Master's thesis

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

Kristoffer Steinsland

Sensor fusion of phased array radio bearing and inertial measurements using factor-graph-based optimization

Master's thesis in Cybernetics and Robotics
Supervisor: Torleiv Håland Bryne
July 2023

NTNU
Norwegian University of
Science and Technology

Kristoffer Steinsland

Sensor fusion of phased array radio bearing and inertial measurements using factor-graph-based optimization

Master's thesis in Cybernetics and Robotics
Supervisor: Torleiv Håland Bryne
July 2023

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

NTNU
Norwegian University of
Science and Technology

Preface

This master's thesis was written during the spring of 2023 as delivery for the 30 credit course "TTK4900 - Engineering Cybernetics, Master's Thesis" and is the final derivable of the Master of Science (MSc) degree in the Robotics and Cybernetics at the Norwegian University of Science and Technology (NTNU). The topic of concern is phased array radio aided inertial navigation using factor graph based optimization. Relevant concepts and background theory regarding this topic is explained. Hence, the prerequisites are limited to basic statistics and linear algebra. The author would like to thank the supervisor of this project associate professor Torleiv Håland Bryne who has given up a considerable amount of his time to provide guidance, feedback, and words of encouragement during the writing of this thesis.

The following is a list of the resources, tools, and help that were available for this project or used as a basis for the work:

- Data from field experiments were provided by the NTNU UAV Lab.
- The supervisor provided a set of `MATLAB` tools for processing data from field experiments.
- Course material from TTK4250 Sensor fusion inspired the development of the error state Kalman filter used for benchmarking of the main contributions of this work. A flight path from this course has also been used in simulations.
- Requlars supervision has been provided by the supervisor.
- A workstation computer has been provided by the Department of Engineering Cybernetics.

Abstract

The global navigation satellite system (GNSS) is the primary sensor used in modern navigation. However, some scenarios make GNSS unreliable or outright unavailable. Phased array radio systems (PARS) give way to a different approach that remedies some of the weaknesses present with GNSS. Phased array ground stations are able to acquire bearing measurements that can be exploited to create a navigation system independent of GNSS. However, PARS measurements are susceptible to errors like multipath, which requires the measurements to be fused with inertial measurements from an inertial measurement unit (IMU). In recent years factor graph optimization (FGO) has become popular in computer vision and simultaneous localization and mapping (SLAM) research. In this work, a FGO-based estimator is developed for the fusion of PARS and IMU measurements and benchmarked with an error state Kalman filter (ESKF). Both estimators were tested in simulation and on data gathered from field experiments. To deal with the multipath issue present in the PARS measurements, two different approaches for outlier rejection are tested for the Kalman filter and FGO estimators, respectively. The results indicate that FGO gives more accurate estimates in the presence of outliers due to robust outlier rejection techniques. Additionally, a solution to the problem of calibrating the PARS antenna mounting position and orientation online using FGO is proposed and tested.

Sammendrag

Globalt navigasjonssatellittsystem (GNSS) er den primære sensoren som brukes i moderne navigasjon. Imidlertid gjør noen scenarier GNSS upålitelig eller helt utilgjengelig. Fase-arrangerte radiosystemer (PARS) muliggjør en annen tilnærming til navigasjon som begrenser noen av svakhetene man finner i GNSS. Fase-arrangerte bakkestasjoner kan samle retningsmålinger som kan bli benyttet til å lage et navigasjonssystem uavhengig av GNSS. En utfordring med PARS-målinger er at de er utsatt for feil som flersti-problemet. Dette krever fusjonering av PARS-målinger med treghetsmålinger fra en treghetsmålingsenhet (IMU). De siste årene har faktorgrafoptimering (FGO) blitt en populær tilnærming innenfor forskning i feltene datasyn og simultan lokalisering og kartlegging (SLAM). I dette arbeidet utvikles en faktorgrafestimator for fusjonering av IMU og PARS målinger og sammenlignes mot et feiltilstands-Kalmanfilter. Begge estimatorene ble testet på både simuleringer og data samlet gjennom eksperimenter i felt. To forskjellige tilnærminger med avvisning av avvikende målinger er testet for Kalmanfilter og FGO estimatorene for å håndtere flersti-problemet til stede i PARS-målingene. Resultatene indikerer at FGO gir mer nøyaktige resultater når avvikende målinger er til stede på grunn av robuste avvisningsteknikk. Til slutt er en løsning til problemet med å kalibrere for posisjon og orientering til PARS-antennene etter montering i sanntid ved bruk av FGO foreslått og testet.

