NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Aleksander Nysted Elvebakk

Fusion of Visual-Inertial Odometry and GNSS-Based Positioning for Localization of Autonomous Ferries

An Evaluation of Sensor Combinations and State-of-the-Art Methods

Master's thesis in Cybernetics and Robotics
Supervisor: Kostas Alexis
Co-supervisor: Erik Wilthil
June 2022

Aleksander Nysted Elvebakk

Fusion of Visual-Inertial Odometry and GNSS-Based Positioning for Localization of Autonomous Ferries

An Evaluation of Sensor Combinations and State-of-the-Art Methods

Master's thesis in Cybernetics and Robotics
Supervisor: Kostas Alexis
Co-supervisor: Erik Wilthil
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Abstract

Self-localization for autonomous systems is essential for unsupervised operations, and for maneuvering drones, estimation of *Visual-Inertial Odometry (VIO)* is a viable option for this. In this thesis, it was found that using VIO estimation for self-localization of an autonomous ferry sailing in an urban environment is a much harder task than the aforementioned case of drone flight. This was due to distant landmarks and slow motions. It was deemed necessary to have a high resolution for cameras and a robust initialization procedure for depth-estimation of features, if the VIO is to be accurate. However, despite showing poor standalone performance, VIO was shown to be very useful when used in tandem with a Global Navigation Satellite System (GNSS) such as GPS. This made it possible to localize the ship in locations where it previously was impossible, e.g. when driving under a bridge. Due to this, it was deemed that camera-based localization adds value also in a maritime setting.

Sammendrag

Selvlokalisering for autonome systemer er helt essensielt for å kunne operere uten tilsyn. For droner og andre høyhastighetsfartøy så er visuell-treghetsbasert odometri (Visual-Inertial Odometry) et godt alternativ. I denne mastergraden ble det oppdaget at å bruke dette som et lokaliseringsverktøy for trege, autonome ferger, seilende i bymiljø var en større utfordring enn for droner. Det var flere grunner til dette, men et stort problem var at landemerkene som ble avbildet var alle langt unna båten. Det ble anslått som nødvendig å ha høyere oppløsning på videokameraene, samt at estimatene for dybden til de avbildede landemerkene måtte ha en mer robust initialisering. Likevel, på tross av dårlig lokaliseringsnøyaktighet i isolasjon, viste det seg at å bruke visuell-treghetsbasert odometri for lokalisering i tandem med GPS var veldig nyttig. Dette gjorde at det var mulig å lokalisere skipet, selv på steder som GPSen hadde problemer. Det ble derfor konkludert med at kamera har lokaliseringsmessig nytteverdi, også på en liten autonom ferge.

Preface

This masters thesis has been written at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU) during the spring of 2022. Parts of the literature review and background theory was developed in the specialization project the fall of 2021. It is the culmination of my five years in the study program Cybernetics and Robotics, and marks the end of my time as a student.

Acknowledgements

I would like to thank my co-supervisor Dr. Erik Wilthil for help with gathering data, solving technical problems, giving feedback on the thesis and in general being a great resource throughout this spring. I would also like to thank my supervisor Professor Konstantinos Alexis for his deep professional knowledge, and being able to give advice when needed. My deepest thanks also goes to Trym Synnevåg for correcting and improving this thesis, it really was *very* helpful.

I would, however, also like to express my deepest thanks to all the great people I have had the pleasure of becoming friends with throughout my time as a student. To mention a few, this includes the wonderful people from the Chillout Lounge ISFiT19, MØKom, VDD-Storstyret UKA19, Welfare Board ISFiT21, Korsgata, MDG6 and many more. Without you these years would never have been possible, nor as enjoyable as they were.

Lastly I would also like to thank my girlfriend Nikoline, as well as my family for their endless support. It has been invaluable during these years, and I would not be where I am today without it.

Contents

Figures

Tables

List of Algorithms

Acronyms

ACF Autocorrelation Function. 15, 16

APE Absolute Positioning Error. xiii, 63, 79

BRISK Binary Robust Invariant Scalable Keypoints. 43

CDF Cumulative Distribution Function. 10

DSO Direct Sparse Odometry. 43

e.g. Exempli Gratia. i, 19, 25, 39, 63

EKF Extended Kalman Filter. 44, 60

GLONASS Global Navigation Satellite System. 5

GNSS Global Navigation Satellite System. i, 2–7, 46, 50, 51, 56, 60, 63, 64, 81, 83–85

GPS Global Positioning System. i, xii, 2, 5, 50

i.e. id est. 2, 6, 15

IID Independent and identically distributed. 15

IMU Inertial Measurement Unit. xi, 3, 6, 7, 31, 32, 45, 51–54, 57, 60, 64, 80, 84, 85

INS Inertial Navigation System. 6, 60, 81, 83

KF Kalman Filter. xvii, 44

MAP Maximum A Posteriori. xi, 36–38

ML Maximum Likelihood. 36, 38, 40

OKVIS Open Keyframe-based Visual-Inertial SLAM. 43, 44

PDF Probability Density Function. 10–12

PMF Point Mass Function. 10

RMSE Root Mean Square Error. xiii, 79

ROVIO Robust Visual-Inertial Odometry. 44

RPE Relative Positioning Error. xii, xiii, 63, 67, 77–79

RTK Real-Time Kinematic. 5, 60

SLAM Simultaneous Localization and Mapping. xvii, 7, 43

SSE Summed Squared Error. 43

SVO Semi-direct Visual Odometry. 44, 46, 51, 56, 58, 64, 67, 80, 81, 85

UAV Unmanned Aerial Vehicle. 45

VINS Visual-Inertial System. 44

VIO Visual-Inertial Odometry. i, iii, xii, 7, 16, 17, 19, 39, 44, 46, 48–50, 52, 60, 63, 80, 81, 83–85

VO Visual Odometry. 7

VSLAM Visual Simultaneous Localization and Mapping. 7, 60

w.r.t with respect to. 44, 46, 52

Chapter 1

Introduction

“Intelligent unmanned autonomous systems are systems that are man-made and capable of carrying out operations or management by means of advanced technologies without human intervention“[1]. This was once a topic reserved for highly advanced weaponry and space exploration systems. However, these systems have matured both in terms of robustness and costs in the last two decades. Today, autonomous systems can be found in a wide variety of industries such as private and public transport, inspections of industrial facilities, and mowing of lawns[2][3][4][5].

1.1 Zeabuz: Autonomous Waterborn Urban Mobility

Zeabuz is a company which specializes in autonomous systems, especially autonomous ships[6]. They aim to make small autonomous ferries which can be used for public and private transport in rivers, fjords and canals. The technology originates from years of reasearch at the Department of Engineering Cybernetics and the Department of Marine Technology at NTNU[7][8][9]. See Figure 1.1 for concept art.

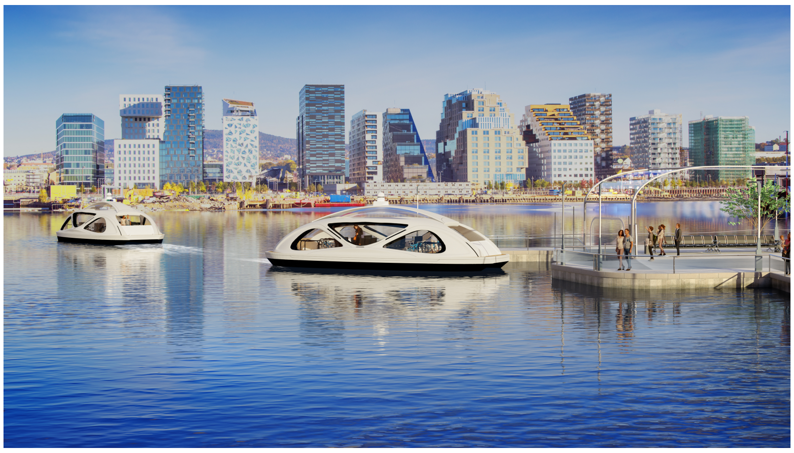

Figure 1.1: Concept of operations set in Bjørvika, Oslo

1.2 Motivation

An autonomous ferry which is to sail in rivers and canals needs to constantly make choices regarding how to move, stop and most importantly, avoid collisions if necessary. This is illustrated in Figure 1.2. Making these types of choices in an informed manner requires that the ferry is aware of the situation and its surroundings. Among other things, this consists of knowing the location of surrounding ships and humans, the distance from the dock, and the location of the autonomous ferry itself.

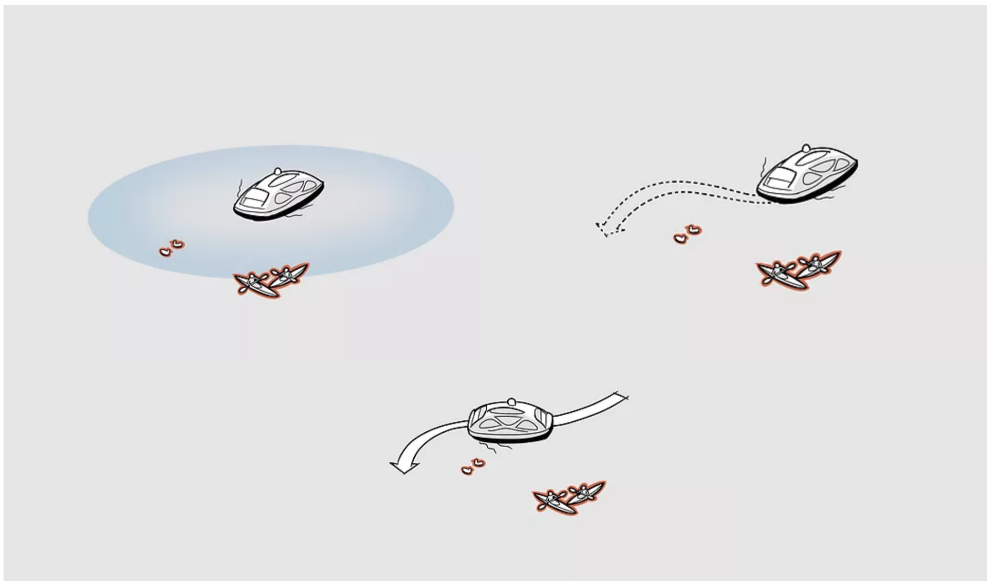

Figure 1.2: The process a ferry goes through by perceiving, planning and acting, when sailing autonomously. Figure is courtesy of Zeabuz [6]

1.2.1 Pose and egomotion

This latter piece of information, i.e. the position, orientation and movement of the autonomous ferry, is crucial to carrying out any autonomous operations. The position and orientation are referred to as the *pose* of the ship, and the movement is referred to as the *egomotion*. In open sea conditions, the absolute position of a ship can be estimated using a Global Navigation Satellite System (GNSS), such as Global Positioning System (GPS). However, GNSS reception in urban environments might be reduced due to echo from buildings, or completely terminated for longer periods when sailing under bridges or through tunnels. GNSS positioning will therefore not suffice for an autonomous urban ferry, and backup solutions must be in order.

1.2.2 Visual-inertial navigation

A camera setup and an Inertial Measurement Unit (IMU) is sensor pairing which is popular for this purpose in the robotics community. An IMU is a small sensor which measures its own acceleration and angular velocity. For the camera setup, either a single monocular camera or a stereo camera setup is used. This combination is popular due to its low cost and small size, and how it fits well on a drone or a small robot. There has been remarkable progress in using cameras and an IMU to estimate motion in the last ten years, and for drones, it serves as a very viable option in the absence of a GNSS signal. Using a camera together with an IMU to estimate motion is referred to as visual-inertial navigation.

1.2.3 Motion and maneuvers

When a drone is flying, it is often maneuvering and moving quickly. Because of this, all degrees of freedom might experience forces such as centrifugal force and gravity. This makes the measurements of the acceleration and angular velocity from the IMU informative of how the drone flies. An autonomous, small ferry will move slowly, and therefore the forces such as the centrifugal forces when turning will be experienced to a much lesser extent. This is good for passengers, but bad for information, as the sensors will carry little information about how the ship truly moves. Due to these differences, the state-of-the-art methods used for visual-inertial estimation in the robotics community will not necessarily be applicable to autonomous ships, and this need to be investigated.

1.3 Milliampere 2

Milliampere 2 is a small electric ship made by NTNU, aiming to serve as a prototype for small, urban ferries. It is used both for researching new technology, as well as for demonstrating and testing existing technology. Among other sensors, the ferry has a stereo camera setup, an IMU and a GNSS receiver, and will thus serve as the test platform to gather data. With two cameras and an IMU, there are three different sensor combinations which can utilized for visual-inertial navigation:

- An IMU and stereo cameras
- An IMU and a monocular camera
- Stereo cameras only

1.4 Problem Definition

The aim of this thesis is two-fold. Firstly, it aims to quantify how state-of-the-art methods for visual-inertial estimation works in a maritime environment. Secondly, it also aims to map which of the sensor combinations from Section 1.3 works best for visual-inertial navigation, first as an isolated case, and then in combination with GNSS data.

1.5 Structure of the report

In Chapter 2 relevant terms to the task and subject are introduced. The theory deemed necessary for understanding the visual-inertial estimation techniques is reviewed in Chapter 3, and in Chapter 4 related work is discussed. Chapter 5 and Chapter 6 reviews the software methods used for sensor fusion and in Chapter 7 the hardware used to collect data is presented. In Chapter 8 the estimation results are presented and discussed. Finally, in Chapter 9 the findings are summarized and concluding remarks are left.

Chapter 2

Relevant Sensors, Concepts and Terms

“Sensor fusion is the practice of combining sensor data from disparate sources such that the resulting information has less uncertainty than would be possible when these sources were used individually“, formulated in [10]. This thesis will focus on sensor fusion in the sense of using data from different types of sensors, or *modalities*, to estimate a common state of a system. This is referred to as *state estimation*. This chapter will serve as a verbal introduction to the sensors, along with terms and concepts relevant to this thesis.

2.1 Global Navigation Satellite System

As mentioned in Chapter 1, the absolute position of a ship, or anything else, can be obtained using a Global Navigation Satellite System (GNSS). This position is measured in *latitude, longitude* and *altitude* and is calculated using signals from several satellites in orbit[11]. Examples of GNSS are Global Positioning System (GPS) and Global Navigation Satellite System (GLONASS).

2.1.1 Accuracy

One issue with GNSS measurements is that the signals can be noisy. Even though the position is absolute, they generally have an accuracy of 5-10 meters [12], which is not nearly enough for autonomous operations. However, there are solutions to this problem. Using Real-Time Kinematic (RTK) positioning, one compares the satellite signals received by a base station whose position is known, with the signals received at the ship. This will increase the accuracy of the position from a GNSS to centimeter precision[13].

2.1.2 Signal reception

A secondary problem, is that the frequency of these positional updates can become low, especially when moving in areas with poor satellite reception. In these cases, there can be several seconds between positional updates, or the connection can be completely broken. Therefore, to estimate the position when moving in these types of areas, other techniques must be used.

2.2 Odometry

The process of using data from motion sensors to estimate the position relative to an initial location, i.e. the last known positional measurement from the GNSS, is referred to as odometry [14]. In early applications for cars, odometry was obtained by integrating wheel encoders. The estimate obtained from this wheel odometry quickly drifted due to wheel slippage and other noise, making the estimate unusable after only a few meters [15]. For high-speed ships, maneuvering at sea might require a higher frequency of positional updates than the GNSS can produce. A widely used solution to this problem is using an Inertial Navigation System (INS). This system numerically integrates measurements from an Inertial Measurement Unit (IMU) as odometry between GNSS updates.

2.3 Inertial Measurement Unit

An Inertial Measurement Unit (IMU) is a commonly used sensor due to its small size and low cost. The sensor consists of three accelerometers and three gyroscopes. The accelerometers measure linear acceleration along the axes connected to its body, and the gyroscopes measure angular velocity around these axes. The rotational degrees of freedom associated with the different angles, as well as a general illustration can be seen in Figure 2.1.

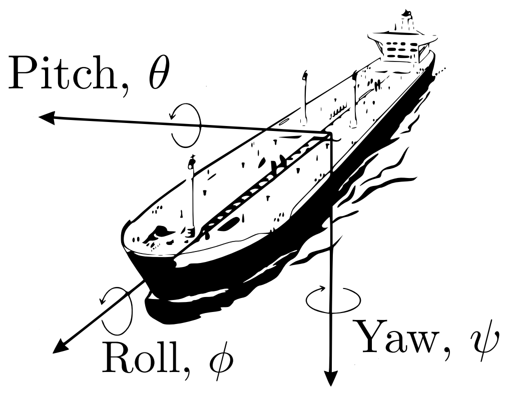

Figure 2.1: The axes from which an IMU measures data. Illustration is highly inspired by illustration in [16]

2.3.1 Measurement noise

An IMU generally gives sensor readings at a very high rate, typically 100-200 Hz. These readings can be integrated to directly perform pose odometry. However, as these readings are extremely noisy, the uncertainty associated with the estimates will quickly become too noisy to trust. Because of this, navigation for an extended time without GNSS measurements, requires more sensors than an IMU.

2.4 Visual-Inertial Odometry

A sensor which is often used together with an IMU to measure motion is a video camera. By tracking how the environment appears to move through a sequence of images, it is possible to simultaneously reconstruct both the motion of the ship relative to the world and a three-dimensional map of the environment. This is referred to as Visual Odometry. When this information of motion is used in tandem with an IMU to estimate odometry, it is referred to as Visual-Inertial Odometry (VIO).

2.4.1 Loop closure

The model of the environment created by the VIO can be used to recognize when one has returned to a previously visited location. This is referred to as *loop closure*. The difference between odometry when using loop closure and not is illustrated in Figure 2.2. The loop closure is performed as a measure to eliminate drift. However, when using Visual-Inertial Odometry for navigation, the inclusion of GNSS measurement will always perform better for this purpose.

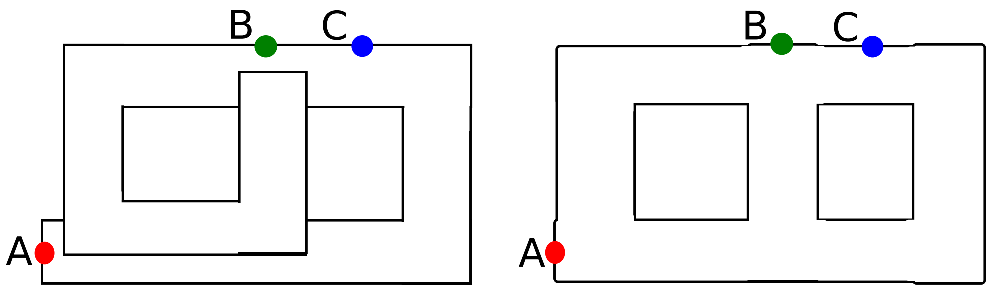

Figure 2.2: Odometry to the left makes the path look like it is in a long hallway. To the right the loop closure recognizes that one has returned to a previously visited position. Illustration in [15].

VO, VIO and VSLAM

Performing VIO with loop closure can often be referred to as Visual Simultaneous Localization and Mapping (VSLAM) or Simultaneous Localization and Mapping (SLAM).

Chapter 3

Background Theory

This chapter aims to first give an introduction to the theory needed for understanding the algorithms used for visual and visual-inertial odometry, thereafter provide an introduction to how the algorithms generally work. This section was partly developed in the specialization project, in the fall of 2021.

3.1 Random Variables and the Gaussian Distribution

As sensor fusion intrinsically is about inference using data which has uncertainty involved, this section will contain a short introduction to random variables and the Gaussian distribution.

3.1.1 Random Variables and Probability Distributions

A random variable does not have a set value, but is considered in terms of its probability distribution function. A scalar random variable X being distributed according to $p(x)$ is written as Equation (3.1)

$$X \sim p(x) \tag{3.1}$$

3.1.2 Discrete and continous random variables

Furthermore, there is a distinction between discrete and continuous random variables. The position of the ship is a continuous quantity, whereas the value a dice will take when thrown, is a discrete random value. The distinction comes from the outcome space of the random variable. If it can take every infinitesimally different value, such as the position in the world, it is said to be continuous. If the random variable only can be realized as certain distinct values, as with the dice throw, it is discrete.

Discrete random variables

In the case of a single dice throw, the dice can take on six different realizations, each with equal probability. To quantify probability for different events, especially for discrete random variables, the Point Mass Function (PMF) is applied. A discrete random variable X, with distribution according to the PMF p(x) is described by (3.2)

$$X \sim p(x) = \text{Probability}\{X = x\} \in \mathbb{R} \tag{3.2}$$

Continuous random variables

For the continuous-valued random variable, e.g. in the case of the position of a ship, it is no longer possible to talk about point probabilities, that is Probability{X=x}. This is due to the point probability of a random variable being zero. Therefore, for the continuous random variables, the probabilities are contained in the Cumulative Distribution Function (CDF) P(x) (3.3)

$$\text{Probability}\{X < x\} = P(x) \tag{3.3a}$$

$$\text{Probability}\{x < X\} = 1 - P(x) \tag{3.3b}$$

$$\text{Probability}\{x_1 < X < x_2\} = P(x_2) - P(x_1) \tag{3.3c}$$

By taking the derivative of the CDF, the Probability Density Function (PDF) p(x) is acquired.

$$p(x) = \frac{\partial P(x)}{\partial x}$$

The cumulative probabilities can now be expressed in terms of an integral over the PDF

$$\text{Probability}\{X < x\} = \int_{-\infty}^{x} p(x)dx$$

$$\text{Probability}\{X > x\} = \int_{x}^{\infty} p(x)dx$$

$$\text{Probability}\{x_1 < X < x_2\} = \int_{x_1}^{x_2} p(x)dx$$

The CDF may not always exist in closed-form, so modeling it as an integral over a PDF is often necessary to get the probabilities.

3.1.3 Independence, conditionality and Bayes rule

Two random variables X and Y are independent if their joint distribution can be written as the product of their marginal distributions. This is mathematically formulated in Equation (3.4).

$$X \sim p_1(x) \tag{3.4a}$$
$$Y \sim p_2(y) \tag{3.4b}$$
$$\text{X and Y independent: } \implies p(X,Y) = p_1(X)p_2(Y) \tag{3.4c}$$

The conditional distribution is defined as

$$p(X|Y) = \frac{p(X,Y)}{p(Y)}$$

and if they are independent, this reduces to just the marginal of X. Bayes rule defines how to switch conditionality

$$p(X|Y) = \frac{p(Y|X)p(X)}{p(Y)}$$

3.1.4 The Univariate Gaussian Probability Distribtuion

One of the most known probability distributions is the Gaussian distribution. The Gaussian distribution is also referred to as the normal distribution and its PDF is often called the bell curve due to its shape. The Gaussian PDF is defined as in Equation (3.5)

$$p(x) = \mathcal{N}(x; \bar{x}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(\frac{(x-\bar{x})^2}{2\sigma^2}\right) \tag{3.5}$$

In Equation (3.5) $\bar{x}$ is the expectation and σ^2 is the variance. The expectation of a random variable is the realization that one expects the variable to take and is defined as a weighted average over the PDF

$$\bar{x} = E[X] = \int xp(x)dx$$

The variance is a measure of the uncertainty of the variable. The variance is defined as

$$\sigma^2 = Var[X] = E[(X - E[X])^2] = E[X^2] - E[X]^2$$

The square root of the variance is the standard deviation

$$\sigma = Std[X] = \sqrt{Var[X]}$$

The Gaussian distribution, with its expectation and standard deviation is illustrated in Figure 3.1.

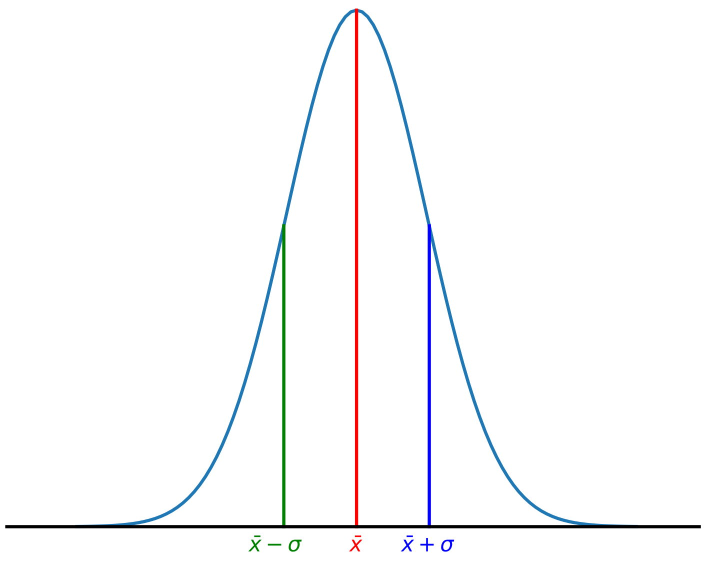

Figure 3.1: The Gaussian probability distribution

Linearity and independence

Given two independent random variables X and Y

$$X \sim \mathcal{N}(\bar{x}, \sigma_x^2)$$
$$Y \sim \mathcal{N}(\bar{y}, \sigma_y^2)$$

and two scalars $a \in \mathbb{R}$ and $b \in \mathbb{R}$. Then the linear combination of the two random variables has the following distribution

$$aX + bY \sim \mathcal{N}(\mu, \sigma^2)$$
$$\mu = a\bar{x} + b\bar{y}$$
$$\sigma^2 = (a\sigma_x)^2 + (b\sigma_y)^2$$

3.1.5 The multivariate Gaussian distribution

In odometry, there are usually several variables or states to be estimated. To describe uncertainty in a multivariate system, one can use a multivariate probability distribution and the Gaussian distribution is easily extended to this case. The random vector **X** is distributed according to the multivariate Gaussian with the PDF

$$\mathbf{X} \sim \mathcal{N}(\bar{\mathbf{x}}, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\bar{\mathbf{x}})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\bar{\mathbf{x}})\right) \tag{3.6}$$

In Equation (3.6) $\bar{\mathbf{x}}$ is a vector of expectations for the random variables contained in the vector $\mathbf{X}$. $\mathbf{\Sigma}$ is the covariance matrix. This matrix both holds the variance of the random variables contained in $\mathbf{x}$ along its diagonal, but also the covariances between the different variables.

$$\underbrace{\begin{bmatrix} X \\ Y \end{bmatrix}}_{\mathbf{X}} \sim \mathcal{N}\left(\underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{\mathbf{x}}; \underbrace{\begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix}}_{\bar{\mathbf{x}}}, \underbrace{\begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 \end{bmatrix}}_{\mathbf{\Sigma}}\right) \tag{3.7}$$

For a multivariate Gaussian with two variables, the height of the graph can be visualized with height curves. This is illustrated in Figure 3.2. Illustration is highly inspired by illustration in [10].

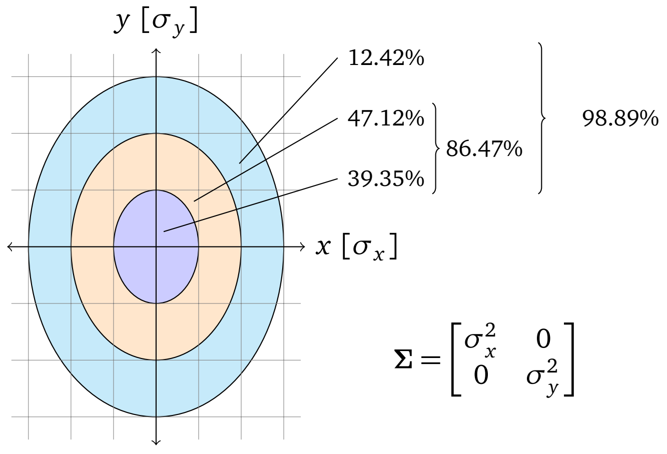
$y\ [\sigma_y]$
12.42%
47.12%
39.35%
86.47%
98.89%
$x\ [\sigma_x]$
$\mathbf{\Sigma} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$

Figure 3.2: Gaussian distribution with two variables. Probability mass contained in height curves shown.

