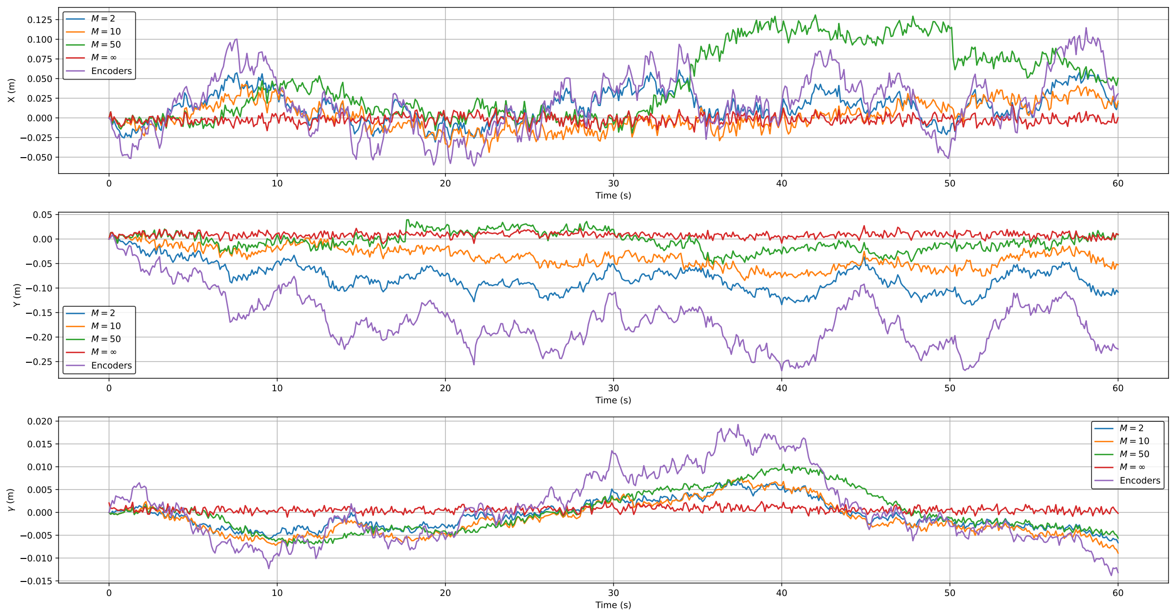


Figure 5-2: Pose estimation error for different window lengths for the first simulated dataset

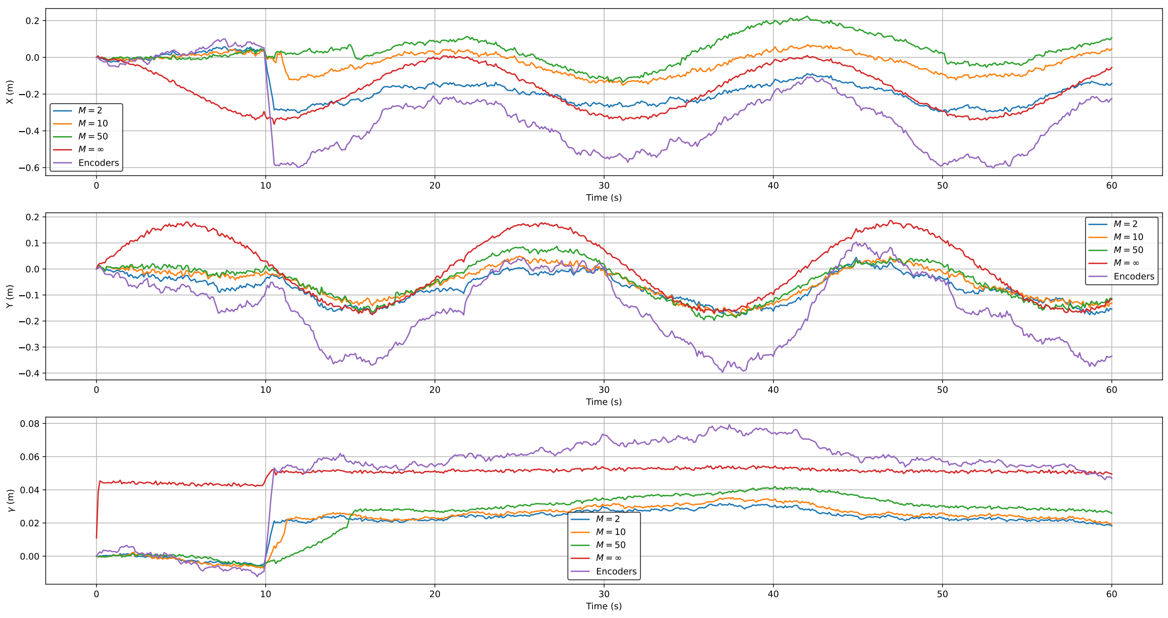


Figure 5-3: Pose estimation error for different window lengths for the second simulated dataset, with an added wheel slip at $t = 10$

From the error plots mentioned, it is clear that the Moving Horizon Estimation (MHE) scheme is able to reduce the error in the pose estimates, especially with respect to yaw angles. We notice a clear trend that the estimation error metrics decrease with larger horizon lengths. This is to be expected, as more information is provided to correct the estimated trajectory.

Interestingly, in the case of wheel slip, the moving horizon scheme appears to perform *worse* when the horizon length increases. We attribute this behaviour to the fact that the "wrong" measurement is included in the estimation process for a longer period, causing each subsequent estimate to be more biased. This effect is most noticeable on the estimate of the yaw angle

Overall, we conclude that the scheme is quite susceptible to violations of the Gaussian noise assumption, but still shows some advantages in terms of reducing the drift. Especially in terms of the yaw angle, the moving-horizon approach is successful in reducing the drift over time.

5-2 2D Pose Estimation: Real World Experiments

We recorded data from all the sensors while manually controlling SAM-UVC at the Loop Robots office. In both the datasets we collected, the final pose is known, up to a certain experimental tolerance. We analyze the performance of our algorithms based on visual inspection and the final poses of the estimated trajectories.

5-2-1 Experiment 1: Straight-Line Trajectory

SAM-UVC was driven in a straight line for 5m. The trajectory obtained by integrating the encoder measurements is shown in Figure 5-5. With this trajectory, we see a visible drift in the