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

MASTER'S THESIS DESCRIPTION SHEET

Name:	Kristoffer Steinsland
Department:	Engineering Cybernetics
Thesis title (Norwegian):	Sensor fusjon av treghetssensor- og retningsmålinger fra fasesstyrt radio ved bruk av faktorgrafbasertoptimering.
Thesis title (English):	Sensor fusion of phased array radio bearing and inertial measurements using factor-graph-based optimization.

Thesis Description: Global navigation satellite systems (GNSS) is the primary outdoor positioning sensor for robotic vehicles such as unmanned arial vehicles (UAVs), unmanned surface vehicles (USVs) and unmanned ground vehicles (UGVs). However, GNSS is susceptible to interference, natural and intentional, due to its low signal power. Alternative radio positioning sources have been investigated. Phase array radio systems (PARS) is a promising technology. AoA using PARS, however, suffer from multipath errors as GNSS. Filtering of such AoA measurements based on other sensors such as inertial measurement units (IMUs) is an opportunity to handle these multipath errors. One promising filtering/estimation strategy is to employ factor-graph-based optimization (FGO).

The following tasks should be considered:

1. Perform a short literature review on
    a. previous filtering methods for integrating IMU and PARS measurements.
    b. IMU and PARS sensor errors.
    c. FGO filtering and estimation methods and solutions in context multisensory with sensor fusion of IMU and miscellaneous navigation sensors.
2. Implement a simple simulator for generating unmanned arial vehicle (UAV) motion signals. Include sensor simulation of IMU and AoA PARS sensor measurements. Add sensor errors such as noise and bias. Use simulated GNSS as a reference.
3. Choose a suitable factor-graph library/framework and implement a GNSS- and a PARS-aided INS filter using FGO. Justify your chosen factor graph framework.
4. Implement an error state Kalman filter for benchmarking purposes. Compare relevant results.
5. Test and verify your results using data from field experiments.
6. Investigate the filters' properties w.r.t. to handling PARS AoA multipath errors using outlier rejection in context used the data from the field experiments.
7. Present and discuss your results.
8. If time permits, add online mounting calibration estimation to your factor-graph-based estimation framework.
9. Conclude your results and suggest further work.

Start date:	2023-02-06
Due date:	2023-07-03
Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Associate professor Torleiv H. Bryne, Dept. of Eng. Cybernetics, NTNU

Contents

List of Figures

List of Tables

Code Listings

Acronyms

AHRS Attitude and Heading Reference System.

AoA Angle of Arrival.

ARW Angular Random Walk.

ECEF Earth Centered Earth Fixed.

ECI Earth Centered Inertial.

ESKF Error State Kalman Filter.

FGO Factor Graph Optimization.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

IMU Inertial Measurement Unit.

INS Inertial Navigation System.

ISAM Incremental Smoothing and Mapping.

LiDAR Light Detection and Ranging.

MAP Maximum A Posteriori.

NED North East Down.

PARS Phased Array Radio System.

PSD Power Spectral Density.

RMSE Root Mean Square Error.

RTK Real Time Kinematic.

SFM Structure from Motion.

SLAM Simultaneous Localization and Mapping.

SLERP Spherical Linear Interpolation.

SVD Singular Value Decomposition.

UAV Unmanned Aerial Vehicle.

VRW Velocity Random Walk.

Chapter 1

Introduction

1.1 Motivation

Global navigation satellite systems is one of the main positioning sensors for manned and unmanned vehicles. The system enables accurate navigation solutions for a vast set of systems in industries such as maritime and aviation. However, due to its low signal power, it is vulnerable to issues such as interference, jamming, and spoofing [1]. Interference and jamming can be intentional or not, but in any case, such electromagnetic disturbances can result in decreased accuracy or make the navigation system cease to function at all. In recent months, there has been a substantial increase in GNSS jamming in northern Norway, with disruptions affecting critical infrastructure like air ambulance [2, 3]. An even more severe threat, spoofing, is when a malicious third party transmits fake GNSS signals to alter the functionality of the navigation system for the end user. It has been shown experimentally that such techniques can be used to make an unmanned aerial vehicle (UAV) follow an undesirable trajectory [4]. In the worst case, such interference could go undetected.