Independence

Consider the random vectors $\mathbf{X} \in \mathbb{R}^n$ and $\mathbf{Y} \in \mathbb{R}^m$ which are distributed as in Equation (3.8)

$$\mathbf{X} \sim \mathcal{N}(\bar{\mathbf{x}}, \mathbf{\Sigma}_x) \tag{3.8a}$$
$$\mathbf{Y} \sim \mathcal{N}(\bar{\mathbf{y}}, \mathbf{\Sigma}_y) \tag{3.8b}$$

they are independent if and only if the following holds[10]:

$$\begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \bar{\mathbf{x}} \\ \bar{\mathbf{y}} \end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_y \end{bmatrix}\right) \tag{3.9}$$

Linearity

Consider the random vector $\mathbf{X} \in \mathbb{R}^n$ which is randomly distributed according to Gaussian distribution $\mathcal{N}(\bar{\mathbf{x}}, \mathbf{\Sigma})$. If $\mathbf{X}$ undergoes the linear transformation $\mathbf{Y} = \mathbf{F}\mathbf{X} +$

b, the random variable **Y** is then distributed according to

$$\mathbf{Y} \sim \mathcal{N}\left(\mathbf{F}\bar{\mathbf{x}} + \mathbf{b}, \mathbf{F}\boldsymbol{\Sigma}\mathbf{F}^\top\right)$$

Marginalization and conditioning

The random vector $\begin{bmatrix}\mathbf{X} & \mathbf{Y}\end{bmatrix}^\top$ is distributed according to

$$\begin{bmatrix}\mathbf{X}\\ \mathbf{Y}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\bar{\mathbf{x}}\\ \bar{\mathbf{y}}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_x & \boldsymbol{\Sigma}_{xy}\\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_y\end{bmatrix}\right)$$

The marginal distribution of **Y** is then given by

$$\mathbf{Y} \sim \mathcal{N}(\bar{\mathbf{y}}, \boldsymbol{\Sigma}_y)$$

and the distribution of **X** conditional on **Y**, $p(\mathbf{X}|\mathbf{Y} = \mathbf{y})$ is given by

$$\begin{aligned} p(\mathbf{X}|\mathbf{Y} = \mathbf{y}) &= \mathcal{N}\left(\bar{\mathbf{x}}_{x|y}, \boldsymbol{\Sigma}_{x|y}\right) \\ \bar{\mathbf{x}}_{x|y} &= \bar{\mathbf{x}} + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_y^{-1}(\mathbf{y} - \bar{\mathbf{y}}) \\ \boldsymbol{\Sigma}_{x|y} &= \boldsymbol{\Sigma}_x - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_y^{-1}\boldsymbol{\Sigma}_{yx} \end{aligned}$$

3.1.6 Canonical form of the Gaussian

As shown above, describing the information contained in a Gaussian distribution can be done in terms of its expectation and covariance. This is known as moment-based parameterization. A second parameterization is the canonical form. The parameters for the canonical form are the *information matrix* $\boldsymbol{\Lambda}$ and the the *potential vector* $\boldsymbol{\eta}$. They are defined as

$$\begin{aligned} \boldsymbol{\Lambda} &= \boldsymbol{\Sigma}^{-1} \\ \boldsymbol{\eta} &= \boldsymbol{\Lambda}\bar{\mathbf{x}} \end{aligned}$$

For a Gaussian distribution $\mathcal{N}(\bar{\mathbf{x}}, \boldsymbol{\Sigma})$, the equivalent canonical form is written

$$\mathcal{N}^{-1}(\boldsymbol{\eta}, \boldsymbol{\Lambda}) = \mathcal{N}(\boldsymbol{\Lambda}^{-1}\boldsymbol{\eta}, \boldsymbol{\Lambda}^{-1})$$

Marginalization and conditioning in canonical form

If the random vectors **X** and **Y** are distributed according to joint Gaussian distribution

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}^{-1}\left(\begin{bmatrix}\boldsymbol{\eta}_x\\ \boldsymbol{\eta}_y\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Lambda}_x & \boldsymbol{\Lambda}_{xy}\\ \boldsymbol{\Lambda}_{yx} & \boldsymbol{\Lambda}_y\end{bmatrix}\right)$$

Then the marginal distribution of **Y** is given by

$$\begin{aligned} p(\mathbf{y}) &= \mathcal{N}^{-1}\left(\boldsymbol{\eta}_*, \boldsymbol{\Lambda}_*\right) \\ \boldsymbol{\eta}_* &= \boldsymbol{\eta}_y - \boldsymbol{\Lambda}_{yx}\boldsymbol{\Lambda}_x^{-1}\boldsymbol{\eta}_x \\ \boldsymbol{\Lambda}_* &= \boldsymbol{\Lambda}_y - \boldsymbol{\Lambda}_{yx}\boldsymbol{\Lambda}_x^{-1}\boldsymbol{\Lambda}_{xy} \end{aligned}$$

and the conditional distribution of $\mathbf{X}$ given $\mathbf{Y} = \mathbf{y}$ is

$$\begin{aligned} p(\mathbf{x}|\mathbf{Y} = \mathbf{y}) &= \mathcal{N}^{-1}\left(\boldsymbol{\eta}_{x|y}, \boldsymbol{\Lambda}_{x|y}\right) \\ \boldsymbol{\eta}_{x|y} &= \boldsymbol{\eta}_x - \boldsymbol{\Lambda}_{xy}\mathbf{y} \\ \boldsymbol{\Lambda}_{x|y} &= \boldsymbol{\Lambda}_x \end{aligned}$$

3.1.7 Stochastic processes and the autocorrelation function

As discussed in Section 3.1.2, the outcome of a dice throw can be described by a discrete random variable. One could model five dice throws as five random variables that are Independent and identically distributed (IID), or a random vector $\mathbf{d} \in \mathbb{R}^5$ where each element is IID. However, if one is to consider the trajectory of a robot, that is its position $\in \mathbb{R}^3$ and how it evolves through time, this would in theory yield a vector with an infinite dimension. The possible realizations of the trajectory are now functions of time on the form $\mathbf{x}(t) : \mathbb{R} \to \mathbb{R}^3$.

Wiener process and white gaussian noise

To introduce stochastic processes, "The grandfather" of all stochastic processes, i.e. the *Wiener process*, must first be defined[10]. Let the stochastic process *x(t)* be defined as

$$\begin{aligned} x(t) &= x(nT) \\ n &\triangleq \frac{t}{T}, \\ x(nT) &= \sum_{i=1}^{n} x_i, \quad i \in 1, 2, ..., n \end{aligned}$$

where the x_i are IID random variables with expectation 0 and variance T. The Wiener process b(t) can then be defined as

$$b(t) = \lim_{T \to 0} x(t)$$

The timestep T goes to zero while n goes to infinity. The Wiener Process is often described as a random walk [10]. Further, white Gaussian noise is another essential stochastic process. In discrete time, white Gaussian noise is a sequence of IID random variables with a Gaussian distribution and zero expectation. In the continuous case, the stochastic process *n(t)*, white Gaussian noise is defined as

$$n(t) = \lim_{\Delta \to 0} \frac{b(t + \Delta) - b(t)}{\Delta} = b'(t)$$

Autocorrelation function

For a stochastic process $\mathbf{x}$(t), the Autocorrelation Function (ACF) can be defined as

$$R(t_1, t_2) = E[\mathbf{x}(t_1)\mathbf{x}(t_2)^{\top}].$$

The ACF measures how a stochastic process correlates with itself at different points in time. Some stochastic processes have an ACF that exhibits the following property: It depends only on the difference $\tau = t_2 - t_1$:

$$R(\tau) = E[\mathbf{x}(t)\mathbf{x}(t+\tau)^\top]$$

If the stochastic process has a constant expectation as well as this property, it is said to be wide sense stationary[10]. For a Gaussian white noise process with unit variance, the ACF is defined as

$$R(\tau) = \delta(\tau),$$

which means it is completely uncorrelated with itself at all times except at the current time, as well as being wide-sense stationary.

3.2 Least Squares Optimization

As will be evident later, optimization is a central part of the VIO problem. Optimization is finding the minimum of an objective function, subject to some constraints. This section will introduce some central concepts within linear and non-linear least-squares optimization.

3.2.1 Linear least squares

Consider the case where there is n *linear* equations

$$\mathbf{e}(\mathbf{x}) = \begin{bmatrix} e_1(\mathbf{x}) \\ e_2(\mathbf{x}) \\ \vdots \\ e_{n-1}(\mathbf{x}) \\ e_n(\mathbf{x}) \end{bmatrix} = \mathbf{0} \tag{3.10}$$

with m variables.

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_{m-1} & x_m \end{bmatrix}^\top$$

When $n > m$, Equation (3.10) generally does not have a solution. However, by formulating this as a least-squares minimization problem, one can get the values of $\mathbf{x}$ which puts the system $\mathbf{e}(\mathbf{x})$ closest to zero.

$$\begin{aligned} \mathbf{x}^* &= \arg\min_{\mathbf{x}} f(\mathbf{x}) && (3.11) \\ &= \arg\min_{\mathbf{x}} \mathbf{e}(\mathbf{x})^\top \mathbf{e}(\mathbf{x}) && (3.12) \\ &= \arg\min_{\mathbf{x}} ||\mathbf{e}(\mathbf{x})||^2 && (3.13) \end{aligned}$$

Since the equations in $\mathbf{e}(\mathbf{x})$ are linear, the system of equations can be written in the following way

$$\mathbf{e}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$$

Thus, the objective function can be written as

$$f(\mathbf{x}) = ||\mathbf{Ax} - \mathbf{b}||^2 \tag{3.14}$$

The values of **x** which minimizes Equation (3.14) are given by

$$\mathbf{x}^* = \arg\min_{x} ||\mathbf{Ax} - \mathbf{b}||^2$$

and for this function to be at a minimum, its gradient must be zero.

$$\begin{aligned} \nabla f(\mathbf{x}^*) = \nabla ||\mathbf{Ax}^* - \mathbf{b}||^2 = \\ \Rightarrow 2\mathrm{A}^\top(\mathbf{Ax}^* - \mathbf{b}) = \mathbf{0} \end{aligned}$$

This can be written in a form which gives us the *normal equations*

$$\mathbf{x}^* = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{b} \tag{3.15}$$

And this can be solved using QR- or Cholesky factorization.

3.2.2 Nonlinear least squares

In the linear case, the solution to the optimization problem comes straight from the normal equations. However, when the systems of equations $\mathbf{e}(\mathbf{x})$ in Equation (3.13) is nonlinear, which generally is the case for VIO, the normal equations can not be used to minimize Equation (3.13) directly. Instead, it can be assumed that $e(\mathbf{x})$ is approximately linear around the current estimate, using its first order Taylor-expansion

$$\mathbf{e}(\mathbf{x}_0 + \delta\mathbf{x}) \approx \mathbf{e}(\mathbf{x}_0) + \frac{\partial\,\mathbf{e}(\mathbf{x}_0)}{\partial\,\mathbf{x}}\delta\mathbf{x}$$

Where $\frac{\partial\,\mathbf{e}(\mathbf{x}_0)}{\partial\,\mathbf{x}}$ is the Jacobian as defined in Equation (3.16).

$$\begin{bmatrix} e_1(\mathbf{x}+\delta\mathbf{x}) \\ e_2(\mathbf{x}+\delta\mathbf{x}) \\ \vdots \\ e_{n-1}(\mathbf{x}+\delta\mathbf{x}) \\ e_n(\mathbf{x}+\delta\mathbf{x}) \end{bmatrix} \approx \begin{bmatrix} e_1(\mathbf{x}_0) \\ e_2(\mathbf{x}_0) \\ \vdots \\ e_{n-1}(\mathbf{x}_0) \\ e_n(\mathbf{x}_0) \end{bmatrix} + \begin{bmatrix} \frac{\partial e_1}{\partial x_1} & \frac{\partial e_1}{\partial x_2} & \cdots & \frac{\partial e_1}{\partial x_{n-1}} & \frac{\partial e_1}{\partial x_n} \\ \vdots & & & & \vdots \\ \vdots & & & & \vdots \\ \frac{\partial e_n}{\partial x_1} & \frac{\partial e_n}{\partial x_2} & \cdots & \frac{\partial e_n}{\partial x_{n-1}} & \frac{\partial e_n}{\partial x_n} \end{bmatrix} \begin{bmatrix} \delta x_1 \\ \delta x_2 \\ \vdots \\ \delta x_{n-1} \\ \delta x_n \end{bmatrix} \tag{3.16}$$

This is now a linear least squares problem

$$\mathbf{e}(\mathbf{x}+\delta\mathbf{x}) \approx \underbrace{\left[\tfrac{\partial\,\mathbf{e}(\mathbf{x}_0)}{\partial\,\mathbf{x}}\right]}_{\mathbf{A}}\underbrace{\left[\delta\mathbf{x}\right]}_{\mathbf{x}} - \underbrace{\left[-\mathbf{e}(\mathbf{x}_0)\right]}_{\mathbf{b}}$$

and can be solved using the normal equations from Equation (3.15).

$$\begin{aligned} \delta\mathbf{x}^* &= (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{b} \\ &= \left(\left(\frac{\partial\,\mathbf{e}(\mathbf{x}_0)}{\partial\,\mathbf{x}}\right)^\top \left(\frac{\partial\,\mathbf{e}(\mathbf{x}_0)}{\partial\,\mathbf{x}}\right)\right)^{-1} \left(\frac{\partial\,\mathbf{e}(\mathbf{x}_0)}{\partial\,\mathbf{x}}\right)^\top (-\mathbf{e}(\mathbf{x}_0)) \end{aligned}$$

Gauss-Newton

Directly applying this step to the objective function produces a new set of values in the vector of equations

$$\mathbf{e}(\mathbf{x}_1) = \mathbf{e}(\mathbf{x}_0 + \delta\mathbf{x}^*) \tag{3.17}$$

This can in turn be linearized and solved again. Iteratively linearizing, solving and applying the solution is called the Gauss-Newton Algorithm and is summarized in Algorithm 1[17].

Algorithm 1 Gauss-Newton Algorithm

```
Require:
  $f(\mathbf{x}) = ||\mathbf{e}(\mathbf{x})||^2$          ▷ Objective function in least squares form
  $\frac{\delta \mathbf{e}(\mathbf{x})}{\delta \mathbf{x}}$          ▷ Analytic Jacobian of error function
  $\hat{\mathbf{x}}_0$          ▷ Initial estimate which is close to solution
  for $k = 0, 1, \dots, k_{\max}$ do
    $\mathbf{b} \leftarrow (-\mathbf{e}(\hat{\mathbf{x}}_k))$          ▷ Calculate constant term at current estimate
    $\mathbf{A} \leftarrow \frac{\delta \mathbf{e}(\hat{\mathbf{x}}_k)}{\delta \mathbf{x}}$          ▷ Calculate Jacobian at current estimate
    $\delta\mathbf{x}^* \leftarrow (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$          ▷ Calculate solution
    $\hat{\mathbf{x}}_{k+1} \leftarrow \hat{\mathbf{x}}_k + \delta\mathbf{x}_k^*$

    if $f(\hat{\mathbf{x}}_{k+1}) < \epsilon_f$ or $|\delta\mathbf{x}^*| < \epsilon_x$ then
      If updated value or step length is below treshold, end earlier
      $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}}_{k+1}$
      return
    end if
  end for
```

If the initial estimate is close to the solution and the objective function is close to quadratic around the solution, this will converge. However, if the quadratic fit is poor or the initial estimate is far away from the solution, the algorithm may not converge.

Levenberg-Marquardt

Levenberg-Marquadt is a trust region method. A trust region method will define an area around its current estimate which it trusts to be quadratic. Instead of just solving the normal equaions in Equation (3.15), a nonnegative constant is added to the diagonal as in Equation (3.18)

$$\left(\mathbf{A}^\top \mathbf{A} + \lambda \operatorname{diag}\left(\mathbf{A}^\top \mathbf{A}\right)\right) \delta\mathbf{x} = \mathbf{A}^\top \mathbf{b} \tag{3.18}$$

if $\lambda = 0$, this reduces to Gauss Newton. If λ is big, the update step will be large when the gradient is small, and smaller when the gradient is large. The complete algorithm can be seen in Algorithm 2.

Algorithm 2 Levenberg-Marquardt Algorithm

```
Require:
  $f(\mathbf{x}) = ||\mathbf{e}(\mathbf{x})||^2$                ▷ Objective function in least squares form
  $\frac{\delta \mathbf{e}(\mathbf{x})}{\delta \mathbf{x}}$                ▷ Analytic Jacobian of error function
  $\hat{\mathbf{x}}_0$                ▷ Initial estimate
  $\lambda \leftarrow 10^{-4}$                ▷ Initial value for trust region parameter
  for $k = 0, 1, \dots, k_{\max}$ do
    $\mathbf{b} \leftarrow (-\mathbf{e}(\hat{\mathbf{x}}_k))$                ▷ Calculate constant term at current estimate
    $\mathbf{A} \leftarrow \frac{\delta \mathbf{e}(\hat{\mathbf{x}}_k)}{\delta \mathbf{x}}$                ▷ Calculate Jacobian at current estimate
    $\delta \mathbf{x}^* \leftarrow (\mathbf{A}^\top \mathbf{A} + \lambda \mathrm{diag}(\mathbf{A}^\top \mathbf{A}))^{-1} \mathbf{A}^\top \mathbf{b}$                ▷ Calculate solution

    if $f(\hat{\mathbf{x}}_k + \delta \mathbf{x}^*) < f(\hat{\mathbf{x}}_k)$ then
      Accept update and increase trust region
      $\hat{\mathbf{x}}_{k+1} \leftarrow \hat{\mathbf{x}}_k + \delta \mathbf{x}^*$
      $\lambda \leftarrow \lambda / 10$
    else
      Reject update and decrease trust region
      $\hat{\mathbf{x}}_{k+1} \leftarrow \hat{\mathbf{x}}_k$
      $\lambda \leftarrow \lambda * 10$
    end if

    if $f(\hat{\mathbf{x}}_{k+1}) < \epsilon_f$ or $|\delta \mathbf{x}^*| < \epsilon_x$ then
      If updated value or step length is below treshold, end earlier
      $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}}_{k+1}$
      return
    end if
  end for
```

The trust region methodology somewhat relaxes the need for an initial estimate which is close to the solution, as it will converge to a local minima, at the expense of speed[17].

3.3 Rigid Body Kinematics

A point in space, e.g. the position of a robot, is a physical entity. To describe this physical entity, one may use a *coordinate vector*. There are several ways of representing the position. Two examples are Cartesian coordinate vectors $\in \mathbb{R}^3$ and homogeneous coordinates $\in \mathbb{P}^3$. Describing the position and orientation of objects in three-dimensional space is a fundamental piece of the VIO problem, so this section will give a summary of the theory behind this. Figures in this section are inspired and guided by the open-source code in [18].

3.3.1 Cartesian coordinates

A Cartesian coordinate vector describing the displacement of the point X in *Euclidean space* $\mathbb{R}^3$ relative to the axes of *Cartesian Coordinate frame* $\mathcal{F}_a$ is denoted $\mathbf{x}^a$.

$$\mathbf{x}^a = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3$$

The *Cartesian Coordinate frame* is a set of orthogonal axes which intersect at a point called the origin. The relation between $\mathbf{x}^a$, $\mathcal{F}_a$ and X is illustrated in Figure 3.3.

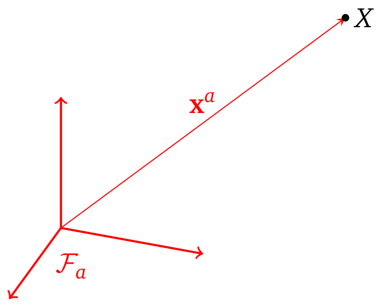

Figure 3.3: Cartesian coordinate vector describing point X

3.3.2 Homogeneous coordinates

An alternative to Cartesian coordinates is to describe points using *homogeneous coordinates* in *Projective space*.

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{bmatrix} \in \mathbb{P}^3$$

A unique property of the projective space is that all points that are proportional to each other are equal, formally noted as:

$$\tilde{\mathbf{x}} = \lambda \tilde{\mathbf{x}} \quad \forall \lambda \in \mathbb{R}^{\setminus}\{0\} \tag{3.19}$$

Conversion between homogeneous and Cartesian coordinates

Given a Cartesian coordinate vector,

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ x \end{bmatrix} \in \mathbb{R}^3$$

this can be expressed as a Homogeneous Coordinate by appending a one as the last coordinate

$$\tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \in \mathbb{P}^3$$

From Equation (3.19), it is evident that this is equivalent to any scaled version of $\tilde{\mathbf{x}}$.

$$\begin{aligned} \tilde{\mathbf{x}} &\in \mathbb{P}^3 \\ \tilde{\mathbf{x}} &= \lambda \tilde{\mathbf{x}} \in \mathbb{P}^3 \\ \check{\mathbf{x}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} &= \lambda \tilde{\mathbf{x}}, \quad \lambda = 1 \end{aligned}$$

There are infinitely many differently scaled versions of this, but only one version has its fourth coordinate set to one. When this is the case, it is referred to as a *normalized* homogeneous coordinate. To signal that it is a normalized homogeneous coordinate, $\check{\mathbf{x}}$ is used. For the conversion from homogeneous to Cartesian coordinates, one can consider the homogeneous coordinate

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{bmatrix} \in \mathbb{P}^3$$

This is converted back to Cartesian coordinates as shown in Equation (3.20)

$$\mathbf{x} = \begin{bmatrix} \tilde{x}/\tilde{w} \\ \tilde{y}/\tilde{w} \\ \tilde{z}/\tilde{w} \end{bmatrix} \in \mathbb{R}^3 \tag{3.20}$$

This implies that it is not possible to convert homogeneous coordinates with $\tilde{w} = 0$, as this represents coordinates that are infinitely far away from the Cartesian coordinate frame origin.

3.3.3 Multiple coordinate frames

As will be evident later in this chapter, having multiple coordinate frames can simplify modelling. This situation is illustrated in Figure 3.4.

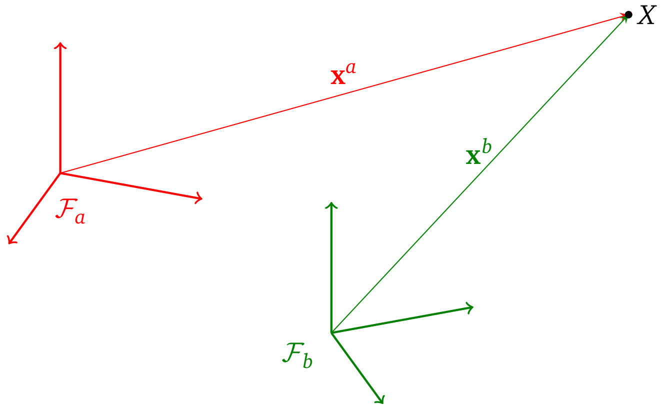

Figure 3.4: Point X described with respect to multiple coordinate frames

The relationship between two Cartesian coordinate frames $\mathcal{F}_a$ and $\mathcal{F}_b$ is determined by the position and orientation of frame $\mathcal{F}_b$ relative to frame $\mathcal{F}_a$.

Rotation between frames

The axes of $\mathcal{F}_b$ are rotated compared to the axes of $\mathcal{F}_a$. To compensate for this, a rotation of the coordinates is done with $\mathbf{R}_{ab}$. This is a *Rotation matrix* and has three degrees of freedom. A rotation of the coordinate system is illustrated in Figure 3.5

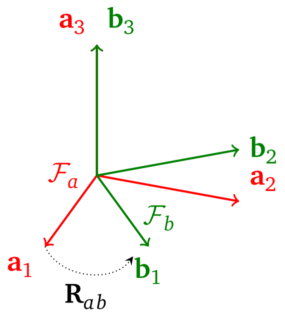

Figure 3.5: Two Frames rotated with respect to each other

Rotation matrices are in the *SO(3)* Lie Group (special orthogonal group of dimension 3), which is a set of matrices with the following attributes

$$SO(3) = \{\mathbf{R} | \mathbf{R} \in \mathbb{R}^{3x3}, \mathbf{R} \text{ is orthogonal and } \det(\mathbf{R}) = 1\} \tag{3.21}$$

The inverse of this matrix is given by

$$\mathbf{R}_{ab}^{-1} = \mathbf{R}_{ab}^{\top} = \mathbf{R}_{ba}$$

Translation Between Frames

The translational offset between the two frames is summarized in the vector $\mathbf{t}^a_{ab}$. This is the position of the origin of $\mathcal{F}_b$ with respect to $\mathcal{F}_a$ given in the coordinates of $\mathcal{F}_a$.

$$\mathbf{t}^a_{ab} = \begin{bmatrix} x^a_{ab} \\ y^a_{ab} \\ z^a_{ab} \end{bmatrix}$$

This operation is illustrated in Figure 3.6

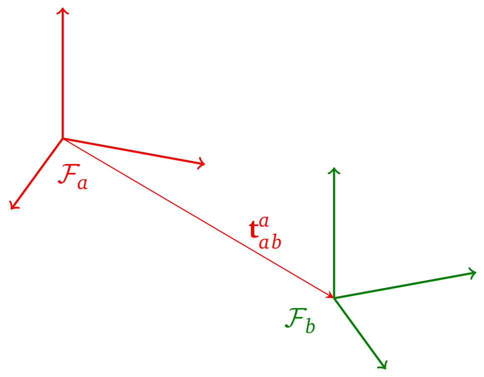

Figure 3.6: Compensating for the translational offset of the origin in $\mathcal{F}_b$ with respect to the origin in $\mathcal{F}_a$

Rotation and translation between frames

The combination of the position and orientation is referred to as the *pose* of the frame. Considering this, the relation between $\mathbf{x}^a$ and $\mathbf{x}^b$ is given by Equation (3.22)

$$\mathbf{x}^a = \mathbf{R}_{ab}\mathbf{x}^b + \mathbf{t}^a_{ab} \tag{3.22}$$

Frame transformations with homogeneous coordinates

The conversion between the two coordinate systems can be summarized in Figure 3.7 and consists of a matrix product and a sum.

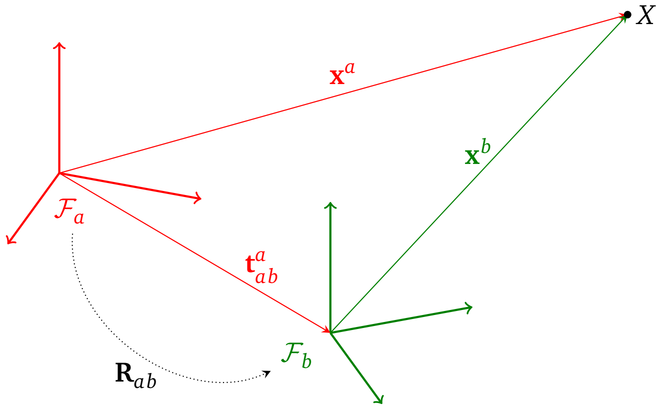

Figure 3.7: Conversion between coordinates

Using homogeneous coordinates however, this operation can be summarized in a single matrix product:

$$\tilde{\mathbf{x}}^a = \mathbf{T}_{ab}\tilde{\mathbf{x}}^b$$

$\mathbf{T}_{ab}$ is a homogeneous transformation matrix

$$\mathbf{T}_{ab} = \begin{bmatrix} \mathbf{R}_{ab} & \mathbf{t}^a_{ab} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \in SE(3)$$

and $SE(3)$ is the Special Euclidian group of dimension three:

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \in \mathbb{R}^4 \;\middle|\; \mathbf{R} \in \mathbb{R}^{3x3}, \quad \mathbf{R}\mathbf{R}^\top = \mathbf{I}, \quad \det(\mathbf{R}) = 1, \quad \mathbf{t} \in \mathbb{R}^3 \right\} \tag{3.23}$$

The inverse of $\mathbf{T}_{ab}$ is given by

$$\mathbf{T}_{ab} = \mathbf{T}^{-1}_{ab} = \begin{bmatrix} \mathbf{R}^\top_{ab} & -\mathbf{R}^\top_{ab}\mathbf{t}^a_{ab} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}$$

The matrix operation is illustrated in Figure 3.8

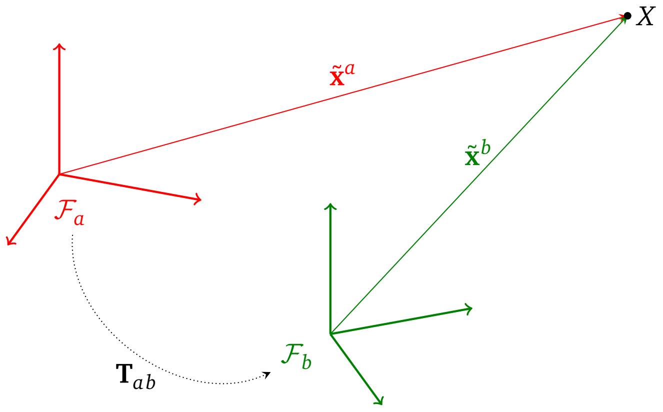

Figure 3.8: Relation between homogeneous coordinates $\tilde{\mathbf{x}}^a$ and $\tilde{\mathbf{x}}^b$

3.3.4 Alternative representations of rotation

In addition to representing rotation by rotation matrices, they can be represented using *Euler angles* and *unit quaternions*.