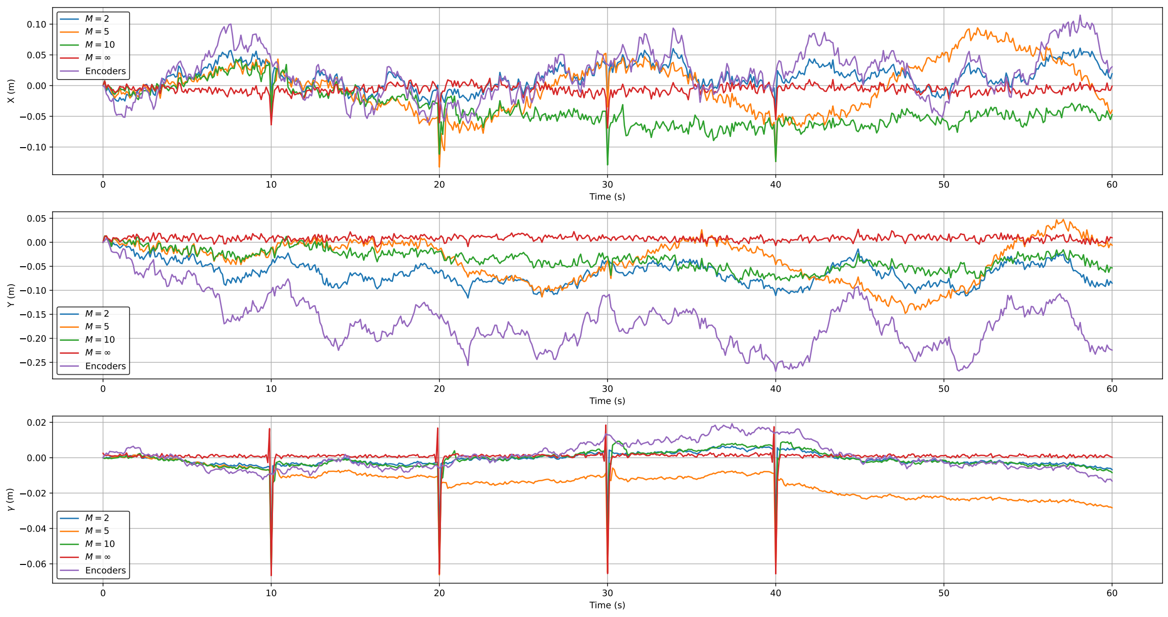
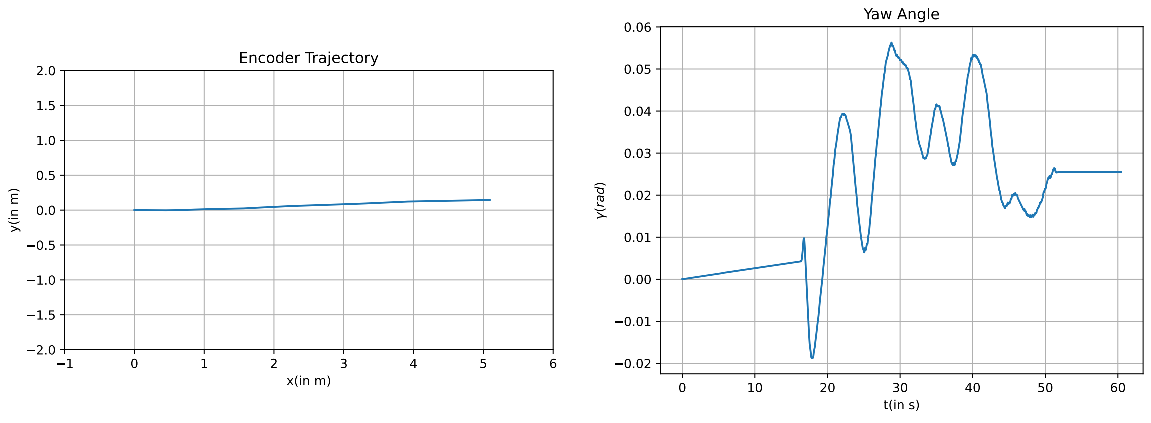


Figure 5-4: Pose estimation error for different window lengths for the third simulated dataset, with line outlier measurements added at $t = 10$, $t = 20$ and $t = 30$

yaw angle in the first 15 or so seconds, when the robot was standing still. This is attributed to the quantization noise on the wheel encoders, with the robot reporting an angular velocity of ${}^{\mathrm{B}}\omega = 2.554 \times 10^{-4}$ rad/s when stationary. This small drift in yaw, when integrated, causes the y estimate to drift off by about 20 cm.



(a) XY Trajectory Estimate **(b)** Yaw Angle Estimate

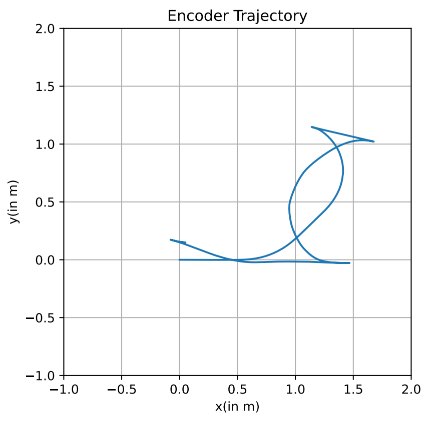
Figure 5-5: Trajectory from integrating wheel encoder measurements for the straight line trajectory

Table 5-2: Comparing the error metrics for results of the MHE on simulated data. A window size of ∞ refers to a full trajectory smoother

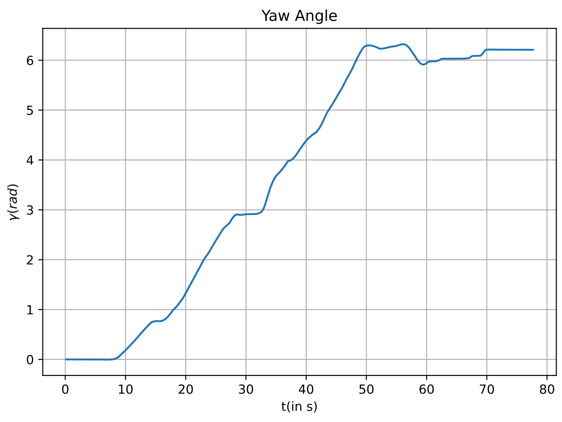
Dataset	Window Size	e_1	e_2	e_3	e_4
Dataset 1	2	0.0878	0.0032	0.0700	0.0062
	10	0.0463	0.0040	0.0434	0.0084
	50	0.0659	0.0045	0.0254	0.0051
	∞	0.0116	0.0009	0.0046	0.0006
Dataset 2	2	0.2151	0.0225	0.1152	0.0187
	10	0.1042	0.0243	0.0767	0.0194
	50	0.1234	0.0285	0.0943	0.0264
	∞	0.2417	0.0504	0.0040	0.0503
Dataset 3	2	0.0732	0.0047	0.0587	0.0062
	10	0.0696	0.0162	0.0754	0.0277
	50	0.0636	0.0063	0.0662	0.0080
	∞	0.0141	0.0058	0.0062	0.0011

5-2-2 Experiment 2: Closed Loop Trajectory

The robot finished in the same pose it as it started from. The trajectory obtained by integrating the encoder measurements is shown in Figure 5-6. These estimates show a visible drift, as the trajectory ends at a different point from where we started.



(a) XY Trajectory Estimate



(b) Yaw Angle Estimate

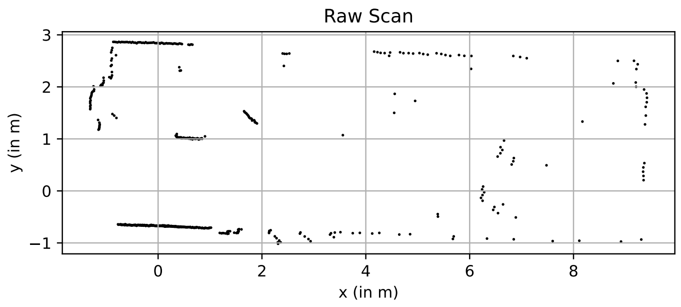
Figure 5-6: Trajectory from integrating wheel encoder measurements for the closed loop trajectory

5-2-3 Results and discussion

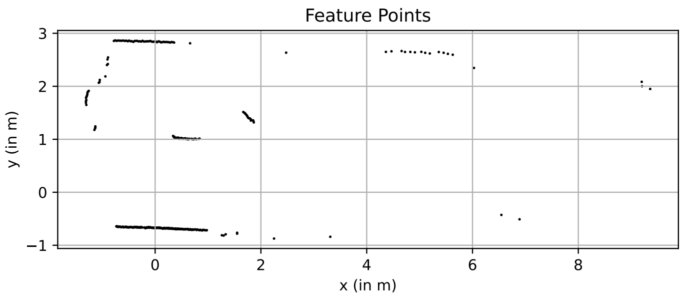
In Figure 5-13, we show the number of line features that were extracted and tracked on both the datasets. We notice that the number of features that are extracted and not tracked are minimal. This implies that our association method works well, and tracks are rarely lost.

Table 5-3: Percentage reduction of scan points of the feature point extraction scheme

Straight Line Trajectory	Closed Loop Trajectory
22.02 %	18.5 %



(a) Raw LIDAR scan



(b) Extracted Points

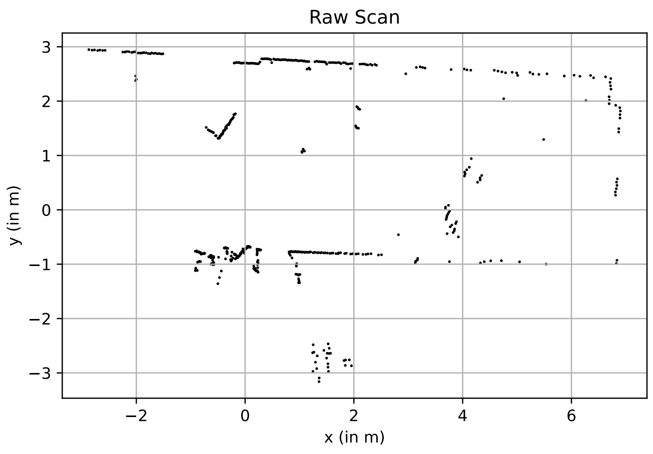
Figure 5-7: Feature Point Extraction at time $t = 20$ s for the straight line trajectory

The feature point extraction method we proposed in Section 3-1-2 succeeded in reducing the number of points to cluster and fit by around 20% on both datasets. The exact percentages are provided in Table 5-3. Figure 5-7, Figure 5-8 provide examples of the feature point extraction scheme.

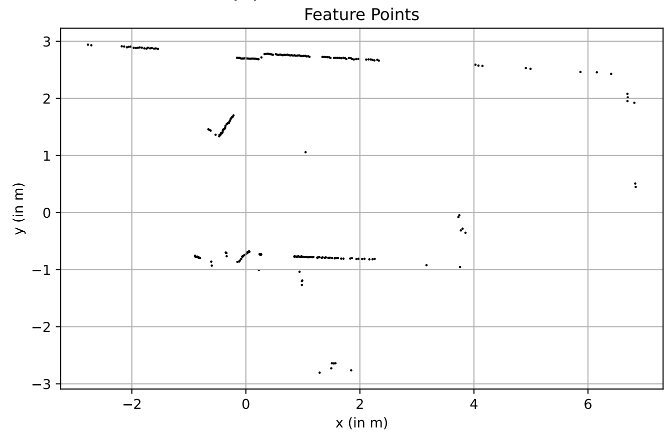
In the first experiment, from $t = 0$ s to $t = 10$ s, we see that the MHE scheme is able to remove most of the drift caused by the wheel encoders. However, some amount of noise has also been introduced on the estimates from the inclusion of the line measurements. Similarly, for most of the trajectory, we see that the y coordinate drifts by only 5 cm. At two time instants, at $t = 37.53$ s and $t = 49.5$ s, we see the disproportionate effect of outliers affecting the estimates.

To understand better the spike at $t = 37.53$ s, we show the results of the feature tracking process at this instant in Figure 5-9. In the zoomed plot, Figure 5-9b, we see that a small group of points has been clustered with the orange line, causing the estimate to be rotated. This kind of error is highly non-Gaussian, and is one example of the sources of these outliers.

Important to note is also the fact that althought the y and γ estimates seem to be corrected, the x estimate has drifted by almost a meter. This is suspected to be due to a combination of multiple effects: (1) for most of the trajectory, only two parallel lines corresponding to the



(a) Raw LIDAR scan



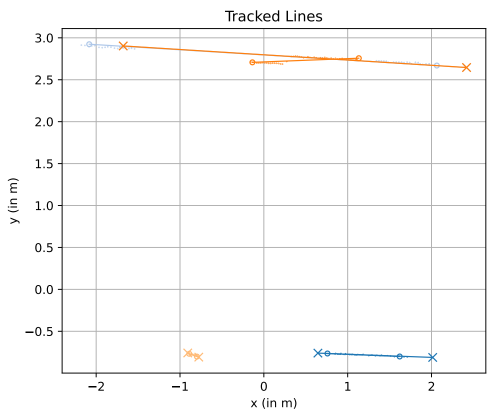
(b) Extracted Points

Figure 5-8: Feature Point Extraction at time $t = 37.53$ s for the straight line trajectory

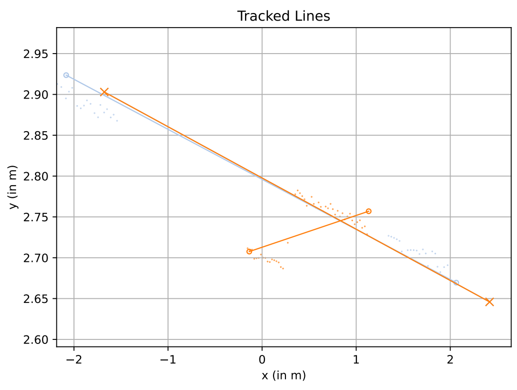
room are visible to the scanner; (2) the effect of motion distortion on the LIDAR scan.

The second experiment shows a large amount of drift and the algorithm can be said to have failed here. However, some interesting inferences can still be drawn from this experiment. First, we note that the true trajectory is a combination of curves and straight lines. There are significant sections of the trajectory that are straight lines. From Figure 5-11b, we deduce that these sections are approximately given by the intervals $t \in [0, 9]$, $t \in [27, 33]$, $t = [49, 57]$, $t = [59, 69]$ and $t = [70, 77]$. In Figure 5-12, we plot the difference in the yaw estimates from purely integrating encoder values, and the MHE scheme. Interestingly, there is close to zero growth in the difference in these time intervals. From this, we infer that the yaw angle estimate is affected whenever the yaw angle of the robot changes.

In [28], the authors discuss how rotational motion of the robot can skew LIDAR scans. They elaborate that the motion distortion is most severe when the LIDAR is simultaneously rotated and translated. This explains the drift in our yaw estimates, and also why this drift was not very noticeable in the straight-line trajectory. This suggests that the error seen in the results is a result of our assumption that the motion distortion on the LIDAR scans are negligible

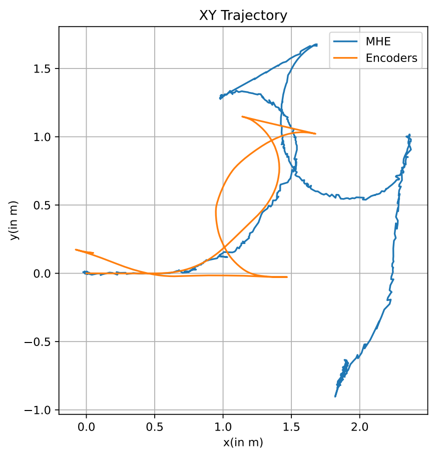


(a) Tracked Features

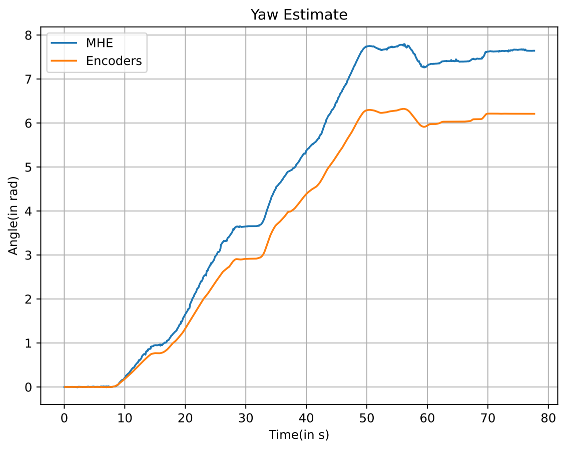


(b) Zoomed image of the outlier. Note: The image is skewed from zooming

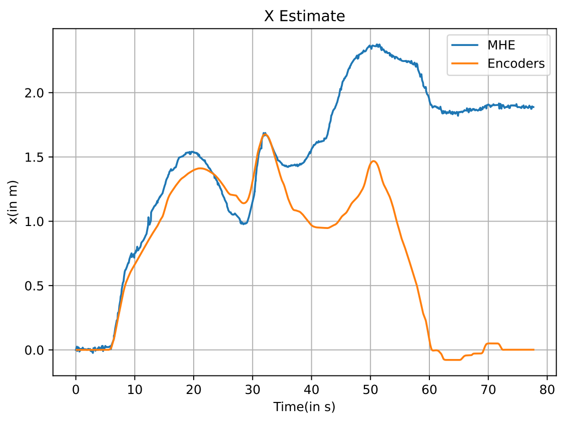
Figure 5-9: Failure of the line tracking module at $t = 37.53$ s in the straight line trajectory. Lines from the preceding scan are propagated forward using (3-18) and plotted along with the lines at the current scan. Two lines of the same color imply they are matched together.