Euler angles

Since a body in space intrinsically has three rotational degrees of freedom, it is possible to *minimally* represent this. A minimal representation uses the same number of variables as degrees of freedom. For rotational degrees of freedom, this is called *Euler Angles*. They are conventionally denoted by θ, ϕ and ψ respectively for pitch, roll and yaw. Euler angle rotations are executed in a sequential manner, where different conventions are applied. The sequence of which the rotations are carried out matters. With the *zyx* conventions, a rotation is given by

$$\begin{aligned}
\mathbf{R}(\psi, \theta, \phi) &= \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi}, \\
\mathbf{R}_{z,\psi} &= \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
\mathbf{R}_{y,\theta} &= \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix} \\
\mathbf{R}_{x,\phi} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix}
\end{aligned}$$

For, e.g. ships considering a dynamical positioning problem and other "slow" dynamical systems, this is a good representation. However, for systems undertaking highly dynamic operations, such as an Unmanned Aerial Vehicle or a robot navigating dynamic terrain, this representation will eventually suffer from singularities, where an infinite number of possible Euler angles will be a valid attitude representation. [10, 16, 19]

Unit quaternions

Just as the multiplication of complex numbers can model rotation in the complex plane, a *unit quaternion* model rotations in three dimensions. A quaternion is a four-dimensional number, which comprises the sum of one real number and three imaginary numbers. To highlight the similarity between two-dimensional complex numbers and quaternions, a quaternion can be written as

$$q = \eta + \epsilon_1 i + \epsilon_2 j + \epsilon_3 k$$

where η represents the real part, ϵ is the complex part, and i, j and k are the complex axes, all perpendicular to each other and to the real-valued axis. However, depending on the convention, a quaternion is often written more compactly as

$$\mathbf{q} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix}, \eta \in \mathbb{R}, \boldsymbol{\epsilon} \in \mathbb{R}^3$$

As the entity to be modelled has three degrees of freedom, the quaternion is set to the unit norm. This brings the degrees of freedom for the quaternion down to three. A quaternion with a unit norm is called a *unit quaternion*. Similarly to standard complex numbers, addition with quaternions are done in a straightforward manner

$$\mathbf{q}_a + \mathbf{q}_b = \begin{bmatrix} \eta_a \\ \epsilon_a \end{bmatrix} + \begin{bmatrix} \eta_b \\ \epsilon_b \end{bmatrix} = \begin{bmatrix} \eta_a + \eta_b \\ \epsilon_a + \epsilon_b \end{bmatrix}$$

Multiplication of quaternions are however more involved. A quaternion product is defined as

$$\mathbf{q}_a \otimes \mathbf{q}_b = \begin{bmatrix} \eta_a \eta_b - \epsilon_a^\top \epsilon_b \\ \eta_b \epsilon_a + \eta_a \epsilon_b + \epsilon_a \times \epsilon_b \end{bmatrix}$$

Some additional useful concepts are the *conjugate quaternion*, the *identity quaternion*, the *quaternion norm* and the *inverse* of a quaternion. The identity quaternion is given by Equation (3.24) with the property described in Equation (3.25). The conjugate of a quaternion is given by Equation (3.26). The norm of a quaternion is given by Equation (3.27). For a unit quaternion, this is always 1. The inverse of a quaternion is given by Equation (3.28) and has the property that Equation (3.29) holds. For a unit quaternion, this always reduces to the conjugate only.

$$\mathbf{q}_I = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \tag{3.24}$$

$$\mathbf{q}_I \otimes \mathbf{q}_a = \mathbf{q}_a \otimes \mathbf{q}_I = \mathbf{q}_a \tag{3.25}$$

$$\mathbf{q}^* = \begin{bmatrix} \eta \\ -\epsilon \end{bmatrix} \tag{3.26}$$

$$||\mathbf{q}|| = \sqrt{\eta^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2} \tag{3.27}$$

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{||\mathbf{q}||} \tag{3.28}$$

$$\mathbf{q}^{-1} \otimes \mathbf{q} = \mathbf{q} \otimes \mathbf{q}^{-1} = \mathbf{q}_I \tag{3.29}$$

Rotation by the means of a quaternion can now be defined. For a rotation given by the angle-axis-representation $v' = \mathbf{R}_{\mathbf{n},\alpha} v$, the rotation can also be written as

$$\begin{bmatrix} 0 \\ \mathbf{v}' \end{bmatrix} = \mathbf{q} \otimes \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \otimes \mathbf{q}^*,$$

$$\mathbf{q} = \begin{bmatrix} \eta \\ \epsilon \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\alpha}{2}\right) \\ \mathbf{n} \sin\left(\frac{\alpha}{2}\right) \end{bmatrix}$$

3.4 Lie Theory

Variables such as pose and orientation lie on the manifold. This complicates things in regards to perturbing, which in turn complicates things with regards to taking derivatives and calculating probability distributions. The groups $SO(3)$ and $SE(3)$ which was introduced in Equation (3.21) and Equation (3.23) are referred to as *Matrix Lie Groups* and are examples of this. To illustrate this, consider the following case

$$\mathbf{R} \in SO(3), \delta\mathbf{R} \in SO(3), \qquad \mathbf{R} + \delta\mathbf{R} \notin SO(3) \tag{3.30a}$$
$$\mathbf{T} \in SE(3), \delta\mathbf{T} \in SE(3), \qquad \mathbf{T} + \delta\mathbf{T} \notin SE(3) \tag{3.30b}$$

Evident from Equation (3.30), perturbing a rotation matrix or pose matrix will not produce elements which still are in these groups. Lie theory [20] describes how one can use the *tangent space* to these manifolds to add, subrtract, take derivatives and describe probability distributions for rotations and poses. Most of this section is heavily based on [18] and [20].

3.4.1 The Lie group

In general, a Lie-group is both a manifold and a group. A group $(\mathcal{G}, \circ)$ consists of a set $\mathcal{G}$ and a composition operator $\circ$. The group must satisfy the group axioms: For the elements $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z} \in \mathcal{G}$

$$\begin{aligned}
\text{Closure under } \circ: &\quad \mathcal{X} \circ \mathcal{Y} \in \mathcal{G} \\
\text{Identity } \mathcal{E}: &\quad \mathcal{E} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{E} = \mathcal{X} \\
\text{Inverse } \mathcal{X}^{-1}: &\quad \mathcal{X}^{-1} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{E} \\
\text{Associativity}: &\quad (\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z})
\end{aligned}$$

In addition to this, a Lie group can perform an *action* $\cdot$ on other sets. Given an element of the Lie-group $\mathcal{X} \in \mathcal{M}$ and an element of an arbitrary set $v \in \mathcal{V}$, a group action $\cdot$ must satisfy the axioms:

$$\begin{aligned}
\text{Identity}: &\quad \mathcal{E} \cdot v = v \\
\text{Compatibility}: &\quad (\mathcal{X} \circ \mathcal{Y}) \cdot v = \mathcal{X} \cdot (\mathcal{Y} \cdot v)
\end{aligned}$$

SO(3) and SE(3)

For the rotation group $SO(3)$ and the pose group $SE(3)$, the elements are matrices, the composition operation is matrix multiplication, and the inversion operation is matrix inversion. The action on vectors is $\mathbf{R} \cdot \mathbf{x} \triangleq \mathbf{R}\mathbf{x}$ for rotations $\mathbf{R} \in SO(3)$ and $\mathbf{T} \cdot \mathbf{x} \triangleq \mathbf{R}\mathbf{x} + \mathbf{t}$ for poses $\mathbf{T} \in SE(3)$.

3.4.2 The tangent space and the Lie algebra

The tangent space to the Lie group $\mathcal{M}$ at $\mathcal{X}$ is denoted $\mathcal{TM}_{\mathcal{X}}$. The structure of the tangent spaces on a Lie group manifold is the same everywhere, but the tangent space at the identity $\mathcal{E}$ is called the *Lie Algebra* of $\mathcal{M}$. This is denoted $\mathfrak{m}$

$$\text{Lie Algebra} : \mathfrak{m} \triangleq \mathcal{TM}_{\mathcal{E}} \tag{3.31}$$

The Lie Algebra is a vector space. Its elements $\boldsymbol{\tau}^{\wedge} \in \mathfrak{m}$ can have a complex structure, but details of this are deemed unnecessary here. They can however also be identified with vectors $\boldsymbol{\tau} \in \mathbb{R}^m$. The capitalized exponential operator maps elements in the Lie Algebra vector space to the manifold.

$$\begin{aligned} \text{Exp} : \mathbb{R}^m &\mapsto \mathcal{M} \\ \text{Exp}(\boldsymbol{\tau}) &= \mathcal{X} \end{aligned}$$

The capitalized logarithmic operator maps elements on the manifold to the Lie Algebra vector space

$$\begin{aligned} \text{Log} : \mathcal{M} &\mapsto \mathbb{R}^m \\ \text{Log}(\mathcal{X}) &= \boldsymbol{\tau} \end{aligned}$$

Figure 3.9 illustrates the tangent space, manifold and its elements.

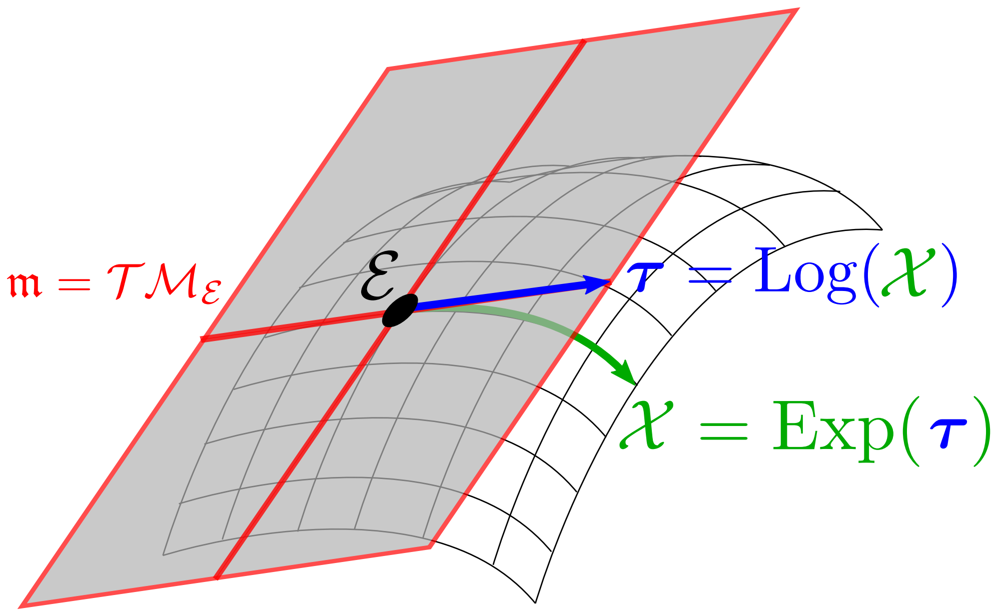
$\mathfrak{m} = \mathcal{TM}_{\mathcal{E}}$
$\mathcal{E}$
$\boldsymbol{\tau} = \text{Log}(\mathcal{X})$
$\mathcal{X} = \text{Exp}(\boldsymbol{\tau})$

Figure 3.9: A manifold with its Lie Algebra vector space as the transparent grey area. In green there is the on-manifold group element $\mathcal{X}$ and in blue its corresponding Lie Algebra vector $\boldsymbol{\tau}$

Tangent space, Logarithmic Map and Exponential Map for SO(3)

The tangent space vector for the rotational matrix Lie group $SO(3)$ group has three degrees of freedom, $\boldsymbol{\theta} \in \mathbb{R}^3$. $\boldsymbol{\theta}$ corresponds to the angle-axis representation of a rotation

$$\mathrm{Log}(\mathbf{R}) = \boldsymbol{\theta} \triangleq \theta \mathbf{u}$$

Where $\mathbf{u}$ is the axis of rotation and θ is the angle of rotation. The Exp-map from the tangent space vector to the rotation matrix is then given by rodriguez formula

$$\mathbf{R} = \mathrm{Exp}(\boldsymbol{\theta}) = \mathbf{I} + \sin\theta \left[\mathbf{u}\right]_\times + (1 - \cos\theta)\left[\mathbf{u}\right]_\times^2 \tag{3.32}$$

In Equation (3.32) $\left[\cdot\right]_\times$ represents the skew symmetric-matrix

$$\left[\mathbf{u}\right]_\times = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}_\times = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}$$

Tangent space, logarithmic and exponential map for SE(3)

The tangent space vector ξ of matrix Lie group for pose, SE(3), has six degrees of freedom, three for translation and three for rotation.

$$\xi = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\theta} \end{bmatrix} \in \mathbb{R}^6 \tag{3.33}$$

The Exponential map is then given by

$$\mathbf{T} = \mathrm{Exp}(\xi) = \begin{bmatrix} \mathrm{Exp}(\boldsymbol{\theta}) & \mathbf{V}(\boldsymbol{\theta})\boldsymbol{\rho} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{4x4}$$

where $\mathbf{V}(\boldsymbol{\theta})$ is defined as

$$\mathbf{V}(\boldsymbol{\theta}) = \mathbf{I} + \frac{1 - \cos\theta}{\theta}\left[\mathbf{u}\right]_\times + \frac{\theta - \sin\theta}{\theta}\left[\mathbf{u}\right]_\times^2$$

The Logarithmic map is defined as

$$\xi = \mathrm{Log}(\mathbf{T}) = \begin{bmatrix} \mathbf{V}^{-1}(\boldsymbol{\theta})\mathbf{t} \\ \boldsymbol{\theta} \end{bmatrix}$$

The variables $\boldsymbol{\rho}$ and $\mathbf{t}$ are thus not the same but related by the matrix $\mathbf{V}(\boldsymbol{\theta})$.

3.4.3 Addition and subtraction on the manifold

Again, consider the manifold $\mathcal{M}$. Using the composition operator $\circ$, it is possible to define a group transformation $\mathcal{Y} \in \mathcal{M}$, in two ways. One composed of first performing group transformation $\mathcal{X}$ then an on-manifold perturbation $\boldsymbol{\delta}$, and one composed of first doing the on-manifold perturbation $\boldsymbol{\delta}$, then $\mathcal{X}$:

$$\mathcal{Y} = \mathcal{X} \circ \boldsymbol{\delta}^{\mathcal{X}} \tag{3.34a}$$
$$= \boldsymbol{\delta}^{\mathcal{E}} \circ \mathcal{X} \tag{3.34b}$$

The composition operator is not communicative, so in general $\boldsymbol{\delta}^{\mathcal{X}} \neq \boldsymbol{\delta}^{\mathcal{E}}$. The on-manifold perturbations $\boldsymbol{\delta}^{\mathcal{E}}$ and $\boldsymbol{\delta}^{\mathcal{X}}$ in Equation (3.34) can also be described using the tangent space perturbation $\boldsymbol{\tau}$. The composition then looks like

$$\mathcal{Y} = \mathcal{X} \circ \mathrm{Exp}(\boldsymbol{\tau}^{\mathcal{X}}) \tag{3.35a}$$
$$= \mathrm{Exp}(\boldsymbol{\tau}^{\mathcal{E}}) \circ \mathcal{X} \tag{3.35b}$$

The tangential perturbation performed in Equation (3.35a) is referred to as right perturbation, and conversely, the perturbation in Equation (3.35b) is referred to as left-perturbation. By doing the transformations in reverse and taking the capitalized logarithm, one can obtain the tangential perturbations which separates $\mathcal{X}$ and $\mathcal{Y}$

$$\boldsymbol{\tau}^{\mathcal{X}} = \mathrm{Log}(\mathcal{X}^{-1} \circ \mathcal{Y}) \tag{3.36a}$$
$$\boldsymbol{\tau}^{\mathcal{E}} = \mathrm{Log}(\mathcal{Y} \circ \mathcal{X}^{-1}) \tag{3.36b}$$

From Equation (3.35) and Equation (3.36) one can define the equivalent of addition and subtraction for Lie groups by performing right perturbations. This is referred to as *right plus* Equation (3.37) and *right minus* operators:

$$\mathcal{Y} = \mathcal{X} \oplus \boldsymbol{\tau}^{\mathcal{X}} \triangleq \mathcal{X} \circ \mathrm{Exp}(\boldsymbol{\tau}^{\mathcal{X}}) \in \mathcal{M} \tag{3.37}$$
$$\boldsymbol{\tau}^{\mathcal{X}} = \mathcal{Y} \ominus \mathcal{X} \triangleq \mathrm{Log}(\mathcal{X}^{-1} \circ \mathcal{Y}) \in \mathcal{T}\mathcal{M}_{\mathcal{X}} \tag{3.38}$$

For completeness, the left plus and left minus operators are defined as

$$\mathcal{Y} = \boldsymbol{\tau}^{\mathcal{E}} \oplus \mathcal{X} \triangleq \mathrm{Exp}(\boldsymbol{\tau}^{\mathcal{E}}) \circ \mathcal{X} \in \mathcal{M}$$
$$\boldsymbol{\tau}^{\mathcal{E}} = \mathcal{Y} \ominus \mathcal{X} \triangleq \mathrm{Log}(\mathcal{Y} \circ \mathcal{X}^{-1}) \in \mathcal{T}\mathcal{M}_{\mathcal{E}}$$

The difference between right plus and left plus is illustrated in Figure 3.10

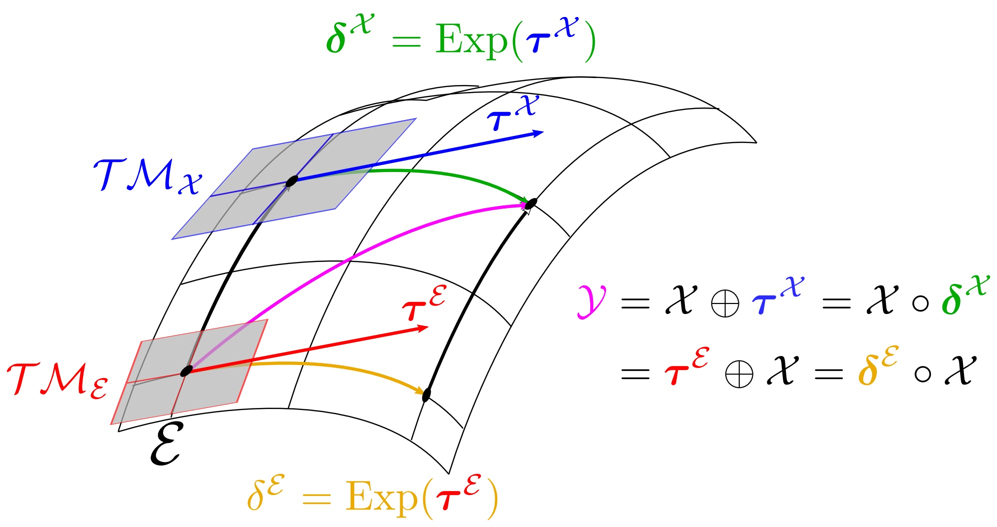
$\boldsymbol{\delta}^{\mathcal{X}} = \mathrm{Exp}(\boldsymbol{\tau}^{\mathcal{X}})$
$\boldsymbol{\tau}^{\mathcal{X}}$
$\mathcal{TM}_{\mathcal{X}}$
$\boldsymbol{\tau}^{\mathcal{E}}$
$\mathcal{Y} = \mathcal{X} \oplus \boldsymbol{\tau}^{\mathcal{X}} = \mathcal{X} \circ \boldsymbol{\delta}^{\mathcal{X}}$
$= \boldsymbol{\tau}^{\mathcal{E}} \oplus \mathcal{X} = \boldsymbol{\delta}^{\mathcal{E}} \circ \mathcal{X}$
$\mathcal{TM}_{\mathcal{E}}$
$\mathcal{E}$
$\delta^{\mathcal{E}} = \mathrm{Exp}(\boldsymbol{\tau}^{\mathcal{E}})$

Figure 3.10: The difference between right and left perturbations, as well as the difference between the right plus and left plus operator

3.4.4 Derivatives on the Manifold

Using the new plus and minus operators defined in Equation (3.37) and Equation (3.38) it is possible to define derivatives of Lie groups. This is necessary when running algorithms such as Algorithm 1 and Algorithm 2 to solve optimization problems involving Lie groups. For a function $f : \mathcal{M} \mapsto \mathcal{N}$, the right derivative is given by

$$\mathbf{J} = \frac{\partial f(\mathcal{X})}{\partial \mathcal{X}} \triangleq \lim_{\tau \to 0} \frac{f(\mathcal{X} \oplus \boldsymbol{\tau}^{\mathcal{X}}) \ominus f(\mathcal{X})}{\boldsymbol{\tau}^{\mathcal{X}}} = \frac{\mathrm{Log}\big(f(\mathcal{X})^{-1} \circ f(\mathcal{X} \circ \mathrm{Exp}(\boldsymbol{\tau}^{\mathcal{X}}))\big)}{\boldsymbol{\tau}^{\mathcal{X}}} \tag{3.39}$$

3.4.5 Probability distributions on the manifold

A random variable on the manifold can be expressed as a perturbation

$$\begin{gathered} \mathcal{X} = \bar{\mathcal{X}} \oplus \boldsymbol{\tau}, \quad \bar{\mathcal{X}} \in \mathcal{M} \\ \boldsymbol{\tau} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \in \mathcal{TM}_{\bar{\mathcal{X}}} \end{gathered}$$

3.5 Sensor Models

When estimating the states of the system, such as position, velocity and orientation, it is necessary to have a model of the sensor for prediction and optimization. In this section, modelling theory for the relevant sensors will be introduced.

3.5.1 IMU modelling

An IMU is usually mounted at the center of gravity of a moving body. That way, one can model its measurements without accounting for lever arms. The body-frame $\mathcal{F}_b$ is a coordinate frame fixed to the IMU and is the frame in which the measurements are modelled. Measurements from an IMU are usually assumed to be affected by two types of noise. One component is a slowly varying bias and another is Gaussian white noise. Consider the true acceleration $\mathbf{a}^b \in \mathbb{R}^3$ and the true angular velocity $\boldsymbol{\omega}^b \in \mathbb{R}^3$ of the system, both given in the body frame. They are then corrupted according to Equation (3.40)

$$\tilde{\mathbf{a}} = \mathbf{a} + \mathbf{b}_a + \mathbf{w}_a \tag{3.40a}$$

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \mathbf{b}_\omega + \mathbf{w}_\omega \tag{3.40b}$$

and the noise components are defined as Equation (3.41)

$$\dot{\mathbf{b}}_a = \mathbf{b}_a + \mathbf{w}_{ba}, \qquad \mathbf{b}_a \in \mathbb{R}^3 \tag{3.41a}$$

$$\dot{\mathbf{b}}_\omega = \mathbf{b}_\omega + \mathbf{w}_{b\omega}, \qquad \mathbf{b}_\omega \in \mathbb{R}^3 \tag{3.41b}$$

$$\begin{bmatrix} \mathbf{w}_a \\ \mathbf{w}_\omega \\ \mathbf{w}_{ba} \\ \mathbf{w}_{b\omega} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_a & 0 & 0 & 0 \\ 0 & \boldsymbol{\Sigma}_\omega & 0 & 0 \\ 0 & 0 & \boldsymbol{\Sigma}_{ba} & 0 \\ 0 & 0 & 0 & \boldsymbol{\Sigma}_{b\omega} \end{bmatrix} \right) \tag{3.41c}$$

Positional inference from IMU

Considering the acceleration measurements and no angular velocity, one can make a toy model of how the position $\mathbf{p} \in \mathbb{R}^3$ would change according to the measurements from the accelerometer. This is shown in Equation (3.42)

$$\dot{\mathbf{p}}^w = \mathbf{R}_{wb}\mathbf{v}^b \tag{3.42a}$$

$$\dot{\mathbf{v}}^b = \mathbf{a}^b \tag{3.42b}$$

$$\mathbf{a}^b = \tilde{\mathbf{a}}^b - \mathbf{b}_a - \mathbf{w}_a \tag{3.42c}$$

$$\dot{\mathbf{b}}_a = \mathbf{b}_a + \mathbf{w}_{ba} \tag{3.42d}$$

3.5.2 Camera modelling

Geometric camera models describes the camera projection process. In the general case a camera projection π is a function that maps a point in 3D from the camera frame $\mathcal{F}_c$ to the image domain Ω

$$\pi : \mathbb{R}^3 \to \Omega$$

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \pi(\mathbf{x}^c)$$

$\Omega \subset \mathbb{R}^2$ is the image domain of all valid pixels. The inverse gives us the point in 3D given a point $\mathbf{u}$ in the projected image and the depth d.

$$\pi^{-1} : \Omega \times \mathbb{R}^+ \rightarrow \mathbb{R}^3$$
$$\mathbf{x}^c = \begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} = \pi^{-1}(\mathbf{u}, d)$$

For camera projections, projective space $\mathbb{P}^2$ and homogeneous coordiantes are helpful. To convert image coordinates to normalized homogeneous coordinates, just append a single one to the end

$$\begin{bmatrix} u \\ v \end{bmatrix} \in \mathbb{R}^2 \mapsto \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \in \mathbb{P}^2$$

The conversion back from homogeneous to Cartesian coordinates is done the following way:

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \in \mathbb{P}^2 \mapsto \begin{bmatrix} \tilde{u}/\tilde{w} \\ \tilde{v}/\tilde{w} \end{bmatrix} \in \mathbb{R}^2$$

Pinhole projection model

The most commonly used camera projection model is the *Frontal Pinhole Projection Model*. In this model, the real-world point $\mathbf{x}^c$, given in the camera frame $\mathcal{F}_c$, is projected onto a plane which is at $z^c = f$. This distance is called the focal length. For an ideal camera, the focal length $f = 1$. This results in the images being projected to the *normalized image plane*, and this process is illustrated in Figure 3.11.

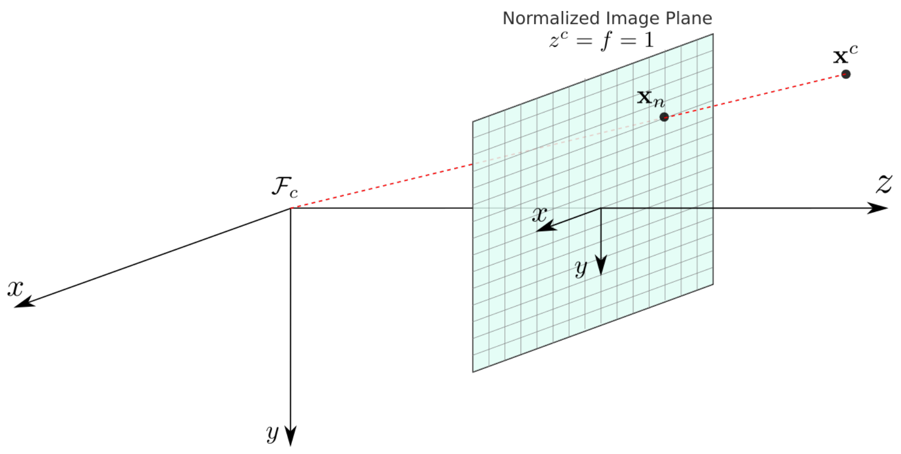

Figure 3.11: The projection of point $\mathbf{x}^c$ to the normalized image plane

Normalized image plane

Mathematically, one can project the three-dimensional point $\mathbf{x}^c$, given in normalized homogeneous coordinates, $\breve{\mathbf{x}}^c$, onto the normalized image plane with homogeneous perspective projection matrix Π_0 and scaling factor $\frac{1}{z^c}$.

$$\breve{\mathbf{x}}_n = \frac{1}{z^c}\Pi_0\breve{\mathbf{x}}^c, \quad \breve{\mathbf{x}}_n \in \mathbb{P}^2, \quad \breve{\mathbf{x}}^c \in \mathbb{P}^3 \tag{3.43a}$$

$$\underset{\breve{\mathbf{x}}_n}{\begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}} = \frac{1}{z^c} \underset{\Pi_0}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}} \underset{\breve{\mathbf{x}}^c}{\begin{bmatrix} x^c \\ y^c \\ z^c \\ 1 \end{bmatrix}} \tag{3.43b}$$

In Equation (3.43), a point in three dimensional projective space is mapped onto a two-dimensional plane.

$$\breve{\mathbf{x}}_n = \mathbf{x}_n^c = \frac{1}{z^c}\mathbf{x}^c \tag{3.44a}$$

$$\underset{\breve{\mathbf{x}}_n}{\begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}} = \underset{\mathbf{x}_n^c}{\begin{bmatrix} x_n^c \\ y_n^c \\ 1 \end{bmatrix}} = \underset{\frac{1}{z^c}\mathbf{x}^c}{\begin{bmatrix} \frac{x^c}{z^c} \\ \frac{y^c}{z^c} \\ \frac{z^c}{z^c} \end{bmatrix}} \tag{3.44b}$$

Here, the point $\mathbf{x}^c$ is scaled down to $\mathbf{x}_n^c$, so it resides in the normalized image plane.