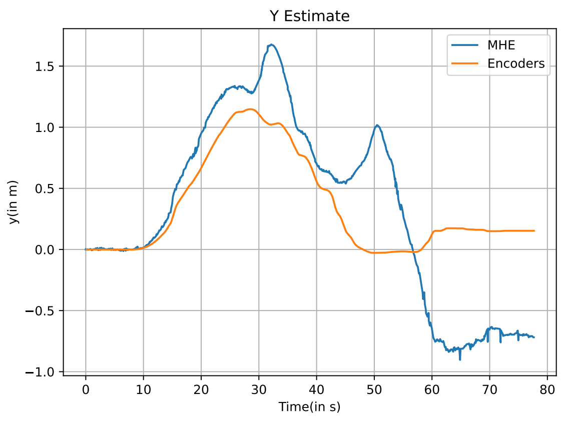
(a) XY Trajectory Estimate



(b) Yaw Angle Estimate

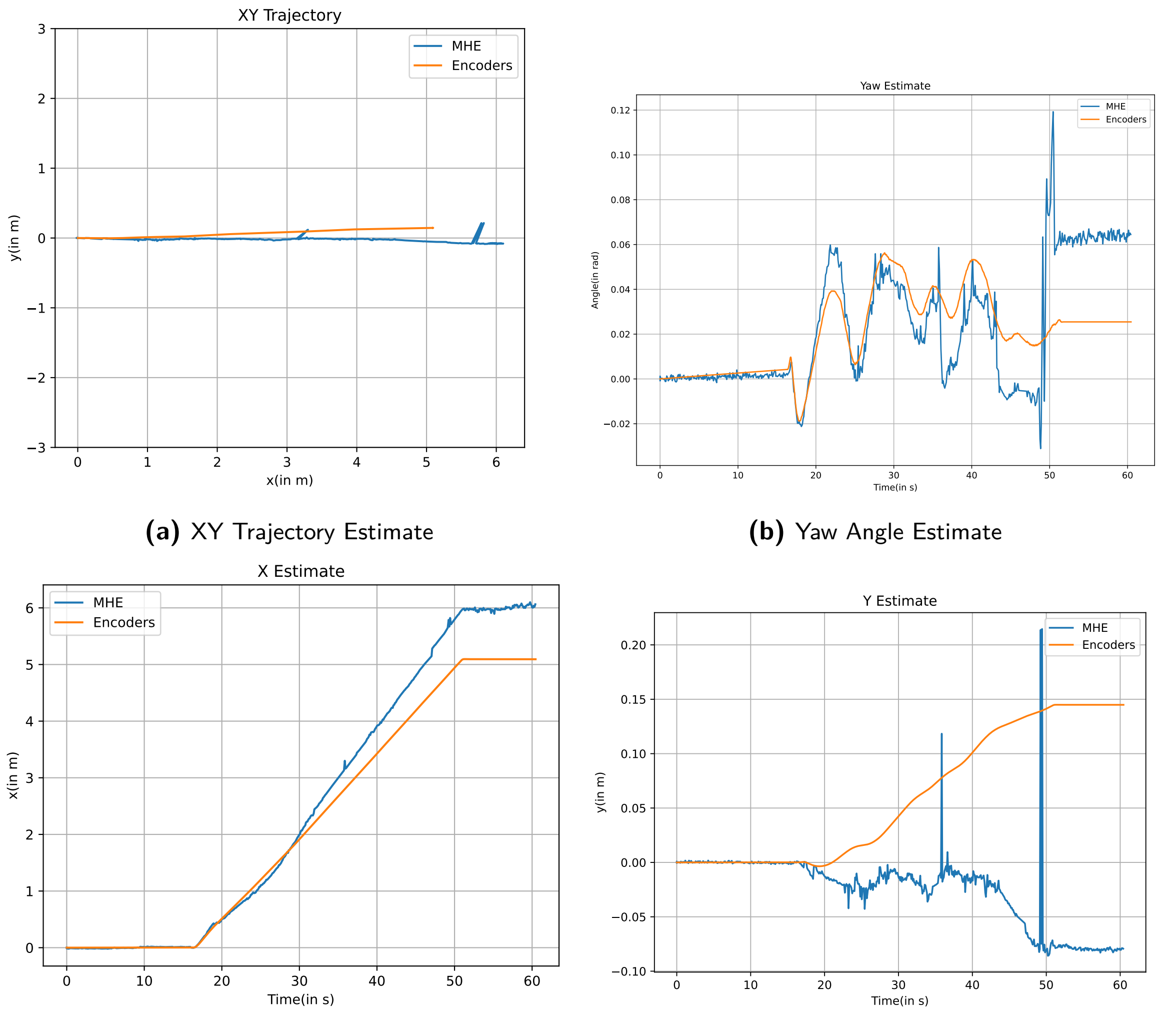


(c) X coordinate estimate from the MHE and encoders



(d) Y coordinate estimate from the MHE and encoders

Figure 5-11: Results of the MHE on the closed loop trajectory



(a) XY Trajectory Estimate

(b) Yaw Angle Estimate

(c) X coordinate estimate from the MHE and encoders

(d) Y coordinate estimate from the MHE and encoders

Figure 5-10: Results of the MHE on the straight line trajectory

5-3 Full 3D Motion Estimation

To test the EKF and validate its performance, we conducted two experiments.

5-3-1 Experiment 1: Wheel Slip

In this experiment, we traversed a long trajectory, with a series of waypoints that were measured and marked prior to the experiment. At the end of the experiment, the robot returned to where it began. Around $t = 44$ s, the robot was driven over a bump, causing a wheel slip. The results of this experiment are shown in Figure 5-14

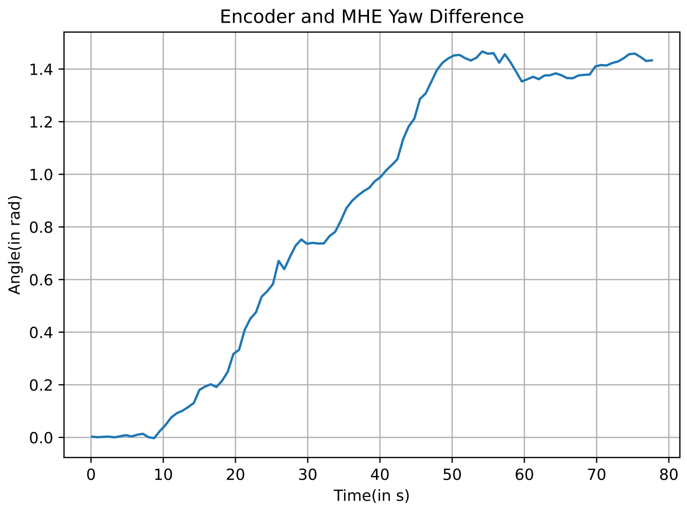


Figure 5-12: The difference in yaw angle between the MHE and encoder estimates for the closed loop trajectory

5-3-2 Experiment 2: Rolling and Pitching

This experiment was conducted to test the orientation estimation capabilities of the EKF when the horizontal plane assumption is violated. The robot underwent the following series of motions:

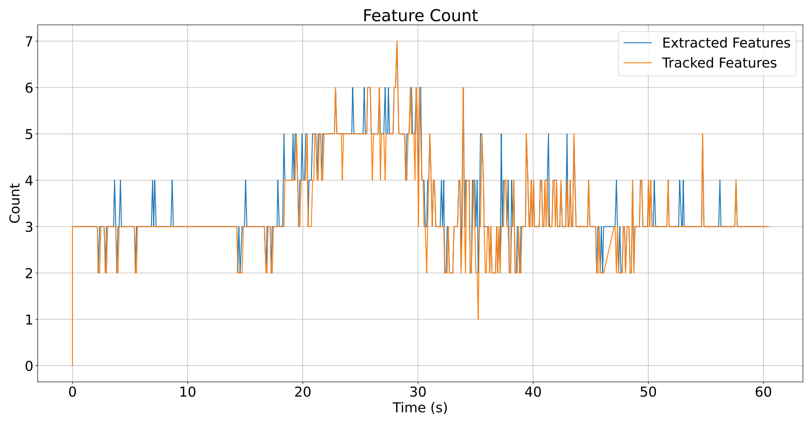
1. Very small motions on the plane
2. Pitched in the positive direction
3. Rolled in the positive direction
4. Small roll in the positive direction
5. Moved in a straight line

When the robot was lowered from the rolling or pitching motions, it was lowered a little bit at a time so as to not drop it quickly onto the floor. The roll and pitch estimates from the experiment are shown in Figure 5-16.

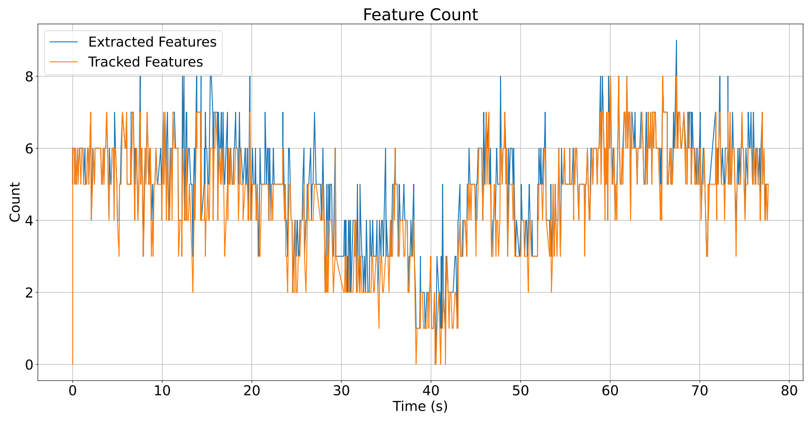
5-3-3 Results and Discussion

From the first experiment, we clearly see the benefit of fusing the IMU data with wheel encoders. The filter is able to correct for most of the wheel slip, and we even see the slip show up as a small spike in the pitch estimate at Figure 5-14b. However, we also see the roll and pitch estimates drift after a while. On closer inspection, we note that this drift increases along with the drift in the accelerometer bias estimates.

We hypothesize that is a result of the lack of observability of the system model. Observability is a property of a system that determines if the states can be estimated with the available data



(a) Straight Line Trajectory



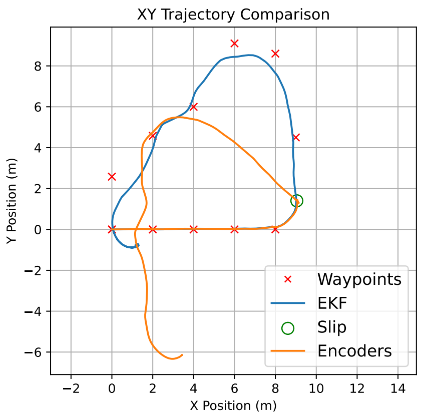
(b) Closed Loop Trajectory

Figure 5-13: Number of extracted and tracked features over time

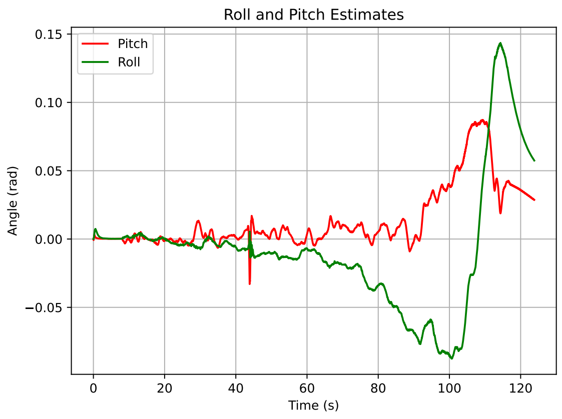
[29]. Typically, observability of a non-linear system is difficult to establish. We present an argument here as to why the accelerometer biases along with roll and pitch are unobservable.

Using solely an IMU without any other source of information, the biases on the accelerometer are not observable without the addition of extra information from other sources, even when the sensor is stationary [15]. This is because the only source of information regarding the roll, pitch and accelerometer biases come from the measurement of the gravity vector (2-9a). However, there is no unique combination of biases, roll and pitch that satisfies this equation. Over long periods of times, these estimates can therefore drift and be unreliable. In [19], the authors use a similar formulation as we do for their EKF and use period zero-roll and pitch updates to keep them from drifting.

The second experiment showcases the ability of the filter to estimate the orientation of the robot even when the horizontal plane assumption is violated. Also in this experiment, we see that the accelerometer biases vary, even though the filter remains stable. On close inspection, we also notice that the stable value of the roll and pitch does not stay the same throughout the trajectory, providing further evidence that these states are not observable in this model.

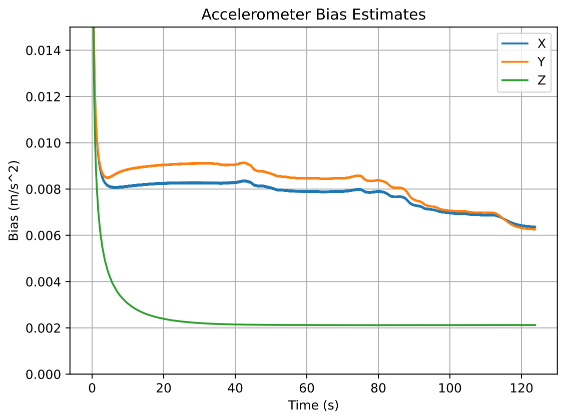


(a) Trajectory estimates on the horizontal plane.