Image plane and the calibration matrix

However, most cameras neither have a centered origin nor a focal length of 1. Cameras have intrinsic differences and this is accounted for in the matrix $\mathbf{K}$

$$\breve{\mathbf{u}} = \mathbf{K}\breve{\mathbf{x}}_n$$

$$\underset{\breve{\mathbf{u}}}{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}} = \underset{\mathbf{K}}{\begin{bmatrix} f_u & s_\theta & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}} \underset{\breve{\mathbf{x}}_n}{\begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}}$$

The matrix $\mathbf{K}$ is called the *intrinsic matrix* or the *calibration matrix*. The terms in the matrix can be explained the following way.

- f_u: Size of the unit length in horizontal pixels.
  - Alternatively it can be described in terms of the product fs_u:
    - f: focal length in metric units
    - s_u: Scaling factor that describes the horizontal pixel density in pixels per metric unit

- f_v: The same as f_u, just for vertical units
- c_u: u-coordinate of principal point in $\mathcal{F}_i$
- c_v: v-coordinate of principal point in $\mathcal{F}_i$
- s_θ: Skew-factor
  - Proportional to $\cot\theta$, where θ is the angle between u and v
  - This is neglected in the rest of the chapter, as it makes some expressions easier

The projection into **u** is illustrated in Figure 3.12. The terms in **K** are estimated

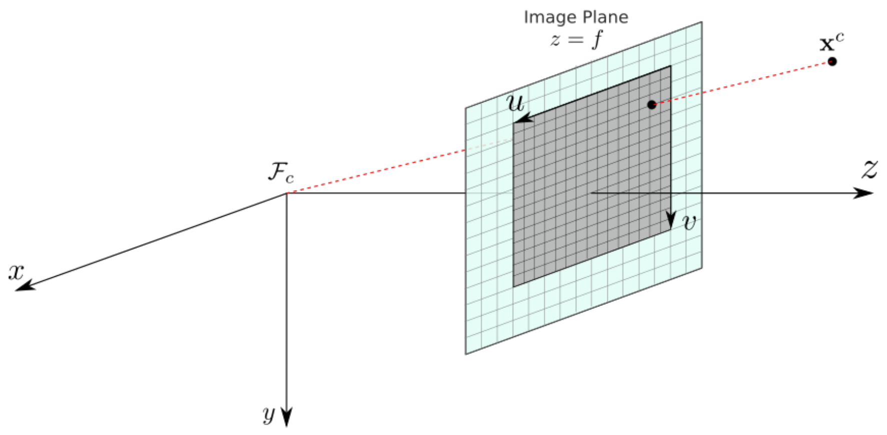

Figure 3.12: Projection of point $\mathbf{x}^c$ onto the image plane

by calibrating a camera. If **K** is known, one can obtain the calibrated normalized image coordinates from the pixel coordinates:

$$\breve{\mathbf{x}}_n = \mathbf{K}^{-1}\breve{\mathbf{u}}$$

Full projection model

Combining the preciding camera theory, one can mathematically project a point in three-dimensional projective space onto the image plane and obtain its pixel coordinates

$$\breve{\mathbf{u}} = \mathbf{K}\frac{1}{z^c}\Pi_0\breve{\mathbf{x}}^c$$

$$\underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\breve{\mathbf{u}}} = \underbrace{\begin{bmatrix} f_u & s_\theta & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \frac{1}{z^c} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\Pi_0} \underbrace{\begin{bmatrix} x^c \\ y^c \\ z^c \\ 1 \end{bmatrix}}_{\breve{\mathbf{x}}^c}$$

The equivalent Euclidean projection function is

$$\mathbf{u} = \pi(\mathbf{x}^c; \mathbf{K}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{K}\frac{1}{z^c}\mathbf{x}^c = \begin{bmatrix} f_u\frac{x^c}{z^c} + c_u \\ f_v\frac{y^c}{z^c} + c_v \end{bmatrix}$$

and the back-projection is

$$\mathbf{x}^c = \pi_p^{-1}(\mathbf{u}, z^c; \mathbf{K}) = z^c \mathbf{K}^{-1} \breve{\mathbf{u}} = z^c \begin{bmatrix} \frac{u-c_u}{f_u} \\ \frac{v-c_v}{f_v} \\ 1 \end{bmatrix}$$

3.6 Probabilistic Estimation

"Estimation is the task of inferring knowledge about an unknown quantity x from data z, which is related to x. In the probabilistic paradigm, the relationship between z and x is in the form of a probabilistic model p(z|x)." [10]. The unknown quantity x is as mentioned in Chapter 2 referred to as the *state* of the system. Two well known probabilistic state estimators are the Maximum Likelihood (ML) estimator and the Maximum A Posteriori (MAP) estimator. The ML estimator is defined as

$$\hat{x} = \arg\max_x p(z|x)$$

If there is some prior knowledge about the state to be estimated, this can be included in the *prior* probability distribution of the state p(x). In this case, the Maximum A Posteriori (MAP) Estimator can be formulated.

$$\begin{aligned} \hat{x} &= \arg\max_x p(x|z) \\ &= \arg\max_x p(z|x)p(x) \end{aligned}$$

If the random variable is Gaussianly distributed, this corresponds to finding the expectation of the distribution $p(x|z)$.

3.6.1 Filtering

The filtering problem consists of estimating the current state of a stochastic, dynamic system from a series of noisy measurements. Thus, it also consists of marginalizing out older states to get the best estimate of the current state of the system, or in mathematical terms: we want to find MAP estimate of the state $\mathbf{x}_k$ such that

$$\hat{\mathbf{x}}_k = \arg\max_{\mathbf{x}_k} p(\mathbf{x}_k|\mathbf{z}_{1:k})$$

The filtering problem is formulated in terms of two models; the process model and the measurement model.

Filtering models

The process model is defined as

$$p(\mathbf{x}_k|\mathbf{x}_{1:k-1}, \mathbf{z}_{1:k})$$

and the measurement model is defined as

$$p(\mathbf{z}_k|\mathbf{x}_{1:k}, \mathbf{z}_{1:k-1})$$

however, when only focusing on filtering, the Markov property is assumed to hold

$$p(\mathbf{x}_k|\mathbf{x}_{1:k-1}, \mathbf{z}_{1:k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$$
$$p(\mathbf{z}_k|\mathbf{x}_{1:k}, \mathbf{z}_{1:k-1}) = p(\mathbf{z}_k|\mathbf{x}_k)$$

3.6.2 Factor graphs and smoothing

As mentioned in the previous section, filtering is marginalizing previous information and finding the best estimate of the most recent state. On the other hand, the *smoothing* problem concerns finding estimates of older states as well. When doing smoothing, one is thus often interested in the full trajectory of the system:

$$\hat{\mathbf{x}}_{1:k} = \arg\max_{\mathbf{x}_{1:k}} p(\mathbf{x}_{1:k}|\mathbf{z}_{1:k})$$

Factor graph

A factor graph is a graphical way of modeling variables and probabilistic dependencies. It is a *bipartite graph*, which means it consists of two types of nodes, *variable nodes* and *factor nodes*. Variable nodes represent states at different timesteps. Factor nodes represent the probabilistic dependencies between the states at the different timesteps, as well as the relationships between states and measurements. This is illustrated in Figure 3.13.

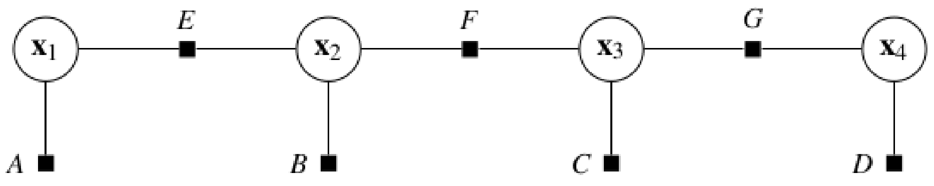

Figure 3.13: A factor graph representing the probabilistic structure of a MAP estimation problem.

3.6.3 On-manifold, nonlinear optimization-based estimation

Consider the case of smoothing, where one has the history of state vectors $\underline{\mathcal{X}}_{1:k}$. For a single state vector $\underline{\mathcal{X}}_i$, some states lie in normal vector spaces $\mathbb{R}^m$ and some states lie on the manifold $\mathcal{M}$.

$$\underline{\mathcal{X}}_i = \begin{bmatrix} \mathbf{x}_i \in \mathbb{R}^m \\ \mathcal{X}_i \in \mathcal{M} \end{bmatrix}$$

By broadening the definition of the manifold, one can claim that the vectors $\mathbf{x} \in \mathbb{R}^m$ also are a Lie group under addition. This implies the following:

$$\begin{aligned} \mathrm{Exp} : \mathbb{R}^m \mapsto \mathbb{R}^m, \quad \mathbf{x} &= \mathrm{Exp}(\mathbf{x}) \\ \mathbf{x}_a \oplus \mathbf{x}_b &= \mathbf{x}_a + \mathbf{x}_b \\ \mathbf{x}_a \ominus \mathbf{x}_b &= \mathbf{x}_a - \mathbf{x}_b \end{aligned}$$

By additionally defining $\underline{\mathcal{M}}$ as the composite manifold which can cover both vector spaces, orientations, poses and other manifolds

$$\underline{\mathcal{M}} = \begin{Bmatrix} \mathcal{M}_0 \\ \vdots \\ \mathcal{M}_i \end{Bmatrix} \tag{3.45}$$

One can say that

$$\underline{\mathcal{X}} \in \underline{\mathcal{M}}$$

Consider then also that one receives measurements $\mathbf{z}_1, \dots, \mathbf{z}_k$ which depends on $\mathcal{X}_0 \dots \mathcal{X}_k$ through the the nonlinear function $h(\underline{\mathcal{X}})$ and are corrupted by white noise. This is mathematically formulated as

$$\begin{aligned} \mathbf{z} &= h(\underline{\mathcal{X}}) + \mathbf{w} \\ \mathbf{w} &\sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}) \\ \mathbf{z} &\in \mathbb{R}^n \\ \underline{\mathcal{X}} &\in \underline{\mathcal{M}} \end{aligned}$$

This implies that

$$\mathbf{z} \sim \mathcal{N}(h(\underline{\mathcal{X}}), \mathbf{\Sigma})$$

Since the noise is white, the measurements $\mathbf{z}_1, \dots, \mathbf{z}_k$ are assumed independent. Thus, they only depend on the underlying state values $\mathcal{X}_1 \dots \mathcal{X}_k$. This implies that

$$\begin{aligned} p(\mathbf{z}_{1:k} | \underline{\mathcal{X}}_{1:k}) &= p(\mathbf{z}_1) p(\mathbf{z}_2) \dots p(\mathbf{z}_k) && (3.47a) \\ &= \mathcal{N}(h(\underline{\mathcal{X}}_1), \mathbf{\Sigma}) \cdot \mathcal{N}(h(\underline{\mathcal{X}}_2), \mathbf{\Sigma}) \dots \mathcal{N}(h(\underline{\mathcal{X}}_k), \mathbf{\Sigma}) && (3.47b) \\ &= \prod_{i=1}^{k} \frac{1}{(2\pi)^{\frac{n}{2}} |\mathbf{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \left(\mathbf{z}_i - h(\underline{\mathcal{X}}_i) \right)^\top \mathbf{\Sigma}^{-1} \left(\mathbf{z}_i - h(\underline{\mathcal{X}}_i) \right) \right) && (3.47c) \end{aligned}$$

The ML estimator of this problem is the value of $\mathcal{X}_{1:k}$ which maximizes Equation (3.47), that is

$$\underline{\mathcal{X}}^*_{1:k} = \arg\max_{\underline{\mathcal{X}}} p(\mathbf{z}_{1:k} | \underline{\mathcal{X}}_{1:k}) \tag{3.48}$$

If one has some prior knowledge of this state, one can incorporate this as $p(\underline{\mathcal{X}}_{1:k})$ in Equation (3.47) and Equation (3.48), which in turn gives the MAP estimator

$$\underline{\mathcal{X}}^*_{1:k} = \arg\max_{\underline{\mathcal{X}}} p(\underline{\mathcal{X}}_{1:k} | \mathbf{z}_{1:k}) = \arg\max_{\underline{\mathcal{X}}} p(\mathbf{z}_{1:k} | \underline{\mathcal{X}}_{1:k}) p(\underline{\mathcal{X}}_{1:k}) \tag{3.49}$$

The maximization in Equation (3.49) is equivalent to minimizing the sum of the *squared Mahalanobis norm*, $||\cdot||^2_{\Sigma}$, between the measurements and the states:

$$\begin{aligned}
\underline{\mathcal{X}}^*_{1:k} &= \arg\min_{\underline{\mathcal{X}}} \sum_{i=1}^{k} \left(\left(\mathbf{z}_i - h(\underline{\mathcal{X}}_i)\right)^\top \mathbf{\Sigma}^{-1} \left(\mathbf{z}_i - h(\underline{\mathcal{X}}_i)\right) \right) && \text{(3.50a)} \\
&= \arg\min_{\underline{\mathcal{X}}} \sum_{i=1}^{k} \left(\mathbf{\Sigma}^{-\frac{1}{2}} \left(\mathbf{z}_i - h(\underline{\mathcal{X}}_i)\right) \right)^\top \left(\mathbf{\Sigma}^{-\frac{1}{2}} \left(\mathbf{z}_i - h(\underline{\mathcal{X}}_i)\right) \right) && \text{(3.50b)} \\
&= \arg\min_{\underline{\mathcal{X}}} \sum_{i=1}^{k} \left\| \mathbf{\Sigma}^{-\frac{1}{2}} \left(\mathbf{z}_i - h(\underline{\mathcal{X}}_i)\right) \right\|^2 && \text{(3.50c)} \\
&= \arg\min_{\underline{\mathcal{X}}} \sum_{i=1}^{k} ||\mathbf{z}_i - h(\underline{\mathcal{X}}_i)||^2_{\Sigma} && \text{(3.50d)}
\end{aligned}$$

Using either Gauss-Newton optimization (Algorithm 1) or Levenberg-Marquardt (Algorithm 2), one can solve the system in Equation (3.50) as a nonlinear optimization problem each time a new measurement arrives.

3.7 Estimating Pose and Structure with a Camera

In Chapter 2, Visual-Inertial Odometry was presented as a simultaneous estimation of pose and construction of the environment model surrounding the pose.

3.7.1 Construction of the model

Consider a very visible landmark, e.g. a black stone in a snowy field. The position of this landmark is given in the world frame $\mathcal{F}_w$ by the three dimensional vector $\mathbf{x}^w \in \mathbb{R}^3$. Consider then a camera with pose $\mathbf{T}_{wc}$ relative to the world frame. The position of the landmark relative to the camera frame $\mathcal{F}_c$ is given by $\mathbf{x}^c$.

$$\breve{\mathbf{x}}^w = \mathbf{T}_{wc}\breve{\mathbf{x}}^c$$

When capturing the image, the position $\mathbf{x}^c$ is projected into the image domain with the projection function $\pi(\cdot)$, and the position in the image domain is given by $\mathbf{u}^c$.

$$\mathbf{u}^c = \pi(\mathbf{x}^c) = \pi(\mathbf{T}_{wc}^{-1}\mathbf{x}^w)$$

Consider then that the camera takes a picture of this landmark from multiple different poses $\mathbf{T}_{wc_1}, \mathbf{T}_{wc_2}, \ldots, \mathbf{T}_{wc_k}$. This produces image projections from multiple views:

$$\begin{gathered}
\mathbf{u}^{c_1} = \pi(\breve{\mathbf{x}}^{c_1}) = \pi(\mathbf{T}_{wc_1}^{-1}\breve{\mathbf{x}}^w) \\
\vdots \\
\mathbf{u}^{c_k} = \pi(\breve{\mathbf{x}}^{c_k}) = \pi(\mathbf{T}_{wc_k}^{-1}\breve{\mathbf{x}}^w)
\end{gathered}$$

Consider now that one has l different landmarks.

$$\mathbf{x}_1^w, \mathbf{x}_2^w, \ldots, \mathbf{x}_l^w$$

If a camera with pose $\mathbf{T}_{wc_1}$ takes a picture of these points, there will be l projections in this camera.

$$\mathbf{u}_1^{c_1}, \mathbf{u}_2^{c_1}, \ldots, \mathbf{u}_l^{c_1}$$

Consider again that the camera captures images of these landmarks from multiple poses $\mathbf{T}_{wc_1}, \mathbf{T}_{wc_2}, \ldots, \mathbf{T}_{wc_k}$. Then there will be l new projections for each picture.

$$\begin{gathered}\mathbf{u}_1^{c_1}, \mathbf{u}_2^{c_1}, \ldots, \mathbf{u}_l^{c_1} \\ \mathbf{u}_1^{c_2}, \mathbf{u}_2^{c_2}, \ldots, \mathbf{u}_l^{c_2} \\ \vdots \\ \mathbf{u}_1^{c_k}, \mathbf{u}_2^{c_k}, \ldots, \mathbf{u}_l^{c_k}\end{gathered}$$

Constructing a model of the environment consists of estimating positions for all these landmarks. See Figure 3.14 for an example of such a model.

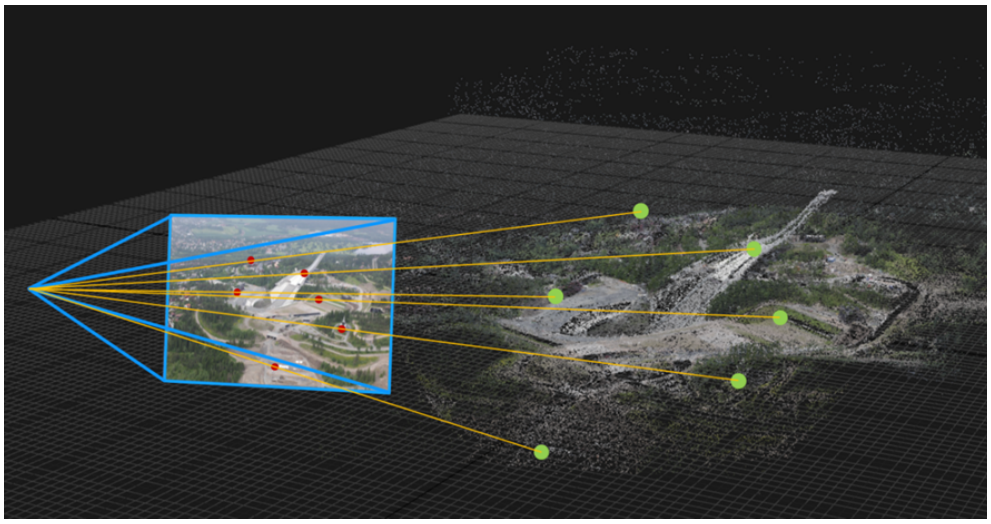

Figure 3.14: A collection of points forming a model of the environment around Holmenkollen. The picture is courtesy of [18].

3.7.2 Bundle adjustment

One can estimate the position of these landmarks, as well as the poses of the camera when taking the pictures, $\mathbf{T}_{wc_1}, \ldots, \mathbf{T}_{wc_k}$ by defining an ML estimation problem, minimizing the *reprojection error*. This is referred to as *bundle adjustment*.

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{i=1}^{l} \sum_{j=1}^{k} \left\| \mathbf{u}_i^{c_j} - h(\mathbf{x}_i^w, \mathbf{T}_{wc_j}) \right\|_{\boldsymbol{\Sigma}_{ij}}^2 \tag{3.51a}$$

$$= \arg\min_{\mathcal{X}} \sum_{i=1}^{l} \sum_{j=1}^{k} \left\| \mathbf{u}_i^{c_j} - \pi(\mathbf{T}_{wc_j}^{-1} \mathbf{x}_i^w) \right\|_{\boldsymbol{\Sigma}_{ij}}^2 \tag{3.51b}$$

$$\mathcal{X} = \begin{bmatrix} \mathbf{x}_1^w & \mathbf{x}_2^w & \dots & \mathbf{x}_l^w & \mathbf{T}_{wc_1} & \dots & \mathbf{T}_{wc_k} \end{bmatrix} \tag{3.51c}$$

Given that the the initial state estimate is close enough to the solution, this can be solved using the nonlinear optimization schemes introduced in Section 3.2.2. The part of the system performing the bundle adjustment is referred to as the backend of the visual-inertial state estimator.

3.8 Frontend and Pre-Processing of Images

Before a bundle adjustment can use data from cameras, the data must be processed into meaningful information. The part of an estimation system handling the data pre-processing is referred to as the *frontend* of the system. In Section 3.7, the position of very visible landmarks in a picture was assumed to be detected, extracted, as well as recognized in several pictures. Methods deploying this type of strategy in the frontend are known as *indirect* or *feature-based* methods. However, contrary to feature-based methods, some methods minimize the pixel intensity differences between images directly to calculate poses. These methods are referred to as *direct methods*.

3.8.1 Feature-based methods

A feature is a point in an image which can be used for determining the pose. Optimally, this should be anchored in a real-world landmark [21]. Feature-based methods use detection schemes to decide which features are good enough to be tracked. Examples of good features are corners, which can be easily tracked throughout images. After deciding on features, they must be found in similar images. To be recognizable, quantitative descriptors are employed. Indirect methods provide robustness to photometric and geometric distortions, but come at the price of higher computational cost and are dependent on the feature extraction step to work.

3.8.2 Direct methods

Direct methods skip the feature extraction step and use the image intensity values directly to optimize the *photometric error*. Consider two images $I_a(\mathbf{u})$ and $I_b(\mathbf{u})$. Given the relative pose $\mathbf{T}_{ab}$ between these two images, one can define the *warp function* w

$$\mathbf{u}^a = w(\mathbf{u}^b, z^b, \mathbf{T}_{ab}) = \pi_p(\mathbf{T}_{ab} \cdot \pi_p^{-1}(\mathbf{u}^b, z^b))$$

This function maps a pixel in I_b to I_a. The photometric error to be minimized is then given by

$$e_p(\mathbf{u}^b, z^b, \mathbf{T}_{ab}) = I_a(w(\mathbf{u}^b, z^b, \mathbf{T}_{ab})) - I_b(\mathbf{u}^b)$$

Where z is the depth of the pixel at the coordinate $\mathbf{u}$. Direct methods typically represent structure as a sparse or dense map rather than building it with three-dimensional points. They do not require that geometrical primitives are recognisable by themselves, but can sample across all parts of the image. This makes direct methods more robust to effects such as motion blur and better suited in sparsely textured environments, but they are vulnerable to photometric and geometric distortions.

3.8.3 Semi-direct methods

Some methods do not directly fit the description of using a fully direct or indirect frontend but use some elements from each. These methods fall under the description "Semi-Direct".

Chapter 4

Related Work

To choose methods fitting for the tasks defined in Chapter 1, a literature review was conducted. This was partially conducted in the fall of 2021 for the specialization project and partially in the spring 2022 for the masters thesis.

4.1 Visual and Visual-Inertial Odometry

There is a wide variety of techniques for visual-inertial state estimation. This section will present some of the most prominent and known methods.

4.1.1 OKVIS: Open Keyframe-based Visual-Inertial SLAM

Open Keyframe-based Visual-Inertial SLAM (OKVIS) [22][23] is a visual-inertial framework for Odometry and SLAM using a feature-based frontend. The keypoints are detected using a SSE-optimized multiscale Harris Corner Detector [24] and a Binary Robust Invariant Scalable Keypoints (BRISK) [25] keypoint descriptor. In a new photo, keypoints are first matched against all other keypoints in the local map of maintained features, followed by outlier rejection using a χ^2-test. In the final step, the bundle adjustment is performed. However, to bound computational complexity, this is not performed over all pictures and landmarks, but only selected *keyframes*. The bundle adjustment is solved with the Ceres solver [26].

4.1.2 DSO: Direct Sparse Odometry

Direct Sparse Odometry (DSO) is a visual odometry method based on a sparse and direct structure from motion formulation, and can be considered the direct methodology counterpart to OKVIS[27][28]. It combines a fully direct probabilistic model (minimizing a photometric error) with consistent, joint optimization of all model parameters. The method does not depend on keypoint detectors or descriptors. Therefore, it can naturally sample pixels from across all image regions that have intensity gradients. This includes edges and smooth intensity variations on essentially featureless walls. The model includes a full photometric calibration accounting for exposure time, lens vignetting and non-linear response functions.

4.1.3 ROVIO: Robust Visual-Inertial Odometry

Robust Visual-Inertial Odometry (ROVIO) [29][30] is a Monocular Visual Inertial Odometry Algorithm based on a semi-direct frontend and an Extended Kalman Filter (EKF) backend. It uses direct pixel intensities errors of multilevel image patches to describe landmarks instead of features. After detecting landmarks, the tracking of multilevel patch features is closely coupled to the underlying EKF by directly using the intensity errors as innovation term during the update step. The position of the 3D-Landmarks is estimated w.r.t the current camera pose. As ROVIO uses an EKF backend, the algorithm is among the simpler ones in terms of technical aspects. Nevertheless, this is a robust technique that works well in low lighting, even in the absence of corners.

4.1.4 SVO: Semi-direct visual odometry

Semi-direct Visual Odometry (SVO)[31][32] is a visual odometry method with a semi-direct frontend for Monocular and Multi-Camera Systems. Several versions are available, but the most tested and mature version uses a keyframe-based backend, just like OKVIS. Once features are extracted, a direct method is applied to track features. Sparse image alignment estimates frame-to-frame by minimizing the intensity difference of features corresponding to the projected location of the same 3D point in several pictures. A subsequent step relaxes the geometric constraint to obtain sub-pixel feature correspondence. This step introduces a reprojection error which is finally refined by solving the bundle adjustment problem. The code is open source and updated to support the latest software.

4.1.5 VINS-Fusion: Optimization-based multi-sensor state estimator

VINS-Mono is a monocular Visual-Inertial System (VINS), known for being robust and versatile [33]. It has an indirect frontend and a keyframe-based backend, very similar to OKVIS. The robustness and versitility originiate from the full functionality from measurement preprocessing, sensor calibration, estimator initialization, tightly coupled VIO, relocalization and global pose graph optimization. In VINS-Fusion, VINS-Mono was extended to support stereo cameras, in addition to a monocular camera. This method is also open source and updated to support the latest software.

4.2 Observability

An observable state is a physical quantity determined with the available sensor data. On the other hand, when a state is unobservable, there are infinitely many different states which could produce the observed sensor data.

4.2.1 Constrained motion

Martinelli showed analytically that the following states are all observable modes during motion which excites all degrees of freedom[34]:

- Absolute roll, ϕ
- Absolute pitch, θ
- Three-dimensional speed given in the body-frame, $\mathbf{v}^b$
- Scale, λ
- The biases of the IMU, $\mathbf{b}_a$ and $\mathbf{b}_g$

This leaves the following unobservable modes

- Absolute position in the global frame
- Absolute yaw, ψ

As mentioned previously, the spatial trajectories of a small Unmanned Aerial Vehicle (UAV) are quite different from the trajectories an autonomous ship produces. Intuitively, it should be a much easier trajectory to estimate, as the movements are slow and predictable. However, on the contrary, in [34] it was proved that constant speed renders scale λ unobservable. Further investigations in [35] found that for planar motion, moving in straight lines, that is no rotation, renders the pitch θ and roll ϕ unobservable as well.

4.3 Fusion of Global and Local Data

When performing state estimation, there is a difference between what is referred to as tightly and loosely coupled fusion of data.

4.3.1 Tightly coupled estimation

A tightly coupled estimation framework will model all sensor modalities and use their raw inputs into the bundle adjustment problem. The reasoning behind it is to best capture all correlations between measurements, additionally to bringing a minimal amount of estimation error into the problem from the beginning. The bundle adjustment would then typically look like Equation (4.1).

$$\mathbf{x} = \arg\min_{\mathbf{x}} = \sum_{\mathbf{z}_{local}} \|\mathbf{z}_{local} - h_{local}(\mathbf{x})\|_{\Sigma}^2 + \sum_{\mathbf{z}_{global}} \left\|\mathbf{z}_{global} - h_{global}(\mathbf{x})\right\|_{\Sigma}^2 \tag{4.1}$$

Applying a tightly coupled approach typically give higher accuracy in terms of estimation error.

Approaches using tight coupling

GVINS [36] is an example of a tightly coupled fusion of local and global measurements. This method is built on top of the visual-inertial odometry estimator VINS-Mono, with only a slight variation for marginalizing older states. Another method having tight coupling between VIO and GNSS is "Tight-coupling of global positional measurements into VIO"[37]. In this paper, SVO was used as the frontend.

4.3.2 Loosely coupled estimation

Loosely coupled state estimation will not model sensor modalities in the bundle adjustment. Instead, solutions from previous state estimation problems are included together in a new bundle adjustment

$$\begin{aligned}
\mathbf{x}^*_{global} &= \arg\min_{\mathbf{x}} \sum_{\mathbf{z}_{global}} \left\| \mathbf{z}_{global} - h_{global}(\mathbf{x}) \right\|^2_{\Sigma} \\
\mathbf{x}^*_{local} &= \arg\min_{\mathbf{x}} \sum_{\mathbf{z}_{local}} \| \mathbf{z}_{local} - h_{local}(\mathbf{x}) \|^2_{\Sigma} \\
\mathbf{x}^*_{fused} &= \sum \left\| \mathbf{x}_{fused} - \mathbf{x}^*_{global} \right\|^2_{\Sigma} + \sum \left\| \mathbf{x}_{fused} - \mathbf{x}^*_{local} \right\|^2_{\Sigma}
\end{aligned}$$

This is faster, as it can be parallelized, at the cost of generally having worse performance than tight coupling w.r.t accuracy. However, for the fusion of local and global sensors, a clear advantage of being loosely coupled is that it gives options in regards to calculating local odometry. VINS-Fusion offers a loosely coupled solution for this matter [38], where the positional estimates from the GNSS are used in tandem with a six-degrees-of-freedom pose estimate from an arbitrary method for fusing local sensors.

Chapter 5

Global Pose Estimation: VINS-Fusion

For fusion of local and global odometry estimates, VINS-Fusion was chosen, due to the advantages presented in Chapter 4. A sketch of the system structure for VINS-Fusion is visualized in Figure 5.1. In this chapter the fusion of local and global odometry estimates is presented in detail.

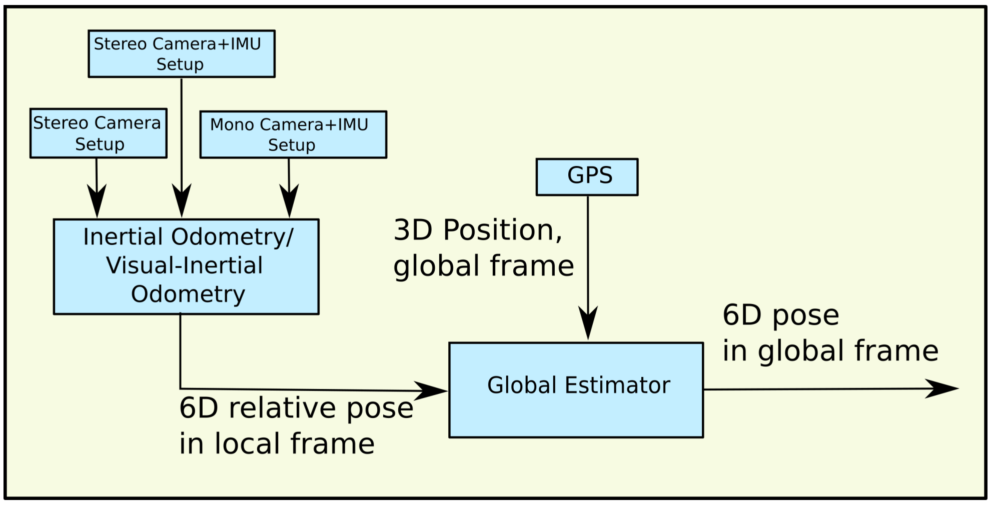

Figure 5.1: An overview of the VINS-Fusion framework

5.1 Loosely Coupled Global Pose Estimation

The estimation problem for global pose is formulated as a bundle adjustment

$$
\begin{aligned}
\mathcal{X}^* = \arg\min_{\mathcal{X}} = &\sum_{t=0}^{n} \left\| \mathbf{z}_t^{VIO} - h_t^{VIO}(\mathcal{X}) \right\|_{\Sigma_{k,t}}^2 \\
&+ \sum_{t=0}^{n} \left\| \mathbf{z}_t^{GNSS} - h_t^{GNSS}(\mathcal{X}) \right\|_{\Sigma_{k,t}}^2 \\
\mathcal{X} &= \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n\} \\
\mathbf{x}_t &= \{\mathbf{p}_{b_t}^{w}, \mathbf{R}_{wb_t}\}
\end{aligned}
$$

Where $\mathbf{p}_{b_t}^{w}$ is the position of the body frame $\mathcal{F}_b$ relative to the world frame $\mathcal{F}_w$ at timestep t, and $\mathbf{R}_{wb_t}$ is its orientation. The probabilistic model underlying this optimization problem can be formulated as a factor graph. In this factor graph, the probabilistic connections between the states are represented as factor nodes. This is illustrated in Figure 5.2

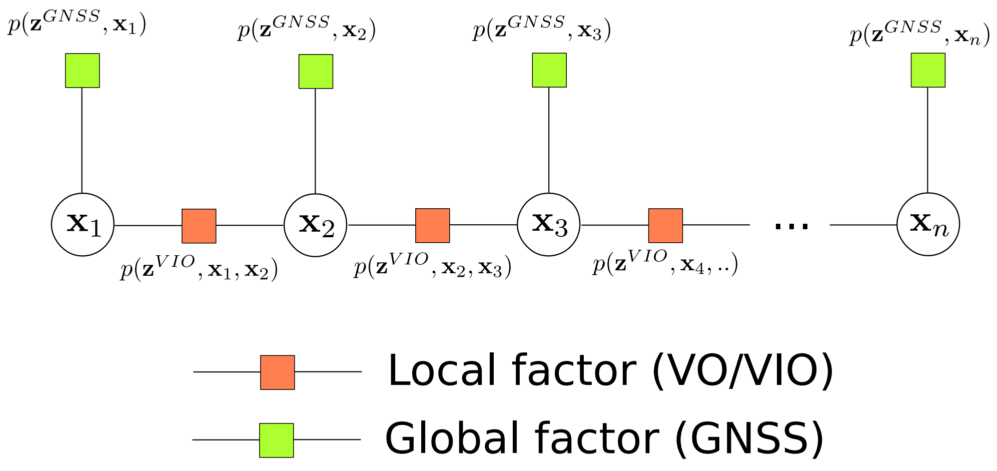
$p(\mathbf{z}^{GNSS}, \mathbf{x}_1)$
$p(\mathbf{z}^{GNSS}, \mathbf{x}_2)$
$p(\mathbf{z}^{GNSS}, \mathbf{x}_3)$
$p(\mathbf{z}^{GNSS}, \mathbf{x}_n)$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
...
$\mathbf{x}_n$
$p(\mathbf{z}^{VIO}, \mathbf{x}_1, \mathbf{x}_2)$
$p(\mathbf{z}^{VIO}, \mathbf{x}_2, \mathbf{x}_3)$
$p(\mathbf{z}^{VIO}, \mathbf{x}_4, ..)$
Local factor (VO/VIO)
Global factor (GNSS)

Figure 5.2: Factor graph representation of probabilistic model underlying the optimization problem in the global pose estimation

5.1.1 Local factors from visual-inertial odometry

As mentioned earlier, the framework assumes that the VIO produces pose estimates relative to a local frame, which is the initial position assumed in the vio. The local frame is denoted $\mathcal{F}_l$.

Considering two sequential time steps $t-1$ and t, the local factor is defined as

$$\begin{aligned}
&\mathbf{z}_t^{VIO} - h^{VIO}(\mathcal{X}) \\
&= \mathbf{z}_t^{VIO} - h^{VIO}(\mathbf{x}_{t-1}, \mathbf{x}_t) \\
&= \mathbf{z}_t^{VIO} - h^{VIO}(\{\mathbf{p}_{b_{t-1}}^w, \mathbf{R}_{wb_{t-1}}\}, \{\mathbf{p}_{b_t}^w, \mathbf{R}_{wb_t}\}) \\
&= \begin{bmatrix} \mathbf{R}_{lb_{t-1}}^{-1}(\mathbf{p}_{b_t}^l - \mathbf{p}_{b_{t-1}}^l) \\ \mathbf{R}_{lb_{t-1}}^{-1}\mathbf{R}_{lb_t} \end{bmatrix} \ominus \begin{bmatrix} \mathbf{R}_{wb_{t-1}}^{-1}(\mathbf{p}_{b_t}^w - \mathbf{p}_{b_{t-1}}^w) \\ \mathbf{R}_{wb_{t-1}}^{-1}\mathbf{R}_{wb_t} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{p}_{b_t}^{b-1} \\ \mathbf{R}_{b_{t-1}b_t} \end{bmatrix} \ominus \begin{bmatrix} \mathbf{R}_{wb_{t-1}}^{-1}(\mathbf{p}_{b_t}^w - \mathbf{p}_{b_{t-1}}^w) \\ \mathbf{R}_{wb_{t-1}}^{-1}\mathbf{R}_{wb_t} \end{bmatrix}
\end{aligned}$$

The factor graph containing the two states $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ and their probabilistic relationship can be seen in Figure 5.3.

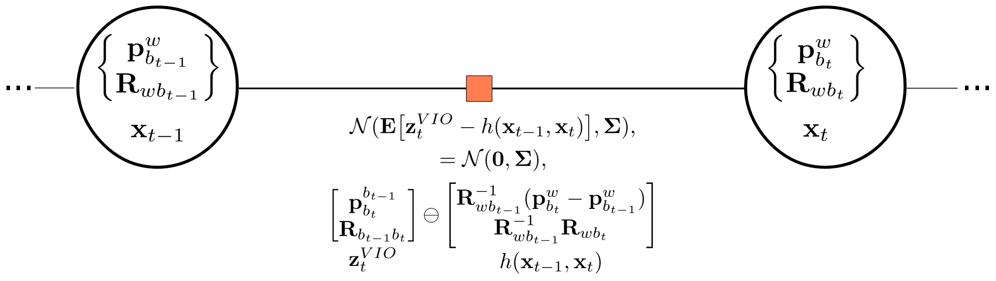
$\begin{Bmatrix} \mathbf{p}_{b_{t-1}}^w \\ \mathbf{R}_{wb_{t-1}} \end{Bmatrix}$
$\mathbf{x}_{t-1}$
$\begin{Bmatrix} \mathbf{p}_{b_t}^w \\ \mathbf{R}_{wb_t} \end{Bmatrix}$
$\mathbf{x}_t$
$\mathcal{N}(\mathbf{E}[\mathbf{z}_t^{VIO} - h(\mathbf{x}_{t-1}, \mathbf{x}_t)], \mathbf{\Sigma}),$
$= \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}),$
$\underbrace{\begin{bmatrix} \mathbf{p}_{b_t}^{b_{t-1}} \\ \mathbf{R}_{b_{t-1}b_t} \end{bmatrix}}_{\mathbf{z}_t^{VIO}} \ominus \underbrace{\begin{bmatrix} \mathbf{R}_{wb_{t-1}}^{-1}(\mathbf{p}_{b_t}^w - \mathbf{p}_{b_{t-1}}^w) \\ \mathbf{R}_{wb_{t-1}}^{-1}\mathbf{R}_{wb_t} \end{bmatrix}}_{h(\mathbf{x}_{t-1}, \mathbf{x}_t)}$

Figure 5.3: Factor graph relating the states $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ with VIO-output as a measurement

5.1.2 Global factors from GNSS measurements

The GNSS measurements are received from a GPS. These consists of longitude, latitude and altitude. An East-North-Up frame is then initalized with its origin set as the first received GNSS measurements. The factor relating the GPS measurements to the states are then given by

$$\mathbf{p}_t^{GPS} = \begin{bmatrix} x_t^w & y_t^w & z_t^w \end{bmatrix}^\top$$
$$\mathbf{z}_t^{GPS} - h_t^{GPS}(\mathbf{x}_t)$$
$$\mathbf{p}_t^{GPS} - \mathbf{p}_t^w$$

Graphically, this is stated as in Figure 5.4

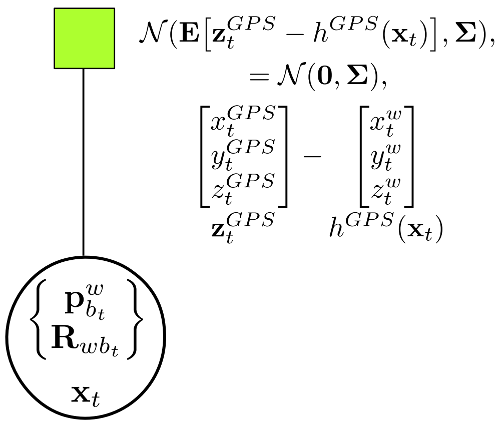
$\mathcal{N}(\mathbf{E}[\mathbf{z}_t^{GPS} - h^{GPS}(\mathbf{x}_t)], \mathbf{\Sigma}),$
$= \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}),$
$\begin{bmatrix} x_t^{GPS} \\ y_t^{GPS} \\ z_t^{GPS} \end{bmatrix} - \begin{bmatrix} x_t^w \\ y_t^w \\ z_t^w \end{bmatrix}$
$\mathbf{z}_t^{GPS}$
$h^{GPS}(\mathbf{x}_t)$
$\begin{Bmatrix} \mathbf{p}_{b_t}^w \\ \mathbf{R}_{wb_t} \end{Bmatrix}$
$\mathbf{x}_t$

Figure 5.4: Probabilistic relation between state and GPS measurement

5.1.3 Optimization

Once the optimization problem is formed, it is solved once every second, using the Ceres solver [26]. By doing this, the transformation between the global frame $\mathcal{F}_w$ and the frame of the last pose $\mathcal{F}_{b_t}$, $\mathbf{T}_{wb_t}$ is calculated. This allows using the poses calculated by the VIO in real time while they arrive. When the optimization problem gets large, the oldest states are marginalized out.

Chapter 6

Local Pose Odometry Estimation: VINS-Fusion and SVO

In Chapter 4, some of the most prominent methods for producing visual-inertial odometry were presented. In this thesis, however, to test state-of-the-art visual-inertial odometry methods in maritime settings, the choice was narrowed down to Semi-direct Visual Odometry (SVO) and VINS-Fusion. In this chapter, the reasoning behind this, as well as how these methods work, will be reviewed.

6.1 VINS-Fusion

The visual-inertial odometry estimator bundled with VINS-Fusion is a modified version of VINS-Mono, a state estimator known for being robust and versatile. It is modified to support the configuration stereo camera setup, in addition to a monocular camera setup and an IMU. To reiterate Chapter 1, this gives three sensor combinations: monocular camera and IMU, stereo camera and IMU, as well as stereo cameras alone. VINS-Fusion has a feature-based frontend and a keyframe-based backend. A keyframe-based backend means it smooths the states involved in these keyframes, as well as the states in a little window of the most recent frames. The estimator also has a powerful loop-closure feature, however as GNSS measurements will serve to eliminate drift, this is disregarded.

6.1.1 Estimated states

The full state vector to be estimated comprises the following

$$\begin{aligned}
\mathcal{X} &= [\mathbf{x}_0, \dots, \mathbf{x}_n, \mathbf{x}_1^{imu}, \dots, \mathbf{x}_n^{imu}, \mathbf{x}^{cam}], \\
\mathbf{x}_t &= [\mathbf{p}_{b_t}^{w}, \mathbf{R}_{wb_t}] \\
\mathbf{x}_t^{imu} &= [\mathbf{v}_{b_t}^{w}, \mathbf{b}_a, \mathbf{b}_g] \\
\mathbf{x}^{cam} &= [\lambda_0, \dots, \lambda_l]
\end{aligned}$$

The superscript w here denotes $\mathcal{F}_l$, which is the local world frame to the estimation problem, and $\mathcal{F}_{b_t}$ is the body frame at timestep t. The terms in the vector is defined as

- $\mathbf{p}^w_{b_t}$: Position of $\mathcal{F}_b$ at timestep t w.r.t $\mathcal{F}_l$ expressed in $\mathcal{F}_l$
- $\mathbf{v}^w_{b_k}$: Velocity of $\mathcal{F}_b$ at timestep t w.r.t $\mathcal{F}_l$ expressed in $\mathcal{F}_l$
- $\mathbf{R}_{wb_t}$: Orientation of $\mathcal{F}_b$ w.r.t $\mathcal{F}_l$ at timestep t
- $\mathbf{b}_a$: Bias of the accelerometer in the IMU
- $\mathbf{b}_g$: Bias of gyroscope in the IMU
- λ_l: The inverse distance of the lth feature from its first observation

6.1.2 Vision preprocessing frontend

Feature quality in tracking is detected and determined using the Shi-Tomasi-score introduced in Good Features to Track [21]. This gives a set of coordinates for the most informative features for tracking. Existing features are tracked frame-to-frame using the KLT sparse optical flow algorithm [39]. KLT optical flow calculates a displacement vector for each feature, such that one obtains the feature coordinates in the next frame. If stereo cameras are used, this technique is also used to find feature correspondences between the stereo cameras. Outlier rejection is done using RANSAC with a fundamental matrix model [40].

6.1.3 IMU preintegration

IMU measurements come at a much higher rate than camera images, and performing the optimization every time a new IMU measurement arrives will not be possible. Because of this, a technique referred to as *IMU preintegration* is widely used in the VIO literature, and VINS-Fusion is no exception.

Integration

The measurements originating from the IMU, which is not corrected w.r.t biases and noise, is denoted $\hat{\mathbf{a}}_t$ and $\hat{\boldsymbol{\omega}}_t$ for acceleration and angular rates respectively. Consider the body frame $\mathcal{F}_{b_t}$ at timestep t and $\mathcal{F}_{b_{t+1}}$ at timestep $t+1$. The change in the position, velocity and orientation between t and $t+1$ can be calculated by integrating the measurements during the time interval $[t, t+1]$:

$$\begin{aligned}
\boldsymbol{\alpha}^{b_t}_{b_{t+1}} &= \int\int_{\tau\in[t,t+1]} \mathbf{R}_{b_t b_\tau}(\hat{\mathbf{a}}_\tau - \mathbf{b}_{a_\tau})d\tau^2 \\
\boldsymbol{\beta}^{b_t}_{b_{t+1}} &= \int_{\tau\in[t,t+1]} \mathbf{R}_{b_t b_\tau}(\hat{\mathbf{a}}_\tau - \mathbf{b}_{a_\tau})d\tau \\
\boldsymbol{\gamma}^{b_t}_{b_{t+1}} &= \int_{\tau\in[t,t+1]} \frac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_\tau - \mathbf{b}_{\omega_\tau})\boldsymbol{\gamma}^{b_t}_{b_\tau} d\tau \\
\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_\tau - \mathbf{b}_{\omega_\tau}) &= \begin{bmatrix} -[\hat{\boldsymbol{\omega}}_\tau - \mathbf{b}_{\omega_\tau}]^\times & \hat{\boldsymbol{\omega}}_\tau - \mathbf{b}_{\omega_\tau} \\ -(\hat{\boldsymbol{\omega}}_\tau - \mathbf{b}_{\omega_\tau})^\top & 0 \end{bmatrix}
\end{aligned}$$

Bias correction

The states $\boldsymbol{\alpha}^{b_t}_{b_{t+1}}, \boldsymbol{\beta}^{b_t}_{b_{t+1}}$ and $\boldsymbol{\gamma}^{b_t}_{b_{t+1}}$ does not directly depend on the underlying position, velocity and orientation. They do, however, depend on the IMU bias. If the biases changes considerably, the preintegrated states must be recalculated. However, if they only change marginally, the states can be modifed using their first-order approximations

$$\begin{aligned}
\boldsymbol{\alpha}^{b_t}_{b_{t+1}} &\approx \hat{\boldsymbol{\alpha}}^{b_t}_{b_{t+1}} + \mathbf{J}^{\alpha}_{b_a}\delta\mathbf{b}_{a_t} + \mathbf{J}^{\alpha}_{b_\omega}\delta\mathbf{b}_{\omega_t} \\
\boldsymbol{\beta}^{b_t}_{b_{t+1}} &\approx \hat{\boldsymbol{\beta}}^{b_t}_{b_{t+1}} + \mathbf{J}^{\beta}_{b_a}\delta\mathbf{b}_{a_t} + \mathbf{J}^{\beta}_{b_\omega}\delta\mathbf{b}_{\omega_t} \\
\boldsymbol{\gamma}^{b_t}_{b_{t+1}} &\approx \hat{\boldsymbol{\gamma}}^{b_t}_{b_{t+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}^{\gamma}_{b_\omega}\delta\mathbf{b}_{\omega_t} \end{bmatrix}
\end{aligned}$$

6.1.4 Keyframe-based factor graph backend

The estimation is done by performing bundle adjustment on selectd keyframes. Just like in Chapter 5, the residuals in the optimization problem are equivalent to factors in a factor graph.

Camera factors

The sensor model for the lth feature which is defined as

$$\begin{aligned}
h(\mathcal{X}^{cam}_l) &= \pi_c\left(\mathbf{T}_{cb}\mathbf{T}_{b_t w}\mathbf{T}_{w b_i}\mathbf{T}_{bc}\pi^{-1}\left(\lambda_l, \begin{bmatrix} u^l_i \\ v^l_i \end{bmatrix}\right)\right) \\
\mathcal{X}^{cam}_l &= [\mathbf{R}_{wb_i}, \mathbf{p}^w_{b_i}, \mathbf{R}_{wb_t}, \mathbf{p}^w_{b_t}, \lambda_l]
\end{aligned}$$

Where the frame $\mathcal{F}_c$ is the camera frame, $\mathcal{F}_{b_i}$ is the frame of the body at the timestep where the lth feature was first observed and $\mathcal{F}_{b_t}$ is the body frame at a later time t. Thus, the

$$\left\|\mathbf{z}_t^l - h(\mathcal{X}_l^{cam})\right\|_{\Sigma}^2$$

$$\left\|\mathbf{z}_t^l - h_t^l(\mathbf{R}_{wb_i}, \mathbf{p}_{b_i}^w, \mathbf{R}_{wb_t}, \mathbf{p}_{b_t}^w, \lambda_l)\right\|_{\Sigma}^2 =$$

$$\left\|\begin{bmatrix} u_t^l \\ v_t^l \end{bmatrix} - \pi_c\left(\mathbf{T}_{cb}\mathbf{T}_{b_t w}\mathbf{T}_{wb_i}\mathbf{T}_{bc}\pi^{-1}\left(\lambda_l, \begin{bmatrix} u_i^l \\ v_i^l \end{bmatrix}\right)\right)\right\|_{\Sigma}^2$$

A term like this is made for every subsequent keyframe where the lth feature is visible. Putting this under a sum gives

$$\sum_{t\in\mathcal{T}}\left\|\begin{bmatrix} u_t^l \\ v_t^l \end{bmatrix} - \pi_c\left(\mathbf{T}_{cb}\mathbf{T}_{b_t w}\mathbf{T}_{wb_i}\mathbf{T}_{bc}\pi^{-1}\left(\lambda_l, \begin{bmatrix} u_i^l \\ v_i^l \end{bmatrix}\right)\right)\right\|_{\Sigma}^2$$

Where $\mathcal{T}$ is the set of keyframes after $\mathcal{F}_i$ where the lth feature is visible. The covariance matrix $\mathbf{\Sigma}$ is the uncertainty of the feature in the image and is a constant value. The complete residual related to every feature is given by

$$\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\left\|\begin{bmatrix} u_t^l \\ v_t^l \end{bmatrix} - \pi_c\left(\mathbf{T}_{cb}\mathbf{T}_{b_t w}\mathbf{T}_{wb_i}\mathbf{T}_{bc}\pi^{-1}\left(\lambda_l, \begin{bmatrix} u_i^l \\ v_i^l \end{bmatrix}\right)\right)\right\|_{\Sigma}^2$$

Where the set $\mathcal{K}$ is the entire set of keyframes and $\mathcal{I}$ is the set of features which is first observed in keyframe k.