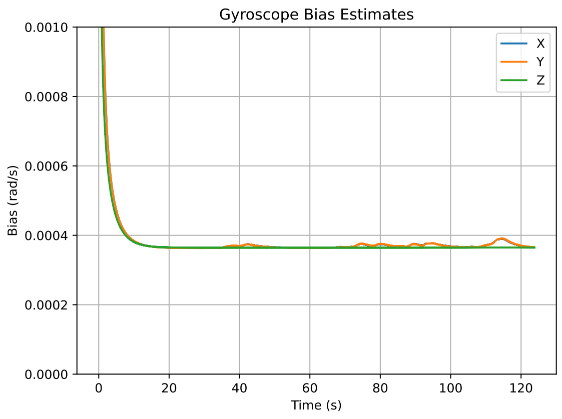


(b) Roll and Pitch Estimates

Figure 5-14: Results of the EKF in the first experiment

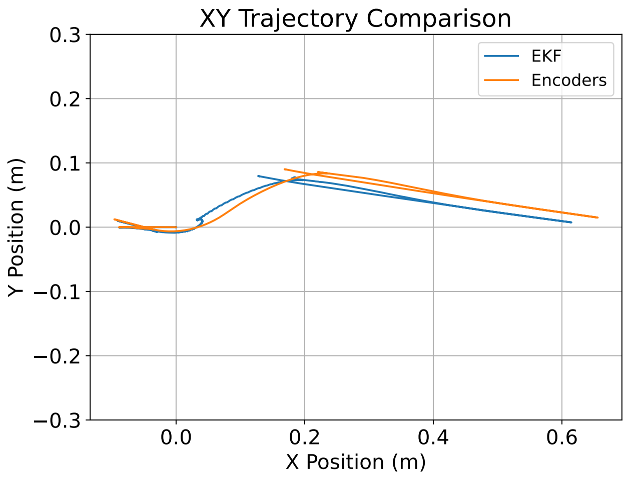


(a) Accelerometer bias estimates

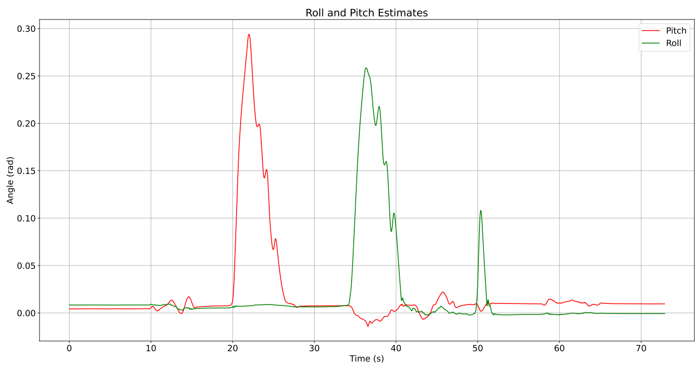


(b) Gyroscope bias estimates

Figure 5-15: IMU bias estimates from the EKF in the first experiment

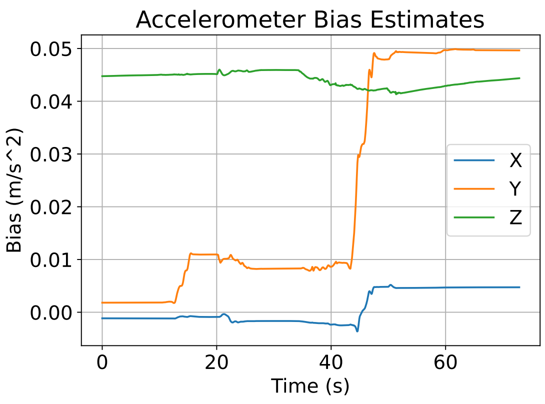


(a) Trajectory estimates on the horizontal plane.

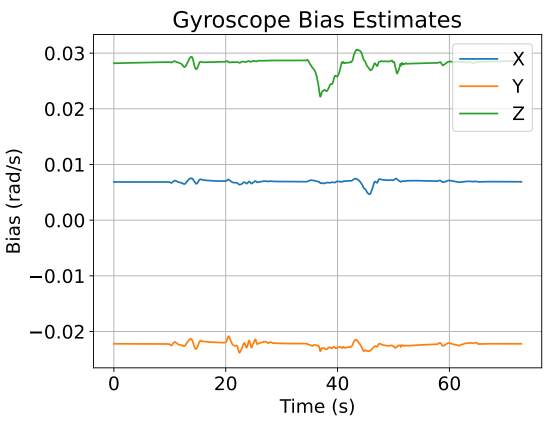


(b) Roll and Pitch Estimates

Figure 5-16: Results of the EKF in the second experiment



(a) Accelerometer bias estimates



(b) Gyroscope bias estimates

Figure 5-17: IMU bias estimates from the EKF in the second experiment

Chapter 6

Conclusions and Recommendations

In this Chapter, we summarize the findings of this work, and propose ideas for future improvements and implementations of this work

To goal of this thesis was to understand how we can fuse information from LIDAR, IMU and wheel encoders to generate odometry with minimal drift while still being feasible for real-time computation. Working towards this goal, we provide answers to the following subquestions.

How does one exploit the LIDAR scanner to reduce drift?

We developed an MHE framework that integrates LIDAR data with wheel encoder information by extracting line primitives from the LIDAR scans. This framework was tested with simulations, which demonstrated promising results; however, its performance was observed to be sensitive to outliers and wheel slips. Similar conclusions were drawn from our analysis of real-world data. We hypothesize that by incorporating a motion distortion model and implementing outlier rejection strategies, this algorithm can achieve accurate pose estimation for the robot on the horizontal plane.

How does one reconcile the 2D and 3D nature of the different sensors?

The EKF we designed successfully fused data from the wheel encoders with the IMU to estimate the robot's trajectory with a satisfactory level of accuracy, even managing to partially mitigate the impact of significant wheel slip on the estimates. Our experimental results also indicate that while the underlying models implicitly assume the robot's movement on a horizontal plane, the filter demonstrates robustness in handling substantial deviations from this assumption.

How can the discussed solutions be implemented efficiently in real-time?

In this work, the EKF was implemented on the onboard computer of the robot as a Robot Operating System (ROS) node, and the odometry we generate was done so in real-time.

We expect, based on implementations of similar works and the existence of efficient solvers, that our MHE scheme will also be feasible for real-time implementations.

6-1 Future Work

We expect that the contributions made in this work enable further work to explore fully tightly-coupled odometry. We propose multiple avenues for future research that are expected to allow the odometry system to achieve the necessary accuracy and robustness to be used in practice.

Firstly, the effect of motion distortion on the LIDAR scans must be modelled and compensated for in line feature extraction module.

Implementation of the line feature extraction module and the MHE in C++, with the use of an efficient solver such as Ceres[30], g20 [31] or gtsam [11] is anticipated to enable real-time application. Moreover, these libraries have built-in functionality to allow the use of *robust* cost functions [32] which allow the optimizer to reject outlier measurements which currently impact the MHE approach.

Finally, tightly-coupled estimation with the IMU, LIDAR and wheel encoder data becomes achievable. To accomplish this, a crucial step involves extending the line features to 3D space as done in [33]. Additionally, the integration of Inertial Measurement Unit (IMU) data necessitates the utilization of preintegration techniques to keep the size of the graph small as outlined in [3] and [34].

We expect that such a tightly-coupled system would be able to realize all the benefits of the different estimators in this work.

Appendix A

Non-Linear Optimization

In this Appendix, we briefly explain how the optimization problem in (3-32) can be solved by iteratively linearising and solving a weighted-linear least squares problem. We also provide the necessary jacobians to do so.

A-1 Linearisation

Linearizing the line measurements and poses around the current estimate, ${}^{\mathrm{W}}\mathbf{l}_j = {}^{\mathrm{W}}\hat{\mathbf{l}}_j + \delta {}^{\mathrm{W}}\mathbf{l}_j$, $\boldsymbol{\xi}^t = \hat{\boldsymbol{\xi}}^t + \delta\boldsymbol{\xi}^t$

$$\begin{aligned} h({}^{\mathrm{W}}\mathbf{l}_j, \boldsymbol{\xi}^t) - z_j^t &\approx h({}^{\mathrm{W}}\hat{\mathbf{l}}_j, \hat{\boldsymbol{\xi}}^t) + H_j^t \delta {}^{\mathrm{W}}\mathbf{l}_j + J\delta\boldsymbol{\xi}^t - z_j^t \\ &= H_j^t \delta {}^{\mathrm{W}}\mathbf{l}_j + J_j^t \delta\boldsymbol{\xi}^t - e_j^t \end{aligned} \tag{A-1}$$

$$\begin{aligned} f(\boldsymbol{\xi}^{t_i}, \boldsymbol{\xi}^{t_{i+1}}) - \mathbf{z}^{t_i,t_{i+1}} &\approx f(\hat{\boldsymbol{\xi}}^{t_1}, \hat{\boldsymbol{\xi}}^{t_2}) + F^{t_i,t_{i+1}}\delta\boldsymbol{\xi}^{t_1} + G^{t_i,t_{i+1}}\delta\boldsymbol{\xi}^{t_2} - \mathbf{z}^{t_i,t_{i+1}} \\ &= F^{t_i,t_{i+1}}\delta\boldsymbol{\xi}^{t_1} + G^{t_i,t_{i+1}}\delta\boldsymbol{\xi}^{t_2} - e^{t_i,t_{i+1}} \end{aligned} \tag{A-2}$$

where $e_j^t = z_j^t - h({}^{\mathrm{W}}\hat{\mathbf{l}}_j, \hat{\boldsymbol{\xi}}^t)$, $e^{t_i,t_{i+1}} = \mathbf{z}^{t_i,t_{i+1}} - f(\hat{\boldsymbol{\xi}}^{t_1}, \hat{\boldsymbol{\xi}}^{t_2})$ are the line measurement prediction error and the wheel encoder measurement error respectively. $F^{t_i,t_{i+1}}$, H_j^t and J_j^t are obtained by evaluating (A-6) and (A-5) at the current estimate. $e^{t_{k-M}} = \mathbf{z}^{t_{k-M}} - \hat{\boldsymbol{\xi}}^{t_{k-M}}$ is the prior error such that $\boldsymbol{\xi}^{t_{k-M}} = \hat{\boldsymbol{\xi}}^{t_{k-M}} + \delta\boldsymbol{\xi}^{t_{k-M}}$