IMU Factors

As described in Section 6.1.3, IMU measurements are preintegrated between two camera images. Consider the two timesteps $t-1$ and t. The sensor model for a preintegrated IMU measurement arriving at time t is then defined as

$$h(\mathcal{X}_t^{IMU}) = \begin{bmatrix} \mathbf{R}_{b_{t-1}w}\left(\mathbf{p}_{b_t}^w - \mathbf{p}_{b_{t-1}}^w + \frac{1}{2}\mathbf{g}d\tau^2 - \mathbf{v}_{b_{t-1}}d\tau\right) \\ \mathbf{R}_{b_{t-1}w}\left(\mathbf{v}_{b_t}^w - \mathbf{v}_{b_{t-1}}^w + \mathbf{g}d\tau\right) \\ \mathbf{R}_{b_{t-1}w}\mathbf{R}_{b_t w} \\ \mathbf{b}_{a_t} - \mathbf{b}_{a_{t-1}} \\ \mathbf{b}_{\omega_t} - \mathbf{b}_{\omega_{t-1}} \end{bmatrix}$$

and thus the residual is given by

$$\left\|\underbrace{\begin{bmatrix} \boldsymbol{\alpha}_{b_t}^{b_{t-1}} \\ \boldsymbol{\beta}_{b_t}^{b_{t-1}} \\ \boldsymbol{\gamma}_{b_t}^{b_{t-1}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{z}_t^{IMU}} \ominus \underbrace{\begin{bmatrix} \mathbf{R}_{b_{t-1}w}\left(\mathbf{p}_{b_t}^w - \mathbf{p}_{b_{t-1}}^w + \frac{1}{2}\mathbf{g}d\tau^2 - \mathbf{v}_{b_{t-1}}d\tau\right) \\ \mathbf{R}_{b_{t-1}w}\left(\mathbf{v}_{b_t}^w - \mathbf{v}_{b_{t-1}}^w + \mathbf{g}d\tau\right) \\ \mathbf{R}_{b_{t-1}w}\mathbf{R}_{b_t w} \\ \mathbf{b}_{a_t} - \mathbf{b}_{a_{t-1}} \\ \mathbf{b}_{\omega_t} - \mathbf{b}_{\omega_{t-1}} \end{bmatrix}}_{h(\mathcal{X}_t^{IMU})}\right\|_{\Sigma}^2$$

Marginalization

As mentioned early in this section, the bundle adjustment is not performed over all states, but only on selected keyframes. As removing states directly is equivalent to conditioning on the current estimates of the states, this can lead to overconfidence. Thus, states are marginalized out. Consider the bundle adjustment

$$
\begin{aligned}
\arg\min_{\mathcal{X}} \sum_{k\in\mathcal{K}} \left(\left(\sum_{m\in\mathcal{M}} \left\| \mathbf{z}_m^{IMU} - h_m^{IMU}(\mathcal{X}_t^{IMU}) \right\|_{\Sigma}^2 \right) + \left(\sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}} \left\| \mathbf{z}_t^{l} - h(\mathcal{X}_l^{cam}) \right\|_{\Sigma}^2 \right) \right) \\
= \arg\min_{\mathcal{X}} \sum\sum \left\| \mathbf{z}_t^s - h_t^s(\mathcal{X}_t^s) \right\|_{\Sigma_t^s} \\
= \arg\min_{\mathcal{X}} \sum\sum \left\| \boldsymbol{\Sigma}^{-\frac{1}{2}} (\mathbf{z}_t^s - h_t^s(\mathcal{X}_t^s)) \right\|^2 \\
= \arg\min_{\mathcal{X}} \sum\sum \left\| \boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{e}_t^s \right\| \\
\approx \arg\min_{\delta\mathcal{X}} \sum_{t\in\mathcal{K}} \sum_{s\in\mathcal{S}} \left\| \boldsymbol{\Sigma}^{-\frac{1}{2}} (\mathbf{e}_t^s + \mathbf{J}_t^s \delta\mathcal{X}) \right\|^2 \\
= \arg\min_{\delta\mathcal{X}} \sum\sum \left\| (\boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{e}_t^s + \boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{J}_t^s \delta\mathcal{X}) \right\|^2
\end{aligned}
$$

The normal equations are then given by

$$
\begin{aligned}
2\sum\sum (\boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{J}_t^s)^\top (\boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{e}_t^s + \boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{J}_t^s \delta\mathcal{X}) = \mathbf{0} \\
\sum\sum ((\mathbf{J}_t^s)^\top \boldsymbol{\Sigma}^{-1} \mathbf{e}_t^s + (\mathbf{J}_t^s)^\top \boldsymbol{\Sigma}^{-1} \mathbf{J}_t^s \delta\mathcal{X}) = \mathbf{0}
\end{aligned}
$$

Which is equivalent to the linear system

$$
\Rightarrow \underbrace{\left[\sum\sum (\mathbf{J}_t^s)^\top \boldsymbol{\Sigma}_t^s \mathbf{J}_t^s \right]}_{\mathbf{H}} \delta\mathcal{X} = \underbrace{\left[-\sum\sum (\mathbf{J}_t^s)^\top \boldsymbol{\Sigma}_t^s \mathbf{e}_t^s \right]}_{\mathbf{b}} \tag{6.1}
$$

To go from all the states $\mathcal{X}_{1:n}$ down to $\mathcal{X}_{m:n}$, one can split the linear system into the states and matrices related to $\mathcal{X}_m = \mathcal{X}_{1:m-1}$, the states which are to be marginalized out, and the remaining states $\mathcal{X}_r = \mathcal{X}_{m:n}$

$$
\begin{bmatrix} \mathbf{H}_{mm} & \mathbf{H}_{mr} \\ \mathbf{H}_{rm} & \mathbf{H}_{rr} \end{bmatrix} \begin{bmatrix} \delta\mathcal{X}_m \\ \delta\mathcal{X}_r \end{bmatrix} = \begin{bmatrix} \mathbf{b}_m \\ \mathbf{b}_r \end{bmatrix}
$$

This is a randomly distributed system described in canonical form by

$$
\begin{bmatrix} \delta\mathcal{X}_m \\ \delta\mathcal{X}_r \end{bmatrix} \sim \mathcal{N}^{-1} \left(\begin{bmatrix} \mathbf{b}_m \\ \mathbf{b}_r \end{bmatrix}, \begin{bmatrix} \mathbf{H}_{mm} & \mathbf{H}_{mr} \\ \mathbf{H}_{rm} & \mathbf{H}_{rr} \end{bmatrix} \right)
$$

and using knowledge from Section 3.1.6, the information related to $\delta\mathcal{X}_m$ can be marginalized into a prior for $\mathcal{X}_r$:

$$
\begin{aligned}
\delta\mathcal{X}_r \sim \mathcal{N}^{-1}(\mathbf{b}_p, \mathbf{H}_p) \\
= \mathcal{N}^{-1} \left(\mathbf{b}_r - \mathbf{H}_{rm} \mathbf{H}_{mm}^{-1} \mathbf{b}_m, \mathbf{H}_{rr} - \mathbf{H}_{rm} \mathbf{H}_{mm}^{-1} \mathbf{H}_{mr} \right)
\end{aligned}
$$

The bundle adjustment problem is then reduced to

$$\mathcal{X}^*_{m:n} = \arg\min_{\mathcal{X}_{m:n}} \left(\left(\sum \sum \left\| \mathbf{z}^s_t - h^s_t(\mathcal{X}_{m:n}) \right\|^2_{\Sigma^s_{m:n}} \right) + \left(\mathbf{H}_p \delta \mathcal{X}_{m:n} - \mathbf{b}_p \right) \right)$$

6.1.5 Reasoning for choice of VINS-Fusion

As VINS-Fusion was chosen for GNSS-fusion, the bundled visual-inertial estimator was a natural choice due to compatibility. Additionally, VINS-Fusion supports all the necessary sensor combinations and has an up-to-date open-source code. According to the review of monocular visual-inertial methods in [41] and [42], VINS-Fusion is also one of the top performers in regards to accuracy and robustness, and thus it was a natural choice.

6.2 SVO: Visual-Inertial Odometry Using a Semi-Direct Frontend

Originally a semi-direct visual odometry frontend, Semi-direct Visual Odometry (SVO)[31][32][43], is now a widely used visual-inertial odometry framework. As stated in Chapter 4, there have been made several versions of SVO utilizing different backends, while the most prominent is the keyframe-based CERES-backend. In the frontend, features are only tracked for selected keyframes, which reduced computation time significantly. Once extracted, a direct method is used to track features. The system runs on two parallel threads, one thread for motion estimation and one thread for mapping.

6.2.1 Motion Estimation

In this thread, the depth of some pixels is assumed known from the mapping thread. The motion estimation consists of three steps. Sparse image alignment, relaxation and refinement.

Sparse image alignment

The sparse image alignment estimates the camera motion, that is the relative pose between two camera frames. This is done by minimizing the photometric error residual of all pixels that observe the same three-dimensional point. The optimization problem is formulated as

$$\mathbf{T}^*_{b_t b_{t-1}} = \arg\min_{\mathbf{T}_{b_t b_{t-1}}} \sum_{\bar{\mathcal{R}}^c_{k-1}} \left\| I_t \left(\pi \left(\mathbf{T}_{cb} \mathbf{T}_{b_t b_{t-1}} \mathbf{T}_{bc} \pi^{-1}(\mathbf{u}, \lambda) \right) \right) - I_{t-1}(\mathbf{u}) \right\|^2 \tag{6.2}$$

$\mathbf{T}_{b_t b_{t-1}}$ is the relative pose between the two body frames and $\mathbf{T}_{bc}$ is the body to camera transformation. The projection function π maps a 3D-point in the camera frame $\mathbf{x}^c \in \mathbb{R}^3$ to the image domain Ω.

$$\pi : \mathbb{R}^3 \rightarrow \Omega$$

The back projection function maps the image point back to the real world, given the real world depth of that pixel z^c

$$\pi^{-1} : \Omega \times \mathbb{R} \rightarrow \mathbb{R}^3$$

The intensity image recorded at timestep t is denoted I_t

$$I_t : \Omega_t \subset \mathbb{R}^2 \rightarrow \mathbb{R}$$

The residual in Equation (6.2) represents two pixels observing the same point. $\mathcal{R}_{t-1} \subseteq \Omega_{t-1}$ is the set pixels in image I_{t-1}, Ω_{t-1} where the depth is known at time $t-1$, and the subset $\bar{\mathcal{R}}_{t-1} \subset \mathcal{R}_{t-1}$ is the set of pixels where the back-projected points are also visible in image I_t, that is the set Ω_t

$$\bar{\mathcal{R}}_{t-1} = \{\mathbf{u} | \mathbf{u} \in \mathcal{R}_{t-1} \wedge \pi\left(\mathbf{T}_{b_t b_{t-1}} \mathbf{T}_{bc} \pi^{-1}(\mathbf{u}, \lambda)\right) \in I_t\}$$

To make this sparse approach more robust, the photometric cost is summed over a small patch around the pixels as well, with assumed similar depth.

Relaxation and alignment of 2D-features

This first direct alignment process only serves as a coarse alignment between two subsequent frames as they arrive. The keypoints which have their depth estimated are all registered with the frame where they first appeared, $\mathcal{F}_r$. KLT optical flow is then applied to get sufficient coordinate matches between this frame and the subsequent frames where they appear.

Refinement

The last step is performing a bundle adjustment. In this step, preintegrated IMU errors are also introduced

$$\begin{aligned}\mathcal{X}^* = \arg\min_{\mathcal{X}} & \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{L}_k^C} \left\| \mathbf{u}_i^* - \pi(\mathbf{T}_{b_k w} \boldsymbol{\rho}_i) \right\|^2 \\ & + \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{L}_k^E} \left\| \mathbf{n}_i^\top \left(\mathbf{u}_i^* - \pi(\mathbf{T}_{b_k w}, \boldsymbol{\rho}_i) \right) \right\|^2 \\ & + \sum \left\| \mathbf{z}^{IMU} - h^{IMU}(\mathcal{X}) \right\|^2\end{aligned}$$

where $\mathcal{K}$ is the set of all keyframes, $\mathcal{L}_k^C$ is the set of all corner features and $\mathcal{L}_k^E$ is the set of all edge features in keyframe k.

6.2.2 Mapping thread

In the mapping thread, feature depth is estimated. Here motion is assumed known from the motion estimation thread. The mapping thread estimates depth from multiple observations utilizing a recursive Bayesian depth filter. A new keyframe is selected, and thus new depth filters are initialized at corners and along edges when the number of tracked features falls below some threshold. New depth features are initialized with large uncertainty and undergo a recursive Bayes-update every subsequent frame. Every depth filter is associated with a reference frame r. For all previous frames, as well as every subsequent frame, a search along the epipolar lines is performed, and the zero mean sum of squared differences is computed. From the pixel with maximum correlation, the depth of the measurement is triangulated. When the depth of point is properly converged, it is inserted into the map used by the relaxed feature alignment in the motion estimation thread.

6.2.3 Reasoning for choice of SVO

Just as with VINS-Fusion, SVO supports all the sensor combinations relevant to this thesis. It is also rated as a good alternative in regards to accuracy and robustness in [41] and [42]. However, what sets it apart from other frameworks, is its handling of depth estimation, with a separate mapping thread. As this is different to how VINS-Fusion handles depth, complementary problems in regards to mapping might be solved by VINS-Fusion and SVO, and thus a more informative result regarding sensor combination might be acquired.

Chapter 7

MilliAmpere2: Experimental Setup

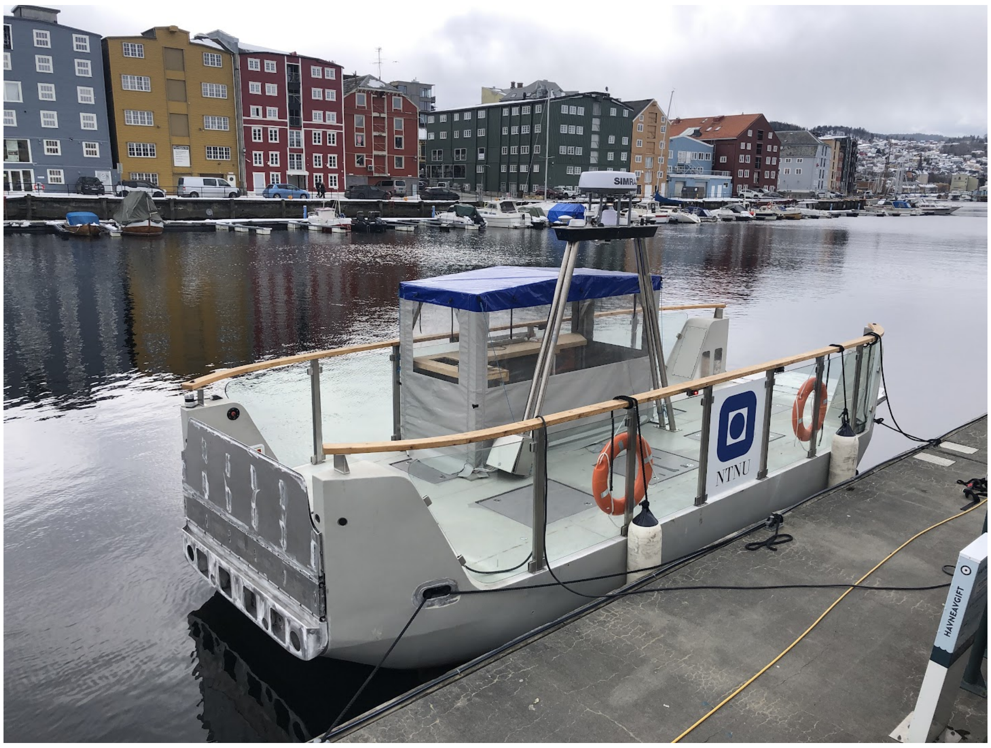

Figure 7.1: MilliAmpere2, docked at Trondheim Sentralstasjon

As mentioned in Chapter 1, MilliAmpere 2 is a prototype for a small and electric ferry made by NTNU [44]. It is equipped with a wide variety of sensors and is used to test autonomy software for ships. It is the successor of the first NTNU ferry prototype, milliAmpere, and aims to improve all of its shortcomings, such as size, thruster power and sensor placement. In this chapter, only the sensors used for VSLAM purposes will be presented. A picture of the ferry can be seen in Figure 7.1.

7.1 SentiPack

MilliAmpere2 uses a bundled system for IMU and GNSS called *SentiPack*. SentiPack consists of a *SentiBoard*, an IMU and a GNSS receiver. The SentiBoard is a microcontroller with a very high temporal resolution, which allows timestamping with an accuracy of 10 nanoseconds.

7.1.1 Inertial Measurement Unit

The IMU model is **ADIS 16490**. It has an output rate of 98.8hz and output integrated values. Since VIO libraries generally prefer raw IMU measurements, they must be converted. If the IMU has the rate r and receives an integrated measurement $\mathbf{x} \in \mathbb{R}^6$, a simple multiplication will give the raw measurement

$$\mathbf{x}_{raw} = \mathbf{x} \cdot r = \begin{bmatrix} a_x \cdot r \\ a_y \cdot r \\ a_z \cdot r \\ g_x \cdot r \\ g_y \cdot r \\ g_z \cdot r \end{bmatrix}$$

7.1.2 RTK-GNSS

The GNSS receiver used in SentiPack is a **Dual u-blox F9P**. This outputs latitude, longitude and altitude of the ship at a rate of 1hz. Using a base station located at Nyhavna, SentiPack gets RTK and thus has centimeter accuracy of its localization.

7.1.3 SentiFusion

In addition to measurements from the IMU and GNSS, SentiPack has an Inertial Navigation System (INS) running, using an EKF. This outputs pose of the ship and serves as ground truth in the experiments.

7.2 Cameras

To capture video, milliAmpere2 has 8 **FLIR Blackfly S GigE 50S5C-C** [45] cameras. The cameras are equipped with a 5mm lens [46] and are mounted in pairs in the corners of the ferry. They capture video at 10hz with a resolution of 1224x1024. The camera pairs sit in housings as shown in Figure 7.2. In this thesis, only two cameras are used. These are the two front-facing cameras located at the two front-facing corners, front-starboard and front-port. The corners are marked in Figure 7.3.

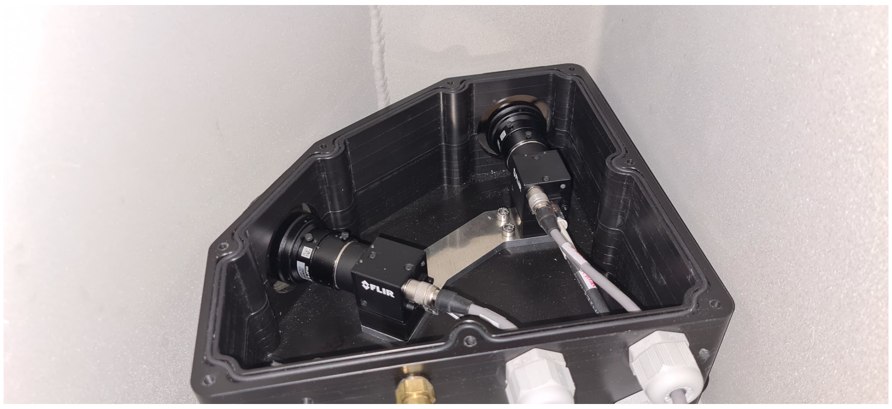

Figure 7.2: The camera housing which each camera pair is mounted in

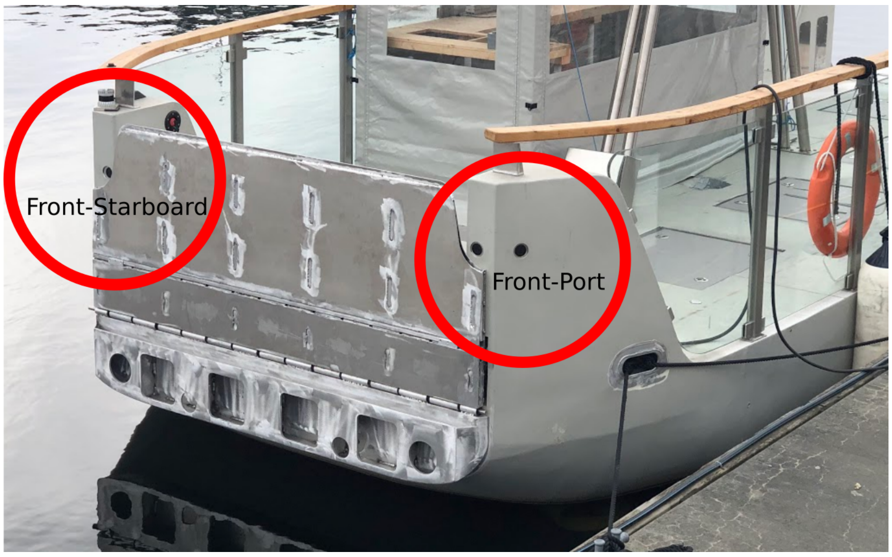

Figure 7.3: The position of the two cameras used in this thesis

7.2.1 Timestamping

The cameras are hardware synchronized and capture video at a rate of 10hz. They are triggered simultaneously to capture an image using the SentiBoard, and the image is paired with the timestamp of the trigger pulse in the camera driver.

Chapter 8

Results and Discussion

In this chapter, the local and global estimation results are presented and discussed.

8.1 Evaluation and Metrics

For evaluation, the metrics Absolute Positioning Error (APE) and Relative Positioning Error (RPE) will be used.

8.1.1 Absolute positioning error

Absolute Positioning Error (APE) measures the absolute error between the ground truth and the estimated path:

$$\mathbf{T}_{wGT_t} \ominus \mathbf{T}_{wb_t}$$

Where $\mathbf{T}_{wGT_t}$ is the pose of the ground truth. This metric quantifies how much an estimator drifts over its lifespan and can be used for e.g. assessing how much drift accumulates when going for a longer time without GNSS measurements.

8.1.2 Relative positioning error

As the odometry from the VIO estimators are fused with GNSS measurements later in this thesis, assessing drift over its entire lifespan is a somehow constructed scenario. The Relative Positioning Error (RPE) measures error between pose deltas and thus quantifies how well an estimator works locally to its measurements.

8.1.3 Pose, position and orientation

The APE or RPE can also be applied to just the orientation or position.

$$\mathbf{R}_{GT_{t-1}GT_t} \ominus \mathbf{R}_{b_{t-1}b_t}$$
$$\mathbf{p}_{GT_t}^{GT_{t-1}} - \mathbf{p}_{b_t}^{b_{t-1}}$$

8.2 Local Pose Estimation

In this section, the results regarding pure visual-inertial odometry using SVO and VINS-Fusion are presented and discussed. These results are produces using three different sensor combinations

- A monocular camaera and an IMU
- Two cameras in a stereo setup and an IMU
- Two cameras alone in a stereo setup

As there is only one monocular sensor combination, this might be referred to as a monocular sensor suite or just mono.

8.2.1 Short and Straight Path with Semi-Distant Features

In this dataset, the ferry starts just outside the dock of Ravnkloa and crosses over to the dock at Fosenkaia. This is a typical operational scenario and will tell what visual-inertial odometry can offer in the case of complete GNSS failure during a simple crossing. The trajectory in this scenario has very little rotation and acceleration involved. Considering the observability theory in Chapter 4, this should make the scale drift. The cameras are facing backwards considering the motion. The trajectory is drawn in Figure 8.1 and a sample of the view from the camera can be seen in Figure 8.2.

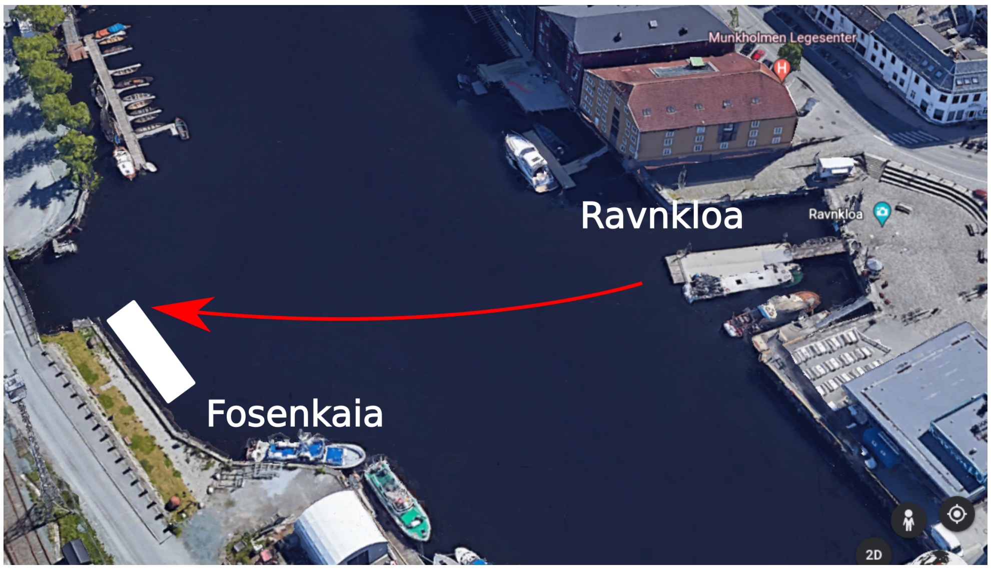

Figure 8.1: An illustration of the trajectory in the first dataset

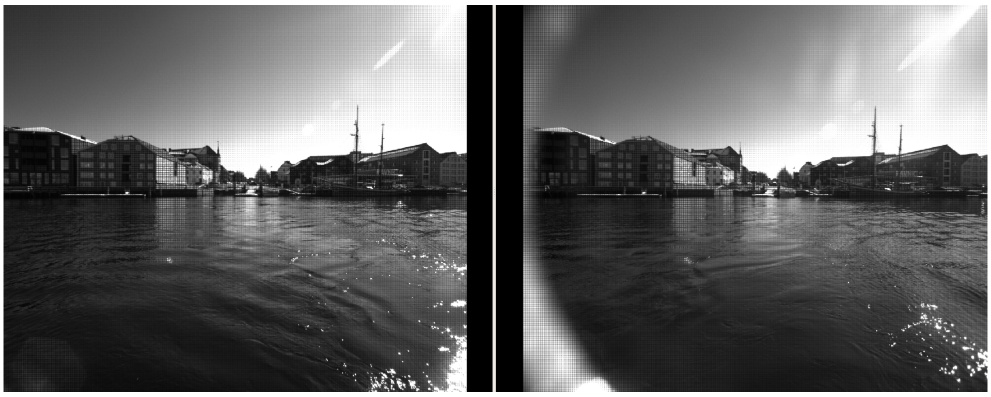

Figure 8.2: The stereo view from the ferry when crossing

Estimated trajectories

The estimated paths can be seen in Figure 8.3, Figure 8.4 and Figure 8.5. This is not informative other than "giving a feeling" for how good these estimates are. Just by inspection, it is possible to see that only the monocular sensor suites are close to the ground truth.

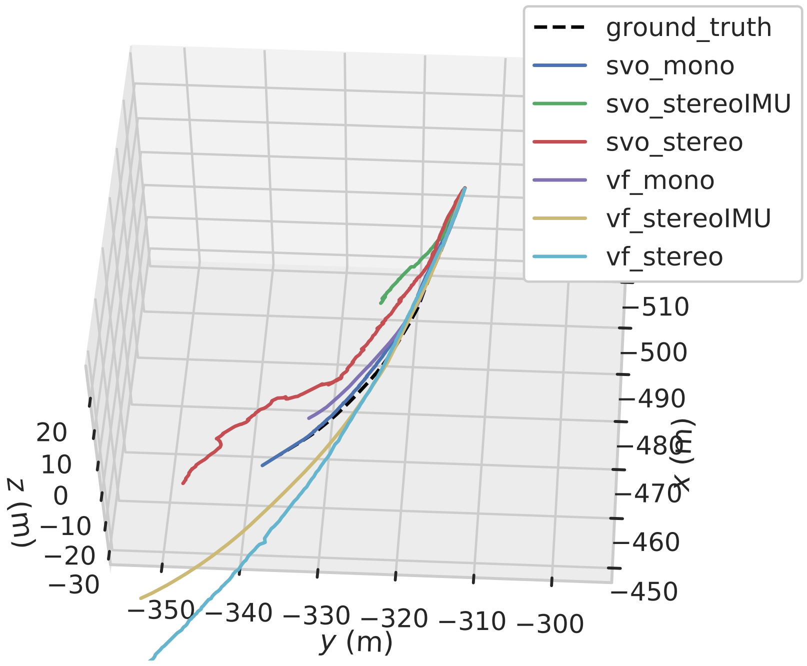

Figure 8.3: All method and sensor combinations

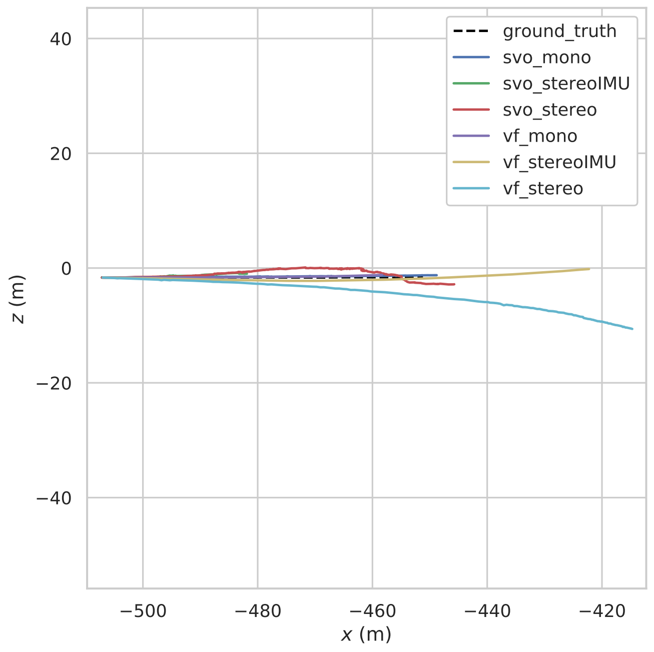

Figure 8.4: Trajectories in the xz-plane

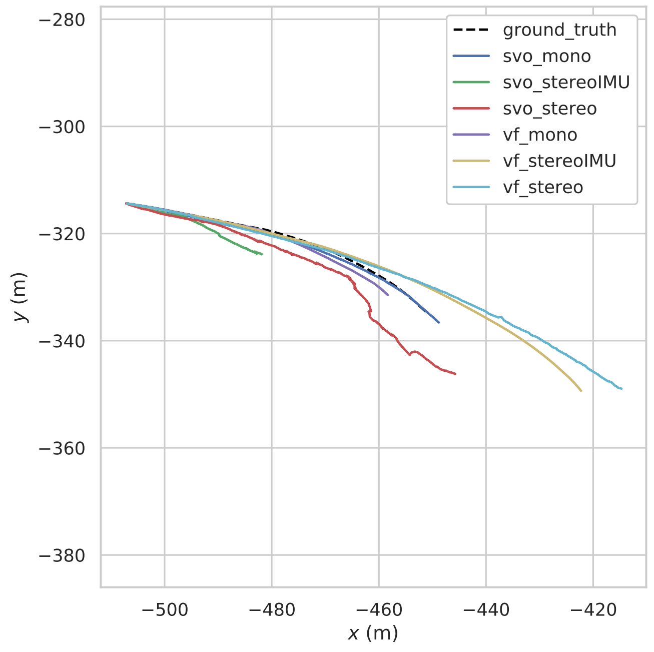

Figure 8.5: Trajectories in the xy-plane

Scale ambiguity

In Figure 8.6, the RPE for the estimators are quantified. For the trajectories estimated by VINS-Fusion, much of the faulty estimation can be credited due to drift in scale. This is shown in Figure 8.7, where the RPE is calculated again, but now with corrected scale. The monocular SVO-estimator seems to have less drift in scale than VINS-Fusion, and this can be credited to its depth-convergent nature, by just tracking properly converged landmarks. Once the scale problem is solved for VINS-Fusion, its monocular accuracy beats SVO for all sensor combinations. For SVO, the performance is not enhanced by correcting for scale. For the multi-camera estimators, this seems to be due to a divergent path.

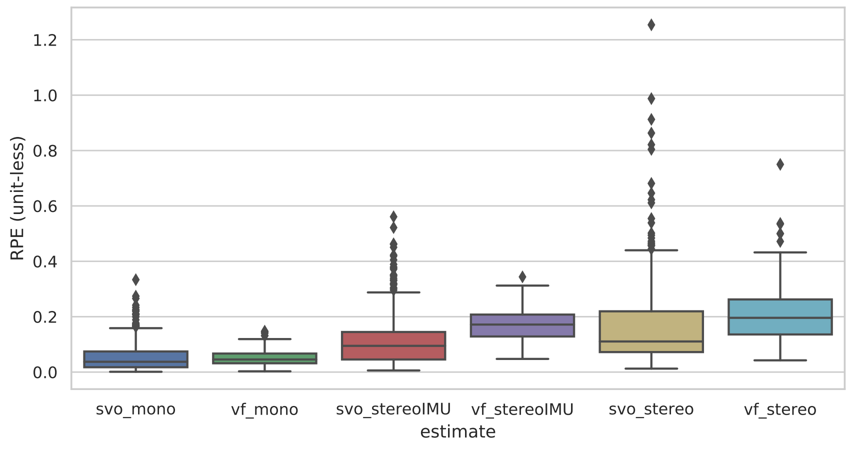

Figure 8.6: Boxplot for Relative Positioning Error (RPE)

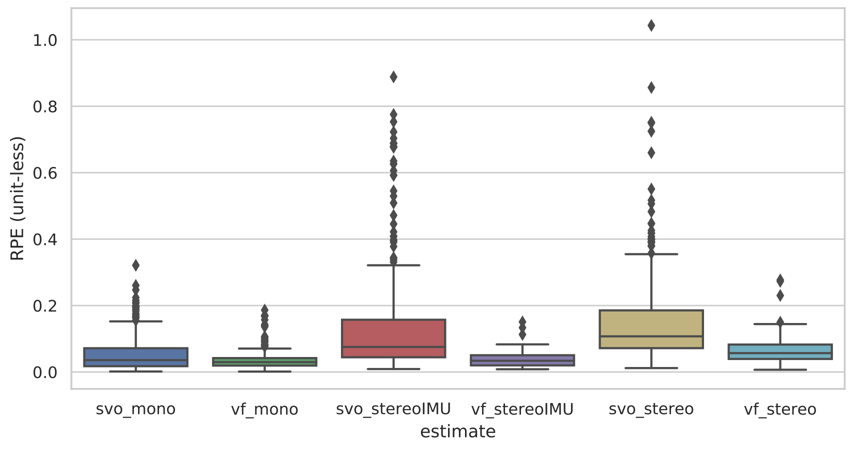

Figure 8.7: Boxplot for Relative Positioning Error (RPE) with corrected scale

Scale-corrected estimated trajectories

The estimated rotational and translational trajectories with corrected scale can be seen in Figure 8.9 and Figure 8.8. It is evident that what separates the accurate from the less accurate estimates is having a substantial amount of drift in the motion directions which is not excited when sailing. These directions include translational z, as well as pitch and roll.

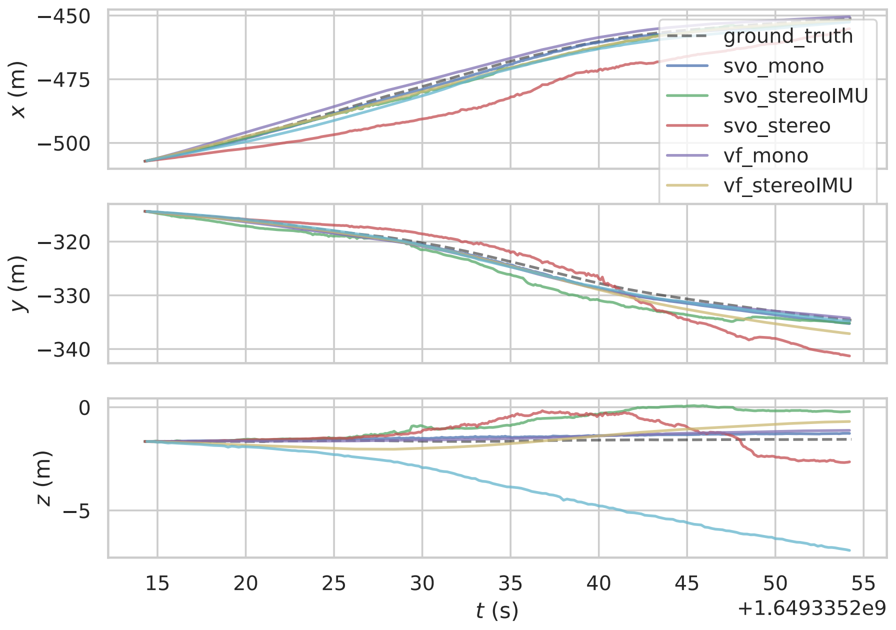

Figure 8.8: Estimated translational degrees of freedom

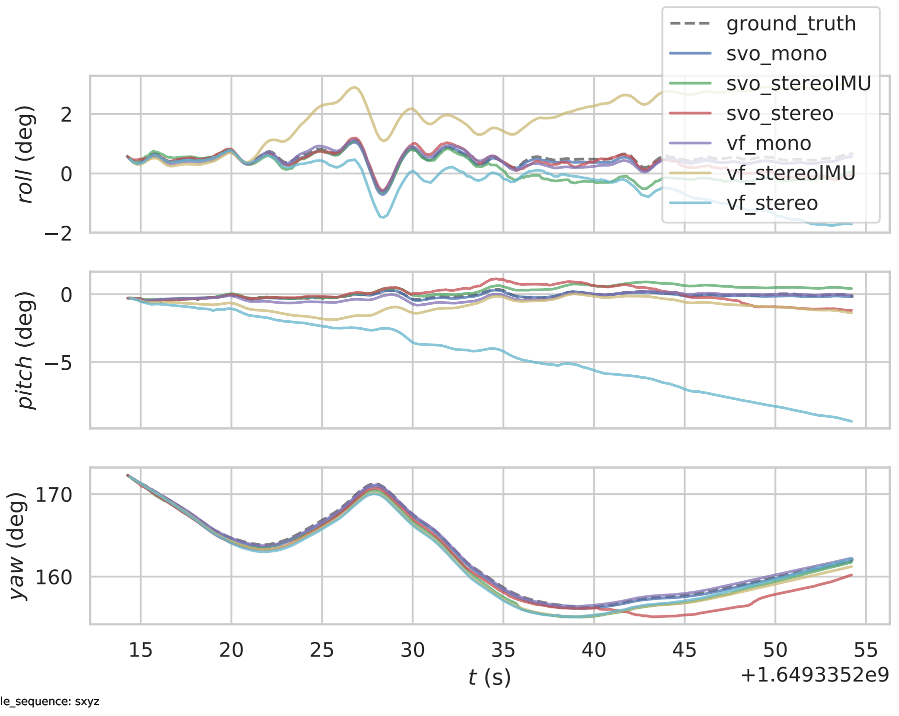

Figure 8.9: Estimated rotational degrees of freedom

8.2.2 Long and Straight Path with Very Distant Features

In this dataset, the ship sails from Trondheim Sentralbanestasjon to Fosenkaia. This is illustrated in Figure 8.10.

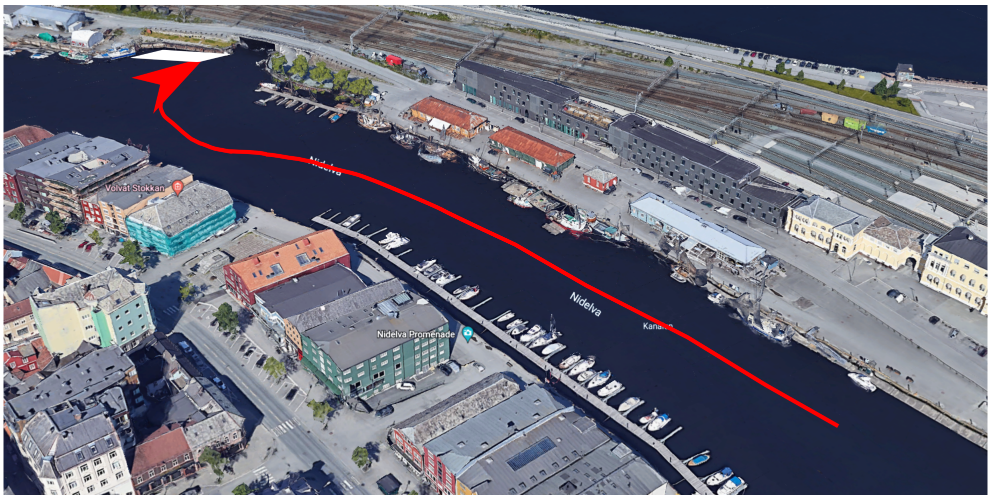

Figure 8.10: The trajectory that is sailed in the second dataset

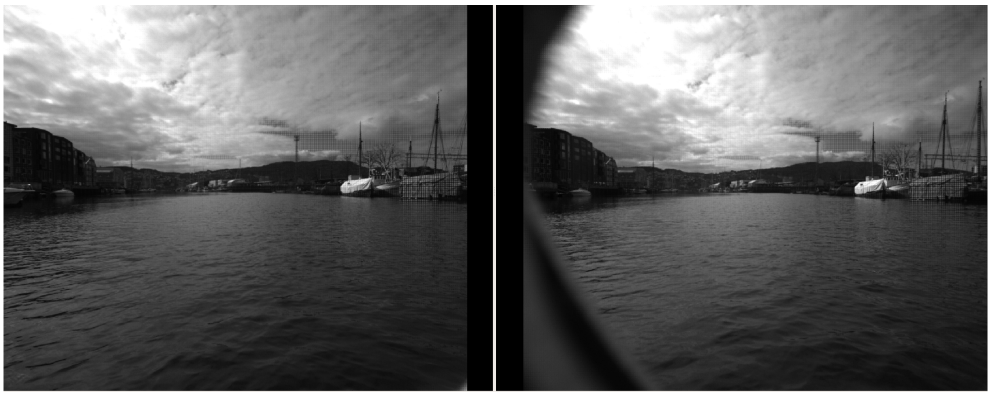

Figure 8.11: Stereo view from the ferry during the second dataset.

Estimated Trajectories

The estimated trajectories for the second dataset can be seen in Figure 8.12, Figure 8.13 and Figure 8.14. It is evident at once that scale is an issue to an even larger extent in this dataset.

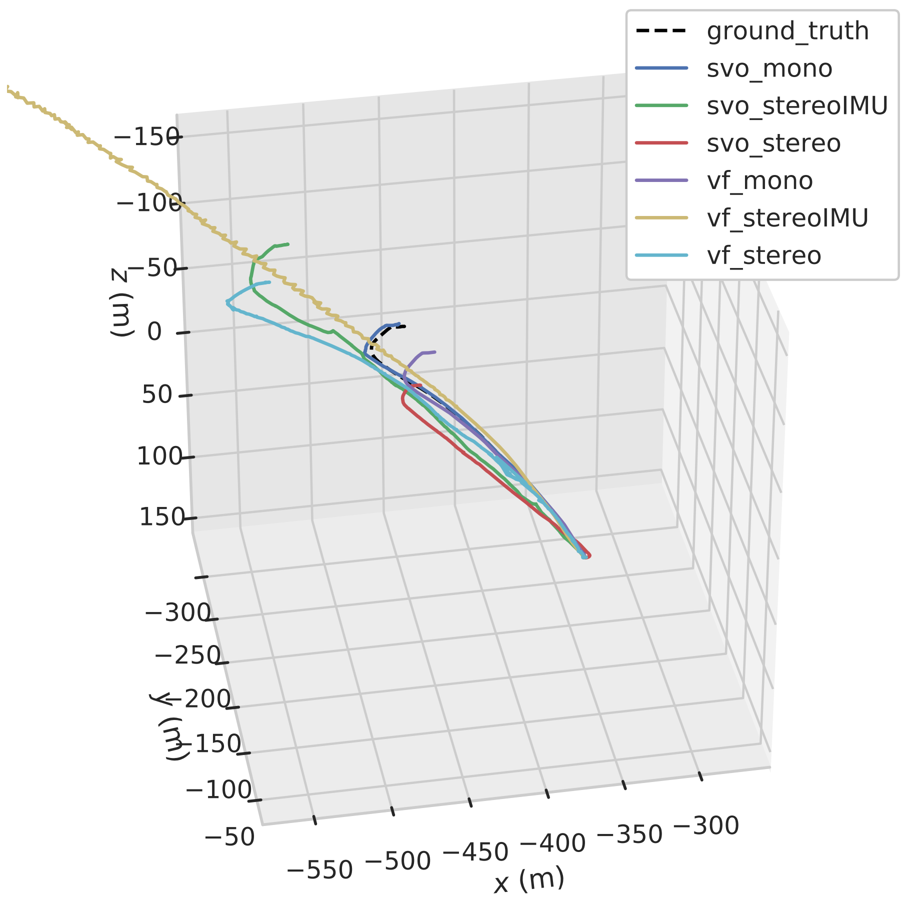

Figure 8.12: Estimated trajectories for the second dataset.

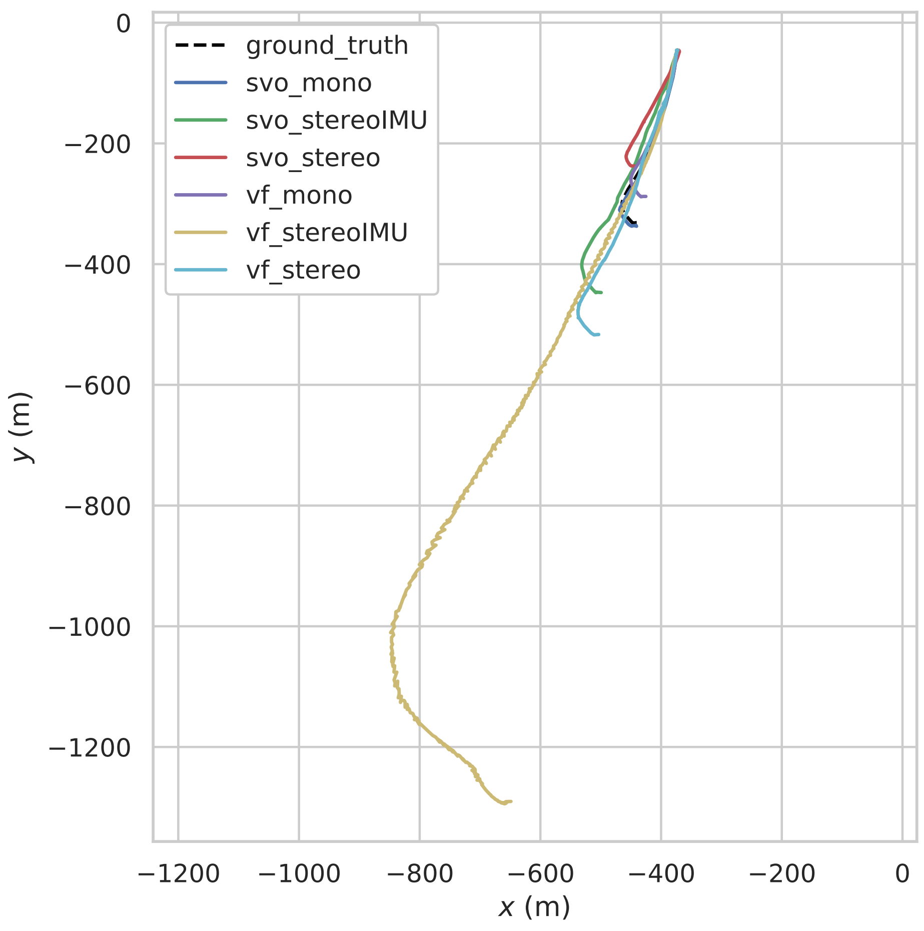

Figure 8.13: Trajectories in the xz-plane

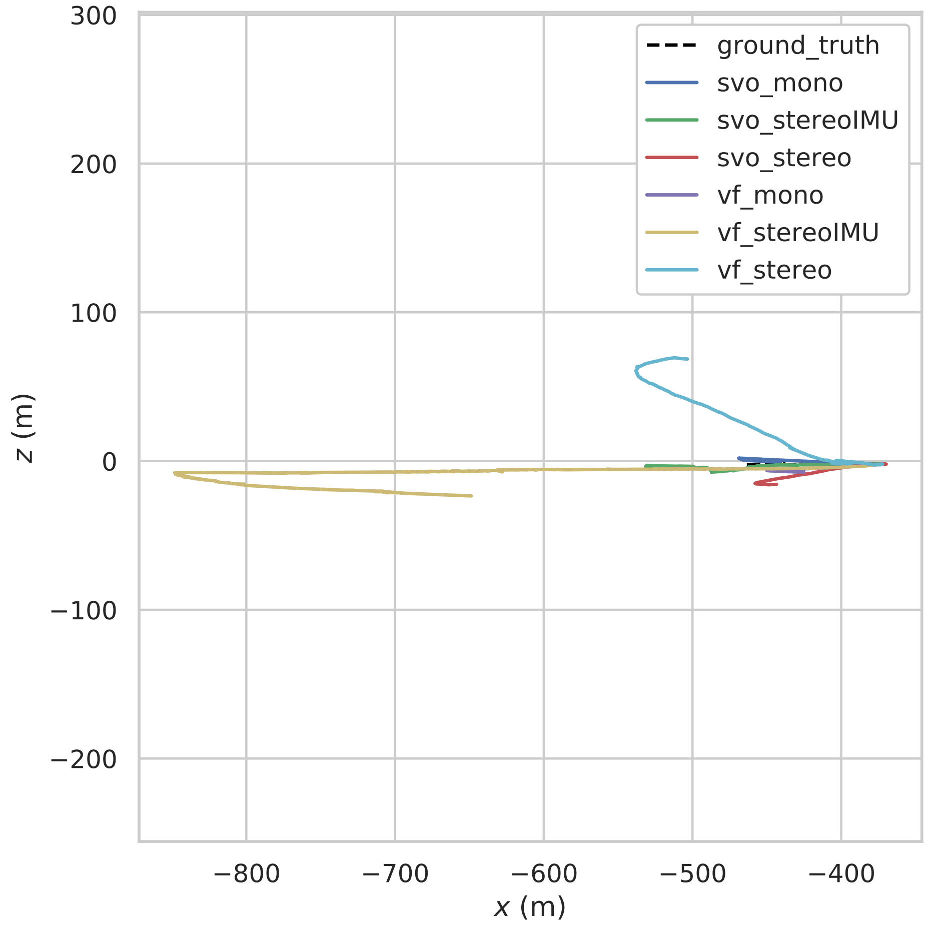

Figure 8.14: Trajectories in the xz-plane

Scale corrected trajectories

The trajectories in the xy-plane with corrected scale can be seen in Figure 8.15. A more detailed view can be seen in Figure 8.16 and Figure 8.17.

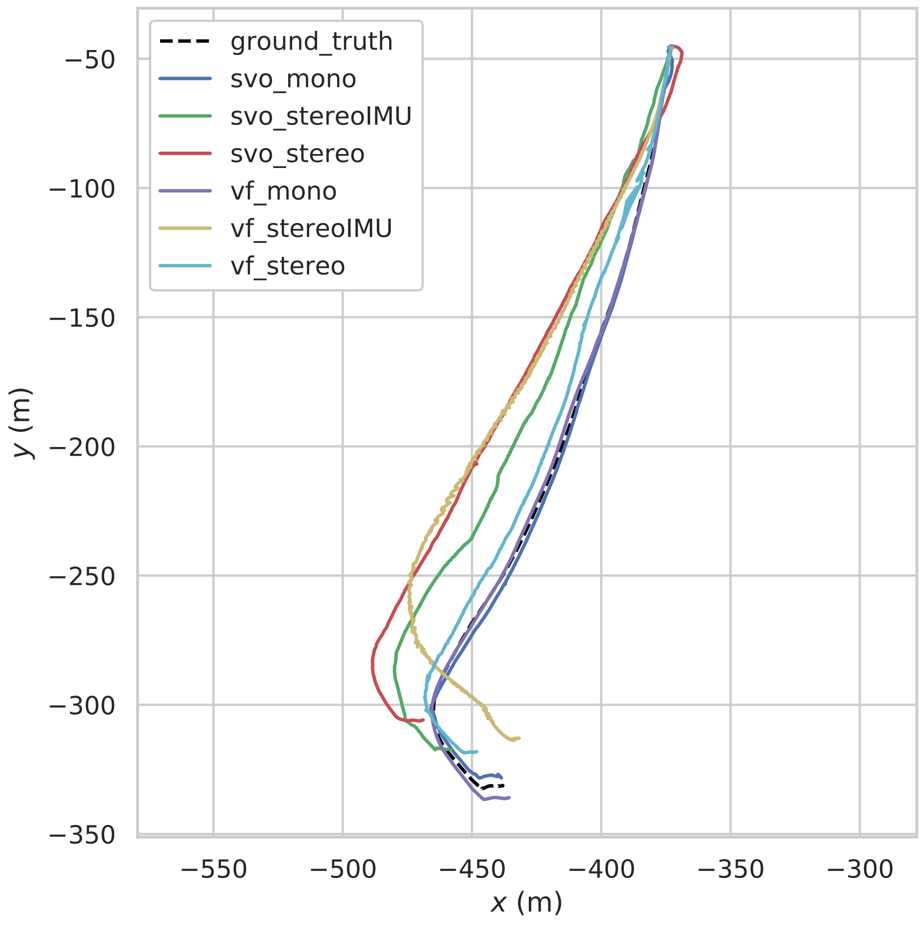

Figure 8.15: Scaled planar trajectory

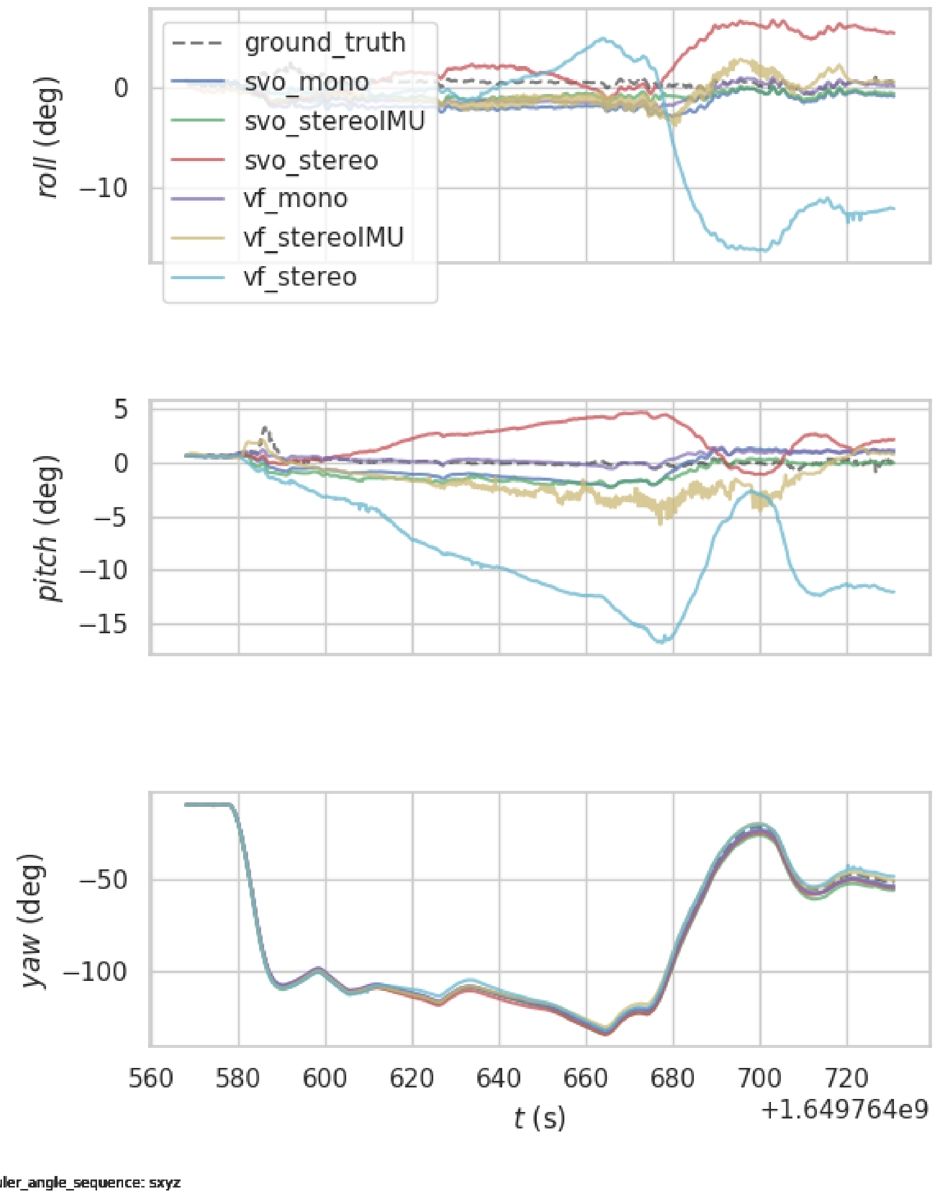

Figure 8.16: Estimated euler angles

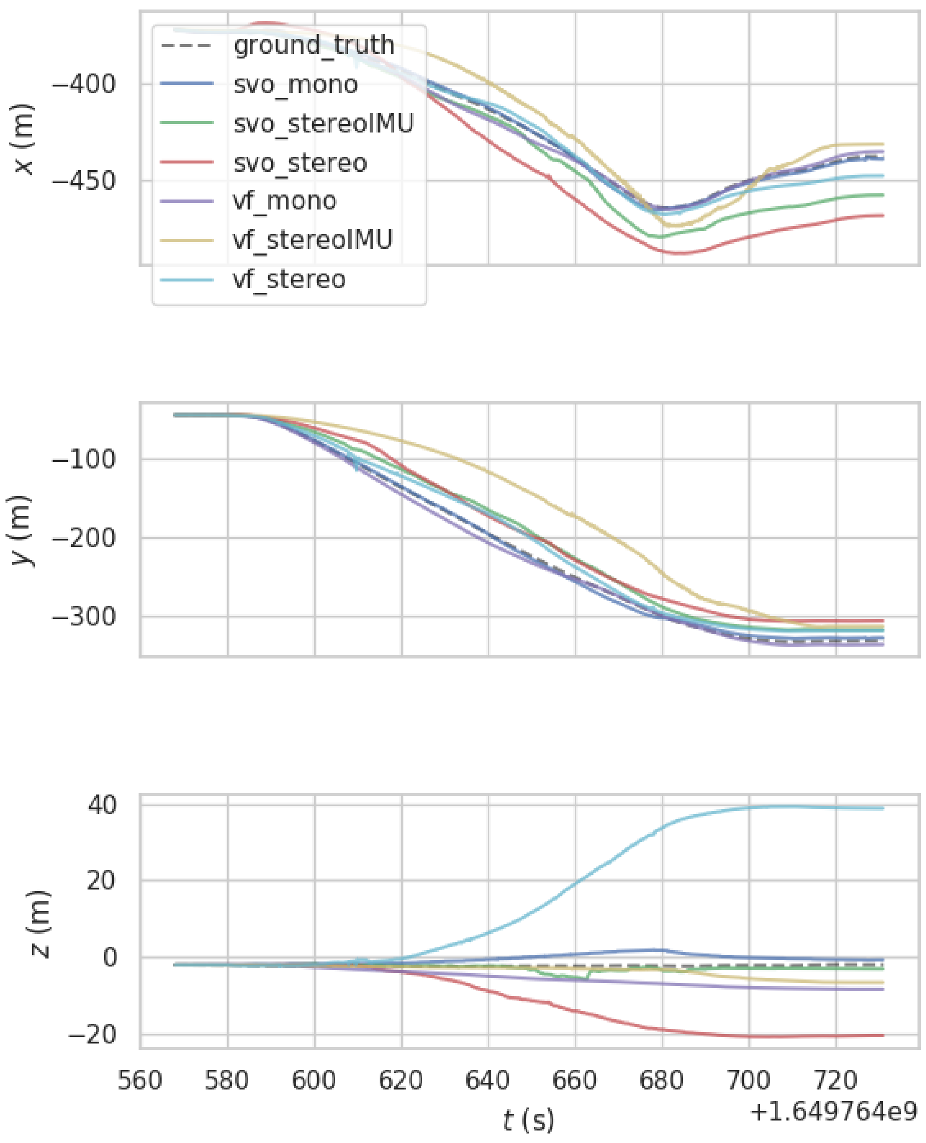

Figure 8.17: Estimated translation

Estimation performance

Once again it is also evident that there is a clear drift in the degrees of freedom which lacks excitation. Furthermore, during most of the trajectory, there are very few close and tangible features. Many features are initialized either in the sky or in the distant mountain tops. Additionally, features are also sometimes initialized in the river. This is shown in Figure 8.18. These features will not give any useful information for the estimator and rather introduce noise. Furthermore, many of the features which are "closer", such as the houses along the shore, still are very distant. A higher resolution could make these features easier to triangulate. In Figure 8.19, the RPE is visualized for the raw, non-corrected estimate. Even with all these sources of drift, errors and other problems, this plot suggests that the estimates are all pretty accurate locally, and thus can be used with some drift-correcting measures.