The problem then reduces to a weighted-least squares problem which we can solve recursively

$$\begin{aligned} \delta\mathcal{X}_k^* = \arg\min_{\delta\mathcal{X}_k} \sum_{t\in\mathcal{K}_k}\sum_{p\in\mathcal{L}_k} &\left(\left|\left| H_j^t \delta {}^{\mathrm{W}}\mathbf{l}_j + J_j^t \delta\boldsymbol{\xi}^t - e_j^t \right|\right|^2_{P_j^t} \right) \\ + \sum_{t_1,t_2\in\mathcal{K}_k, t_1\neq t_2} &\left(\left|\left| F^{t_i,t_{i+1}}\delta\boldsymbol{\xi}^{t_1} + G^{t_i,t_{i+1}}\delta\boldsymbol{\xi}^{t_2} - e^{t_i,t_{i+1}} \right|\right|^2_{P^{t_i,t_{i+1}}} \right) \\ &+ \left|\left| \delta\boldsymbol{\xi}^{t_{k-M}} - e^{t_{k-M}} \right|\right|^2_{\Sigma_K^{t_{k-M}}} \end{aligned} \tag{A-3}$$

We can collect all the jacobians, residuals and weight matrices as A, b, Σ respectively, and eliminate the weight matrix as follows:

$$\begin{aligned}
\delta\mathcal{X}_k^* &= \underset{\delta\mathcal{X}_k}{\arg\min}\, ||A\delta\mathcal{X}_k - b||_\Sigma \\
&= \underset{\delta\mathcal{X}_k}{\arg\min}\, (A\delta\mathcal{X}_k - b)^T \Sigma^{-1} (A\delta\mathcal{X}_k - b) \\
&= \underset{\delta\mathcal{X}_k}{\arg\min}\, (A\delta\mathcal{X}_k - b)^T \Sigma^{-1/2}\Sigma^{-1/2} (A\delta\mathcal{X}_k - b) \\
&= \underset{\delta\mathcal{X}_k}{\arg\min}\, \left(\Sigma^{-1/2}A\delta\mathcal{X}_k - \Sigma^{-1/2}b\right)^T \left(\Sigma^{-1/2}A\delta\mathcal{X}_k - \Sigma^{-1/2}b\right) \\
&= \underset{\delta\mathcal{X}_k}{\arg\min}\, \left|\left|\tilde{A}\delta\mathcal{X}_k - \tilde{b}\right|\right|
\end{aligned} \tag{A-4}$$

where $\tilde{A} = \Sigma^{-1/2}A$ and $\tilde{b} = \Sigma^{-1/2}b$

Slightly abusing our notation to express $\mathcal{X}_K$ as vector, we can solve this iteratively and update $\mathcal{X}_k$ using the Levenberg-Marquardt method as

$$\mathcal{X}_k \leftarrow \mathcal{X}_k + \left(\lambda I + \tilde{A}^T\tilde{A}\right)^{-1} \tilde{A}^T\tilde{b}$$

where λ is a hyperparameter used to control the convergence behaviour.

The uncertainty of the resulting state is given by the inverse of the hessian, which can be approximated as

$$P_{\mathcal{X}_k} = \left(\tilde{A}^T\tilde{A}\right)^{-1}$$

. The uncertainty associated with the last state in the window $\boldsymbol{\xi}_k$ can then be read off the corresponding block of $P_{\mathcal{X}_k}$, which depends on the ordering of the variables in the vector.

A-2 Jacobians

The line factor in (3-22) can be linearised around an estimate ${}^{\mathrm{W}}\hat{\mathbf{l}}$, $\hat{\boldsymbol{\xi}}$ by using a first order approximation. The relevant jacobians can be evaluated using

$$\begin{aligned}
\left.\frac{\partial h({}^{\mathrm{W}}\mathbf{l}, \boldsymbol{\xi})}{\partial {}^{\mathrm{W}}\mathbf{l}}\right|_{{}^{\mathrm{W}}\hat{\mathbf{l}}, \hat{\boldsymbol{\xi}}} &= \begin{bmatrix} 1 & \hat{x}\sin\hat{\alpha} - \hat{y}\cos\hat{\alpha} \\ 0 & 1 \end{bmatrix} \\
\left.\frac{\partial h({}^{\mathrm{W}}\mathbf{l}, \boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right|_{{}^{\mathrm{W}}\hat{\mathbf{l}}, \hat{\boldsymbol{\xi}}} &= \begin{bmatrix} -\cos\hat{\alpha} & -\sin\hat{\alpha} & 0 \\ 0 & 0 & -1 \end{bmatrix}
\end{aligned} \tag{A-5}$$

Similarly, the jacobians for the pose factors in (3-25) are given by

$$\begin{aligned}
\left.\frac{\partial f\left(\boldsymbol{\xi}^{t_1},\boldsymbol{\xi}^{t_2}\right)}{\partial\boldsymbol{\xi}^{t_1}}\right|_{\hat{\boldsymbol{\xi}}^{t_1},\hat{\boldsymbol{\xi}}^{t_2}} &= \begin{bmatrix} -\cos\gamma^{t_1} & -\sin\hat{\gamma}^{t_1} & -\left(x^{t_2}-x^{t_1}\right)\sin\hat{\gamma}^{t_1}+\left(y^{t_2}-y^{t_1}\right)\cos\hat{\gamma}^{t_1} \\ \sin\hat{\gamma}^{t_1} & -\cos\hat{\gamma}^{t_1} & -\left(x^{t_2}-x^{t_1}\right)\cos\hat{\gamma}^{t_1}-\left(y^{t_2}-y^{t_1}\right)\sin\hat{\gamma}^{t_1} \\ 0 & 0 & 1 \end{bmatrix} \\
\left.\frac{\partial f\left(\boldsymbol{\xi}^{t_1},\boldsymbol{\xi}^{t_2}\right)}{\partial\boldsymbol{\xi}^{t_2}}\right|_{\hat{\boldsymbol{\xi}}^{t_1},\hat{\boldsymbol{\xi}}^{t_2}} &= \begin{bmatrix} \cos\hat{\gamma}^{t_1} & \sin\hat{\gamma}^{t_1} & 0 \\ -\sin\hat{\gamma}^{t_1} & \cos\hat{\gamma}^{t_1} & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{A-6}$$

Appendix B

Noise covariances and hyperparameters

Here we present the tuning for the filters and estimators used in all the real-world experiments:

B-1 2D Pose Estimation

Table B-1: Feature Extractor and LIDAR noise settings

c_{thresh}	θ_{max}	d_{min}	σ_d	σ_ϕ	λ
10^{-2}	10^{-2}	10^{-1}	10^{-2}	10^{-4}	10^{-2}

Table B-1 provides the thresholds for the feature extraction scheme and the LIDAR noise parameters.

The covariance of the noise affecting the encoder factor was tuned manually, and set to $Q_w = 10^{-4} \times I_3$.

B-2 Full 3D Motion Estimation

The IMU noise parameters are set according to the datasheet of the IMU [35] as

$$\begin{aligned}\sqrt{Q_{\eta,acc}} &= \quad 1.39256 \times 10^{-3} I_3 \\ \sqrt{Q_{\eta,gyro}} &= 1.309 \times 10^{-4} I_3 \\ \sqrt{Q_{\epsilon,acc}} &= \quad 6.86 \times 10^{-5} I_3 \\ \sqrt{Q_{\epsilon,gyro}} &= 1.9896 \times 10^{-5} I_3\end{aligned} \tag{B-1}$$

The noise parameters for the measurement model were tuned manually, and set to

$$\sqrt{Q_h} = \begin{bmatrix} 1\times10^{-2} & 0 & 0 & 0 \\ 0 & 1\times10^{-1} & 0 & 0 \\ 0 & 0 & 1\times10^{-4} & 0 \\ 0 & 0 & 0 & 1\times10^{-2} \end{bmatrix} \tag{B-2}$$

The initial estimate for the state was set to $\mathbf{x}^{t_0} = 0_{15\times1}$, and the associated uncertainty was set to

$$\text{diag}(\sqrt{P_0}) = \begin{bmatrix} 10^{-15} \\ 10^{-15} \\ 10^{-15} \\ 10^{-15} \\ 10^{-15} \\ 10^{-15} \\ 10^{-3} \\ 10^{-3} \\ 10^{-15} \\ 10^{-1} \\ 10^{-1} \\ 10^{-1} \\ 10^{-1} \\ 10^{-1} \\ 10^{-1} \end{bmatrix} \tag{B-3}$$

where the diag operator extracts the diagonal elements of a matrix. The other elements were all set to zero. Notice the large uncertainty on the last 6 states, which correspond to the bias estimates. These uncertainties are larger than the rest since we have arbitrarily initialised the filter with an estimate of 0 for these states.

Bibliography

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Transactions on Robotics*, vol. 32, pp. 1309–1332, 12 2016.

[2] J. Zhang and S. Singh, “LOAM: Lidar Odometry and Mapping in Real-time,” in *Robotics: Science and Systems X*, Robotics: Science and Systems Foundation, 7 2014.

[3] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, “On-Manifold Preintegration for Real-Time Visual–Inertial Odometry,” *IEEE Transactions on Robotics*, vol. 33, pp. 1–21, 2 2017.

[4] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, “FAST-LIO2: Fast Direct LiDAR-Inertial Odometry,” *IEEE Transactions on Robotics*, vol. 38, pp. 2053–2073, 8 2022.

[5] T. Qin, P. Li, and S. Shen, “VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator,” *IEEE Transactions on Robotics*, vol. 34, pp. 1004–1020, 8 2018.

[6] D. Wisth, M. Camurri, and M. Fallon, “VILENS: Visual, Inertial, Lidar, and Leg Odometry for All-Terrain Legged Robots,” *IEEE Transactions on Robotics*, vol. 39, pp. 309–326, 2 2023.

[7] X. Zuo, P. Geneva, W. Lee, Y. Liu, and G. Huang, “LIC-Fusion: LiDAR-Inertial-Camera Odometry,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5848–5854, IEEE, 11 2019.

[8] D. Wisth, M. Camurri, S. Das, and M. Fallon, “Unified Multi-Modal Landmark Tracking for Tightly Coupled Lidar-Visual-Inertial Odometry,” *IEEE Robotics and Automation Letters*, vol. 6, pp. 1004–1011, 4 2021.

[9] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5135–5142, IEEE, 10 2020.

[10] X. Zuo, Y. Yang, P. Geneva, J. Lv, Y. Liu, G. Huang, and M. Pollefeys, "LIC-Fusion 2.0: LiDAR-Inertial-Camera Odometry with Sliding-Window Plane-Feature Tracking," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5112–5119, IEEE, 10 2020.

[11] F. Dellaert and M. Kaess, "Factor Graphs for Robot Perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[12] S. Pfister, S. Roumeliotis, and J. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 1, pp. 1304–1311, IEEE, 2003.

[13] S. T. Pfister, "Weighted line fitting and merging," *California Institute of Technology, Tech. Rep*, 2002.

[14] "REP 103 – Standard Units of Measure and Coordinate Conventions (ROS.org)."

[15] M. Kok, J. D. Hol, and T. B. Schön, "Using Inertial Sensors for Position and Orientation Estimation," *Foundations and Trends® in Signal Processing*, vol. 11, no. 1-2, pp. 1–153, 2017.

[16] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, pp. 11–15, 1 1972.

[17] M. A. Fischler and R. C. Bolles, "Random sample consensus," *Communications of the ACM*, vol. 24, pp. 381–395, 6 1981.

[18] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, pp. 52–57, 3 2002.

[19] C. Kilic, J. N. Gross, N. Ohi, R. Watson, J. Strader, T. Swiger, S. Harper, and Y. Gu, "Improved Planetary Rover Inertial Navigation and Wheel Odometry Performance through Periodic Use of Zero-Type Constraints," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 552–559, IEEE, 11 2019.

[20] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle Adjustment — A Modern Synthesis," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1883, pp. 298–372, Springer Verlag, 2000.

[21] F. Girrbach, *Moving horizon estimation for inertial motion tracking:: algorithms and industrial applications.* PhD thesis, Dissertation, Universität Freiburg, 2021, 2021.

[22] H. Strasdat, J. Montiel, and A. J. Davison, "Visual SLAM: Why filter?," *Image and Vision Computing*, vol. 30, pp. 65–77, 2 2012.

[23] D. Wisth, M. Camurri, and M. Fallon, "Robust Legged Robot State Estimation Using Factor Graph Optimization," *IEEE Robotics and Automation Letters*, vol. 4, pp. 4507–4514, 10 2019.

[24] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental Smoothing and Mapping," *IEEE Transactions on Robotics*, vol. 24, pp. 1365–1378, 12 2008.

[25] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research*, vol. 31, pp. 216–235, 2 2012.

[26] M. Brossard, A. Barrau, and S. Bonnabel, "Ai-imu dead-reckoning," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585–595, 2020.

[27] K. Tracy, "A square-root kalman filter using only qr decompositions," 2022.

[28] A. Al-Nuaimi, W. Lopes, P. Zeller, A. Garcea, C. Lopes, and E. Steinbach, "Analyzing LiDAR scan skewing and its impact on scan matching," in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8, IEEE, 10 2016.

[29] P. J. Antsaklis and A. N. Michel, *Linear Systems.* Birkhäuser Boston, 2006.

[30] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 5 2022.

[31] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, IEEE, 5 2011.

[32] K. MacTavish and T. D. Barfoot, "At all Costs: A Comparison of Robust Cost Functions for Camera Correspondence Outliers," in *2015 12th Conference on Computer and Robot Vision*, pp. 62–69, IEEE, 6 2015.

[33] A. J. Trevor, J. G. Rogers, and H. I. Christensen, "Planar surface SLAM with 3D and 2D sensors," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3041–3048, 2012.

[34] J. Henawy, Z. Li, W.-Y. Yau, and G. Seet, "Accurate IMU Factor Using Switched Linear Systems for VIO," *IEEE Transactions on Industrial Electronics*, vol. 68, pp. 7199–7208, 8 2021.

[35] TDK InvenSense, *IIM-46234 and IIM-46230 Datasheet*, 10 2022. 1.3.

Glossary

List of Acronyms

IMU	Inertial Measurement Unit
LIDAR	Laser Imaging Detection and Ranging
SLAM	Simultaneous Localization and Mapping
RGB-D	Red-Green-Blue-Depth
DoF	Degrees-of-Freedom
RANSAC	RANdom SAmple Consensus
ODR	Output Data Rate
MHE	Moving Horizon Estimation
MAP	Maximum A-Posteriori
RMSE	Root Mean Square Error
MSC-KF	Multi State Constraint-Kalman Filter
VIO	Visual-Inertial Odometry
EKF	Extended Kalman Filter
ROS	Robot Operating System