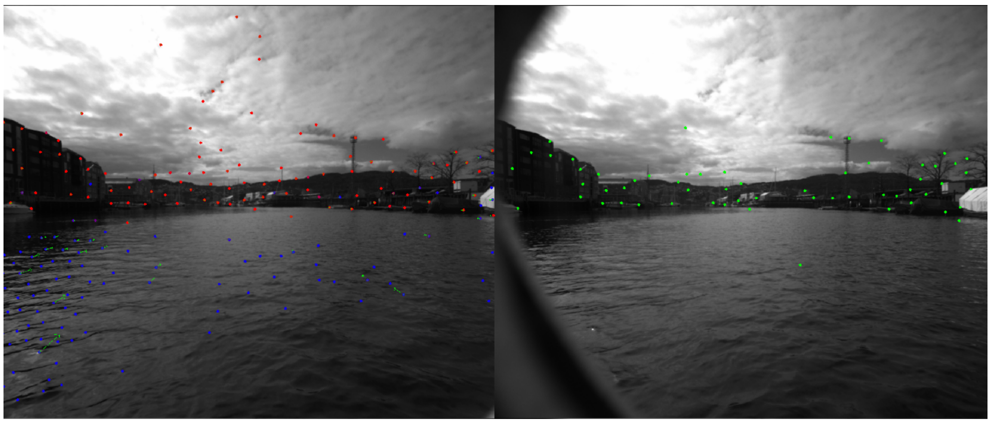

Figure 8.18: VINS-Fusion Stereo with features initialized in the sky, distant scenery and river

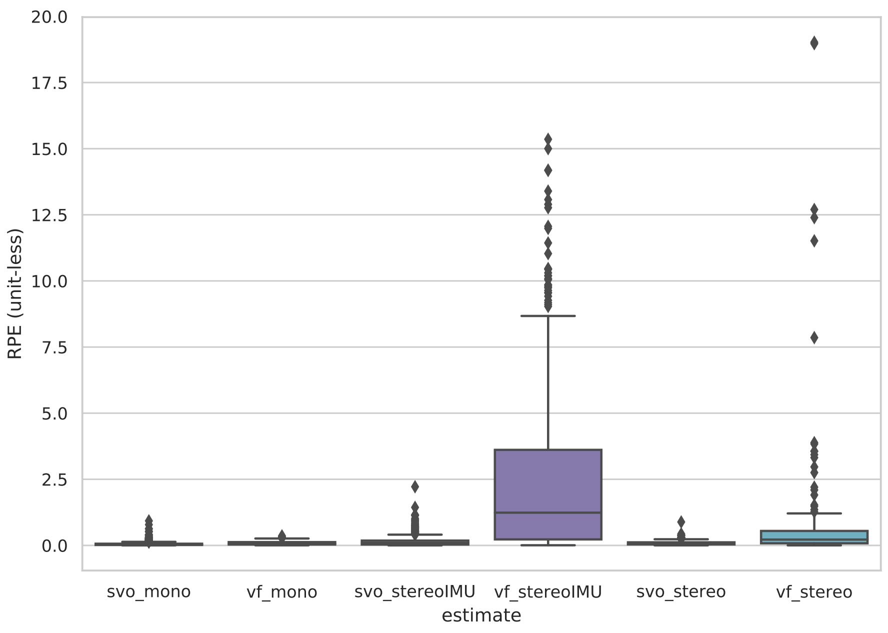

Figure 8.19: Box plots for the RPE in the second dataset

8.2.3 Summary of Performance

The RMSE values for the translational APE given in meters can be seen in Table 8.1. The RMSE for the APE related the orientation can be seen in Table 8.2 the RMSE for the RPE can be seen in Table 8.3.

	Mono		Stereo+IMU		Stereo	
	VINSFusion	SVO	VINSFusion	SVO	VINSFusion	SVO
Dataset1	4.44808	**2.16709**	20.4712	22.3034	24.9475	7.93014
Dataset2	31.0204	**7.53749**	575.604	83.9225	138.686	66.6406

Table 8.1: Root Mean Square Error (RMSE) values for the Absolute Positioning Error (APE) in the two datasets. This APE is related to position, given in meters

	Mono		Stereo+IMU		Stereo	
	VINSFusion	SVO	VINSFusion	SVO	VINSFusion	SVO
Dataset1	0.304383	**0.243381**	2.05569	1.02385	5.34307	1.47624
Dataset2	**2.20548**	2.70781	3.07706	3.42738	12.1584	4.66823

Table 8.2: Root Mean Square Error (RMSE) values for the Absolute Positioning Error (APE) in the two datasets. This APE is related to orientation, given in degrees

	Mono		Stereo+IMU		Stereo	
	VINSFusion	SVO	VINSFusion	SVO	VINSFuison	SVO
Dataset1	**0.0594244**	0.0761816	0.175409	0.152424	0.237337	0.232179
Dataset2	0.109572	**0.0850113**	3.57238	0.207382	1.35082	0.108765

Table 8.3: Root Mean Square Error (RMSE) values for the Relative Positioning Error (RPE) in the two datasets

8.2.4 Discussion on Local Visual-Inertial Odometry

The local visual-inertial odometry estimates are not great for either SVO and VINS-Fusion. The erroneous estimation manifests mostly as drift in scale, but also drift in other degrees of freedom. The result is that only the most robust of the methods, that is the monocular methods, have a performance which could be used in a real scenario for pure VIO. This can be happening due to many reasons.

Landmark initialization

Convergence of landmark position was a problem during the estimation. One reason for this could be that the estimators chose poor features. Poor features are features which are not anchored in rigid three-dimensional landmarks, but rather the sky or in the river. A way to discriminate the horizon and river from the image should help with this. Additionally, as most landmarks are far away from the ship, the resolution of the cameras must be higher. Increasing the resolution would make it easier to discriminate between camera angles, and make depth convergence more robust. Solving the problem with depth-convergence could yield great results, bringing the sensor suites based on stereo cameras to a competitive level. This is apparent from SVO outperforming VINS-Fusion in almost every aspect before scale-correction. Landmarks and features are an aspect of visual-inertial odometry estimation which is very different for a maritime setting compared to a in-door drone flight.

Motion-constrained optimization

Additionally, there is an observed drift in the degrees of freedom which are not excited. A penalizing term for this type of estimated motion could be added. This type of erroneous estimation could originate from the issues related to observability discussed earlier in this thesis. By incorporating the prior knowledge of planar motion with little to no pitch and roll, discrimination between IMU bias and motion-induced measurements could be easier when performing the non-accelerated motion, such as constant speed and total stop.

8.3 Global Pose Estimation

In the previous section, it was found that using VIO-methods in maritime conditions with big open areas function is sub-optimal. However, the reason for including VIO in the navigation system is not to use it in areas where the GNSS-reception is good and works well, but to aid it in conditions where it otherwise struggles, such as under bridges and in tunnels. VIO systems generally work well in areas rich with close features, and as will be evident in the following section, a tunnel under a bridge is an example of exactly this.

8.3.1 Path through a tunnel

In this dataset, the ship sails as illustrated in Figure 8.20. It starts outside the canal and sails under the railway through a tunnel. It ends by docking at Ravnkloa.

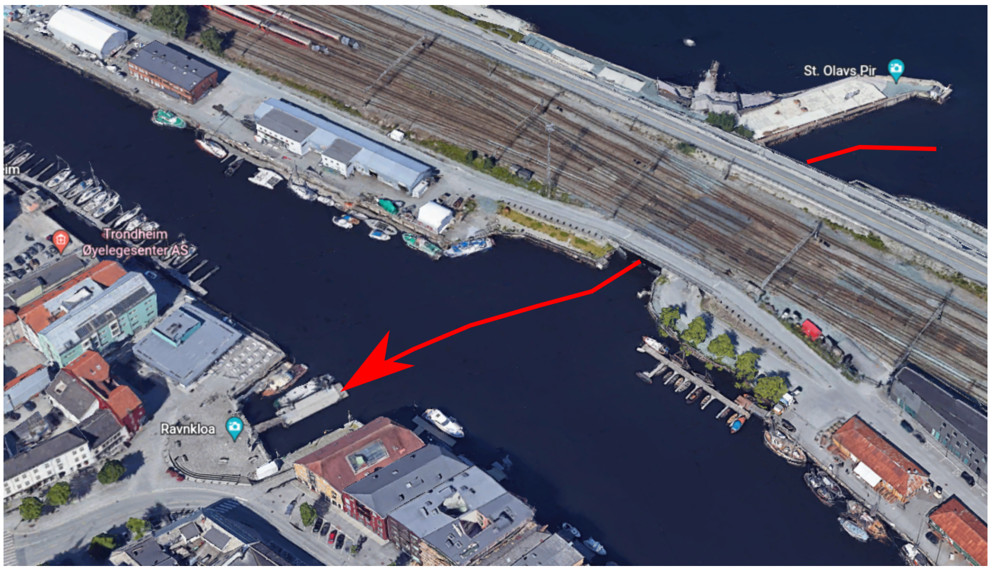

Figure 8.20: The path of the ship

Estimated trajectories

For this dataset, VINSFusion with a monocular sensor-suite is used to estimate the Visual-Inertial Odometry. This was used due to its theoretical edge in performance over SVO in optimal conditions, in addition to its natural compatibility with VINSFusion-GNSS. The ground truth is, as mentioned previously, the path estimated by the INS bundled with SentiPack. Due to the missing reception, this estimation process fails. Therefore there is no ground truth for this scenario. The resulting three-dimensional path from the VIO estimator, the GNSS-VIO fused estimator and the ground truth is shown in Figure 8.21 and Figure 8.22.

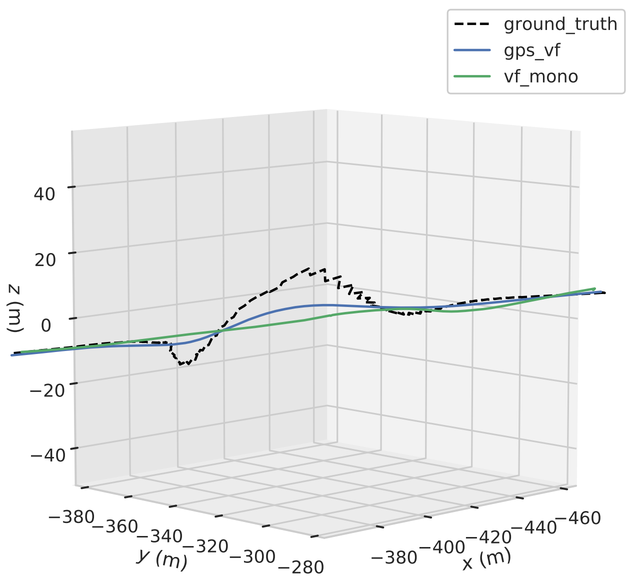

Figure 8.21: The estimated trajectories when sailing under the bridge.

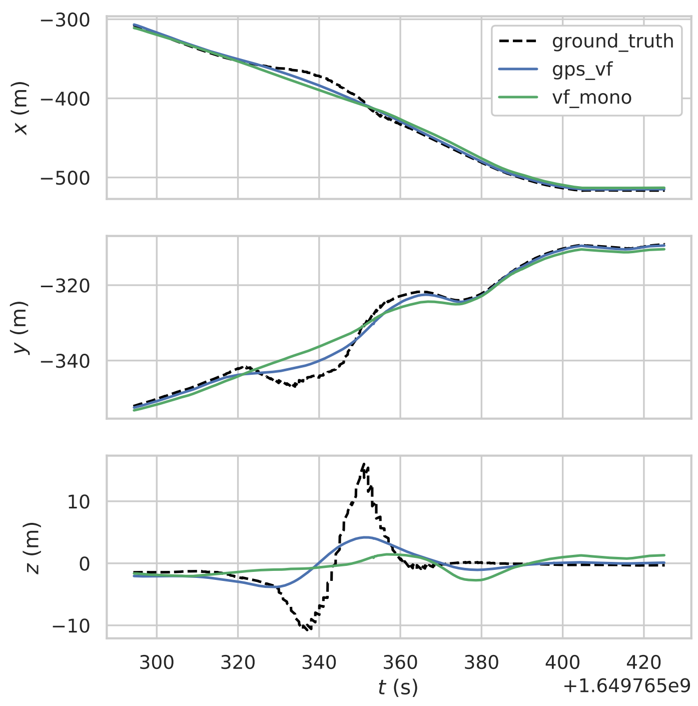

Figure 8.22: XYZ view of the estimated trajectories.

8.3.2 Discussion

The INS struggles to estimate the path when sailing through the tunnel. The VIO estimator, however, thrives in these conditions. A picture of the estimated path and map can be seen in Figure 8.23

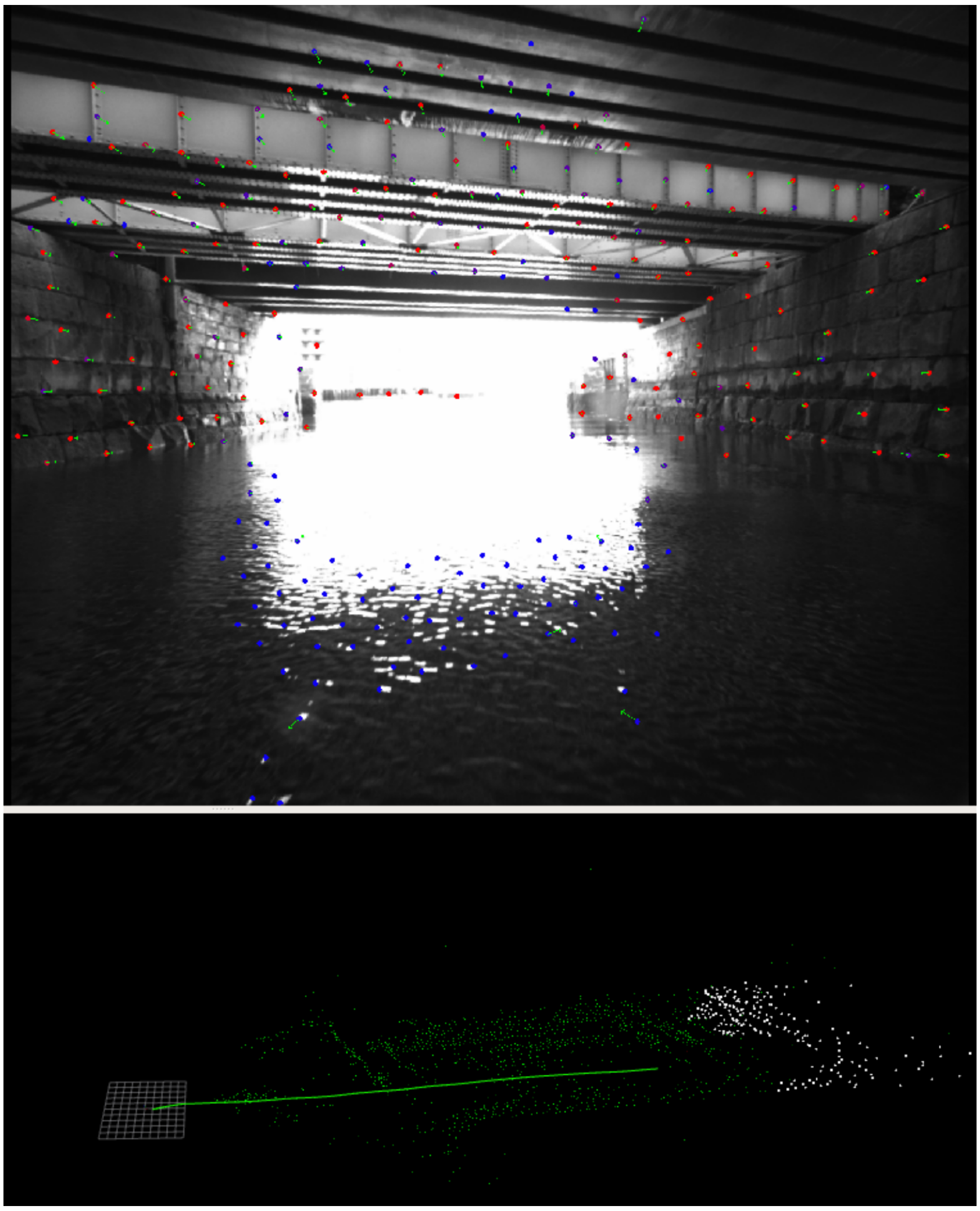

Figure 8.23: The view of the monocular camera, as well as the map it is estimating when sailing through the tunnel.

Since there are a significant amount of close features, VINSFusion has an easy time estimating the VIO accurately. The resulting GNSS-VIO-fused estimate receives some of the oscillation caused by the loss of GNSS reception, but the VIO estimate does a good job of dampening this. In an optimal setup, the covariances of the GNSS measurements should be inflated more, such that only the Visual-Inertial Odometry is trusted during the time in the tunnel. However, the point of the dataset still stands: using a camera as a motion sensor in addition to an IMU and the GNSS has value also at sea, even though using VIO as a sole method for localization has subpar results.

Chapter 9

Conclusion

In this thesis, SVO and VINSFusion, two state-of-the-art methods for estimation of Visual-Inertial Odometry have been tested at sea. This was done using three sensor combinations: 1) A monocular camera and an IMU, 2) a stereo camera setup and an IMU and 3) a stereo camera setup alone. For this purpose, only the monocular sensor setup showed sound performance for localizing the ship. Drift in scale showed to be a big problem, especially for the stereoscopic sensor combinations. A proposed reason for this was that as most landmarks observed were either semi-distant or very distant from the ship, the depth of the corresponding features was not converging properly to their real position. A proposed solution was incorporating a more robust check on depth convergence for points before using them for bundle adjustment. Also, using a higher resolution for the video cameras as well as filtering out the sky and the sea from the camera image was deemed as potentially performance-improving measures. Additionally, the monocular sensor setup was tested together with a method for fusing GNSS measurements and Visual-Inertial Odometry. This was deemed a success, as localization based on GNSS and Visual-Inertial Odometry hold complementary properties. Localization when sailing through a tunnel where there was a corrupted GNSS reception was greatly improved by incorporating measurements from a camera. For this reason, it was deemed that despite the bad isolated performance, using a camera setup as a part of the localization sensor suite holds great value in urban, and maritime conditions.

Bibliography

[1] T. Zhang, Q. Li, C.-s. Zhang, H.-w. Liang, P. Li, T.-m. Wang, S. Li, Y.-l. Zhu and C. Wu, 'Current trends in the development of intelligent unmanned autonomous systems,' *Frontiers of information technology & electronic engineering*, vol. 18, no. 1, pp. 68–85, 2017.

[2] B. Dynamics. 'Spot for industrial inspections.' (2022), [Online]. Available: `https://www.bostondynamics.com/solutions/inspection` (visited on 16/05/2022).

[3] P. Dutta, A. Baruah, A. Konwar *et al.*, 'A technical review of lawn mower technology,' *ADBU Journal of Engineering Technology*, vol. 4, 2016.

[4] A. Autonomy. 'Applied autonomy.' (2022), [Online]. Available: `https://www.appliedautonomy.no/` (visited on 16/05/2022).

[5] Tesla. 'Tesla autopilot.' (2022), [Online]. Available: `https://www.tesla.com/autopilot` (visited on 16/05/2022).

[6] Zeabuz. 'Zeabuz: Maritime autonomy for mobility change-makers.' (2022), [Online]. Available: `https://www.zeabuz.com/` (visited on 16/05/2022).

[7] D. of Marine Technology. 'Department of marine technology, faculty of engineering.' (2022), [Online]. Available: `https://www.ntnu.edu/itk` (visited on 24/05/2022).

[8] D. of Engineering Cybernetics. 'Department of engineering cybernetics, faculty of information technology and electrical engineering.' (2022), [Online]. Available: `https://www.ntnu.edu/itk` (visited on 24/05/2022).

[9] NTNU. 'Norwegian university of science and technology.' (2022), [Online]. Available: `https://www.ntnu.no/` (visited on 24/05/2022).

[10] E. Brekke, 'Fundamentals of sensor fusion.'

[11] B. Hofmann-Wellenhof, H. Lichtenegger and E. Wasle, *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more*. Springer Science & Business Media, 2007.

[12] F. Van Diggelen and P. Enge, 'The world's first gps mooc and worldwide laboratory using smartphones,' in *Proceedings of the 28th international technical meeting of the satellite division of the institute of navigation (ION GNSS+ 2015)*, 2015, pp. 361–369.

[13] T. Takasu and A. Yasuda, 'Development of the low-cost rtk-gps receiver with an open source program package rtklib,' in *International symposium on GPS/GNSS*, International Convention Center Jeju Korea, vol. 1, 2009.

[14] M. O. Aqel, M. H. Marhaban, M. I. Saripan and N. B. Ismail, 'Review of visual odometry: Types, approaches, challenges, and applications,' *SpringerPlus*, vol. 5, no. 1, pp. 1–26, 2016.

[15] 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, no. 6, pp. 1309–1332, 2016.

[16] T. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 2021, ISBN: 9781119575054. [Online]. Available: `https://books.google.no/books?id=srJEvgEACAAJ`.

[17] S. Wright, J. Nocedal *et al.*, 'Numerical optimization,' *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[18] T. V. Haavardsholm, *A handbook in visual slam*, `https://github.com/tussedrotten/vslam-handbook`, 2021.

[19] O. Egeland and J. T. Gravdahl, *Modeling and simulation for automatic control*. Marine Cybernetics Trondheim, Norway, 2002, vol. 76.

[20] J. Sola, J. Deray and D. Atchuthan, 'A micro lie theory for state estimation in robotics,' *arXiv preprint arXiv:1812.01537*, 2018.

[21] J. Shi *et al.*, 'Good features to track,' in *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, IEEE, 1994, pp. 593–600.

[22] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige and R. Siegwart, 'Keyframe-based visual-inertial slam using nonlinear optimization,' *Proceedings of Robotis Science and Systems (RSS) 2013*, 2013.

[23] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart and P. Furgale, 'Keyframe-based visual–inertial odometry using nonlinear optimization,' *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[24] C. Harris, M. Stephens *et al.*, 'A combined corner and edge detector,' in *Alvey vision conference*, Citeseer, vol. 15, 1988, pp. 10–5244.

[25] S. Leutenegger, M. Chli and R. Y. Siegwart, 'Brisk: Binary robust invariant scalable keypoints,' in *2011 International conference on computer vision*, Ieee, 2011, pp. 2548–2555.

[26] S. Agarwal, K. Mierle and T. C. S. Team, *Ceres Solver*, version 2.1, Mar. 2022. [Online]. Available: `https://github.com/ceres-solver/ceres-solver`.

[27] J. Engel, V. Koltun and D. Cremers, 'Direct sparse odometry,' *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[28] J. Engel, V. Koltun and D. Cremers, 'Direct sparse odometry,' *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 40, no. 03, pp. 611–625, 2018.

[29] M. Bloesch, S. Omari, M. Hutter and R. Siegwart, 'Robust visual inertial odometry using a direct ekf-based approach,' pp. 298–304, 2015. DOI: `10.1109/IROS.2015.7353389`.

[30] M. Bloesch, M. Burri, S. Omari, M. Hutter and R. Siegwart, 'Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback,' *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[31] C. Forster, M. Pizzoli and D. Scaramuzza, 'SVO: Fast semi-direct monocular visual odometry,' in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 15–22. DOI: `10.1109/ICRA.2014.6906584`.

[32] C. Forster, Z. Zhang, M. Gassner, M. Werlberger and D. Scaramuzza, 'SVO: Semidirect visual odometry for monocular and multicamera systems,' *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, 2017. DOI: `10.1109/TRO.2016.2623335`.

[33] T. Qin, P. Li and S. Shen, 'Vins-mono: A robust and versatile monocular visual-inertial state estimator,' *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[34] A. Martinelli, 'Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination,' *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 44–60, 2011.

[35] K. J. Wu, C. X. Guo, G. Georgiou and S. I. Roumeliotis, 'Vins on wheels,' in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 5155–5162.

[36] S. Cao, X. Lu and S. Shen, 'Gvins: Tightly coupled gnss–visual–inertial fusion for smooth and consistent state estimation,' *IEEE Transactions on Robotics*, 2022.

[37] D. S. Giovanni Cioffi, 'Tightly-coupled fusion of global positional measurements in optimization-based visual-inertial odometry,' in *IEEE Int. Conf. Intel. Robots and Syst. (IROS)*, 2020.

[38] T. Qin, S. Cao, J. Pan and S. Shen, 'A general optimization-based framework for global pose estimation with multiple sensors,' *arXiv preprint arXiv:1901.03642*, 2019.

[39] B. D. Lucas, T. Kanade *et al.*, 'An iterative image registration technique with an application to stereo vision,' Vancouver, British Columbia, 1981.

[40] G. Page, 'Multiple view geometry in computer vision, by richard hartley and andrew zisserman, cup, cambridge, uk, 2003, vi+ 560 pp., isbn 0-521-54051-8.(paperback£ 44.95),' *Robotica*, vol. 23, no. 2, pp. 271–271, 2005.

[41] C. Campos, R. Elvira, J. J. G. Rodrıguez, J. M. Montiel and J. D. Tardós, 'Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam,' *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[42] J. Delmerico and D. Scaramuzza, 'A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots,' in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 2502–2509.

[43] C. Forster, L. Carlone, F. Dellaert and D. Scaramuzza, 'On-manifold preintegration for real-time visual–inertial odometry,' *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[44] Autoferry. 'Autoferry: Autonomous all-electric passenger ferries for urban water transport (autoferry).' (2022), [Online]. Available: `https://www.ntnu.edu/autoferry/about` (visited on 19/05/2022).

[45] FLIR. 'Flir blackfly s gigabit ethernet 50s5c-c product page.' (2022), [Online]. Available: `https://www.flir.eu/products/blackfly-s-gige/?model=BFS-PGE-50S5C-C` (visited on 19/05/2022).

[46] Kowa. 'Lm5jcm | 2/3" 5mm 2mp c-mount lens.' (2022), [Online]. Available: `https://www.kowa-lenses.com/en/lm5jcm-2mp-industrial-lens-c-mount` (visited on 19/05/2022).

Aleksander Nysted Elvebakk

Fusion of Visual-Inertial Odometry and GNSS-Based Positioning for Localization of Autonomous Ferries

NTNU
Norwegian University of
Science and Technology