To combat these issues and other scenarios where GNSS is not available, such as indoors, the area of GNSS-denied navigation has explored various approaches. Sensors such as visual cameras or light detection and ranging (LiDAR) can be used to obtain odometry or solve the simultaneous localization and mapping problem, and there has been substantial progress within these domains in terms of accuracy and robustness [5–7]. In indoor environments, there is usually an ample amount of features that can be detected using such sensors. However, each of these modalities have distinct failure modes: e.g. cameras can fail in textureless environments and LiDAR-based estimates might degrade in self-similar environments. Multi-modal SLAM systems have been developed to combat this [8], however, the large amount of data needed to be processed in real-time is a limiting factor. In the context of outdoor UAVs there might be little to no features present in the environment making such approaches of little benefit. One examples is UAV operations over open waters. Phased-array Radio Systems offer a different approach to GNSS-denied navigation which will be explored in this thesis.

1.2 Background

1.2.1 Phased-array Radio

Phased-array radio systems went from a theoretical idea to an application in the 1940s when it was first developed in the context of RADAR aircraft detection [9]. Since then, the technology has matured and is used in 5G cellular networks and satellite communication [10, 11]. Moreover, in 2019 with the release of the Bluetooth 5.1 specification, cheap PARS technology has become widely available and at low cost. It has been demonstrated that Bluetooth can be used to provide a UAV navigation solution up to 700 meters from the base station [12].

A phased-array radio system consists of an array of antenna elements that operate at a given phase shift that, in summation, allows for steering a beam in a particular direction relative to the array [13]. In Figure 1.1 the fundamental building blocks of a linear antenna array is displayed. N array elements are evenly placed with a distance d from one element to the next. When each signal from one antenna to the next is delayed by Δt, the following equation holds for the transmission angle θ

$$\theta = \arcsin\left(\frac{\Delta t c}{d}\right) \tag{1.1}$$

[14], where c is the speed of the signal. By controlling the phase shift through each phase shifter ϕ, the transmission angle can be precisely controlled.

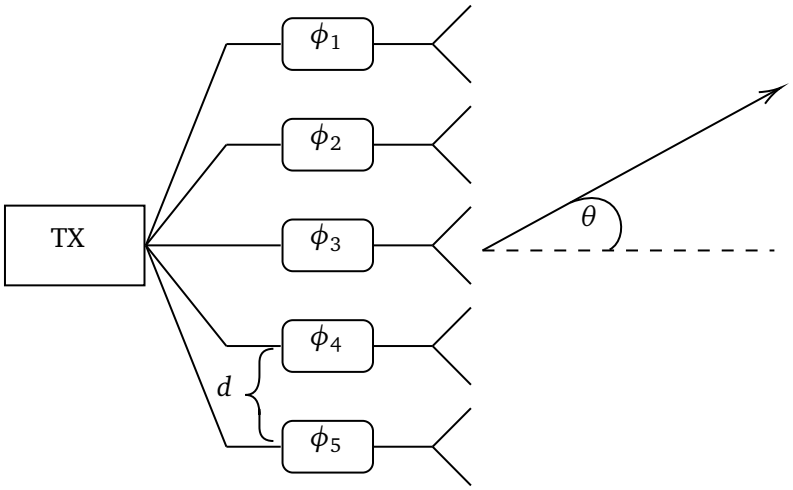


Figure 1.1: Phased-array. Image inspired by: [15] [CC0 1.0] Creative Commons

The PARS system can also be used in reverse to measure the angle of arrival (AoA) of a signal [14]. With a two-dimensional array, this gives a bearing measurement to the transmitter; that is an azimuth angle ψ and an elevation angle α from the antenna array receivers to a transmission source such as a UAV. It is possible to also get a range measurement based on for instance signal strength or round-trip timing, but this will not be a topic for this thesis; only bearing measurements are considered. Using this

type of system relieves some of the issues mentioned with standard GNSS navigation such as jamming and other types of interference. The downside of such a positioning system is the increased complexity and the need for ground equipment with limited range. Further, the pose of the ground stations needs to be determined accurately, but algorithms have been developed to handle this [16].

1.2.2 Navigation

The so-called "science of navigation" is to, determine the position, velocity, and attitude of a vehicle with respect to a known coordinate system as accurately as possible [17]. Positioning is the determination of the position only of an object without regard for velocity or attitude. There are multiple ways to get position measurements but the most prevailing one is the global navigation satellite systems (GNSS) which as of today consists of the American Global Positioning System (GPS), Russian GLONASS, Chinese BeiDou and European Galileo. GNSS can provide accurate position measurements, but at low frequency typically $1-10$Hz. Furthermore, there can be significant errors in the measurements like multipath errors where radio waves take multiple paths to the receiver resulting in obscured measurements. In contrast to GNSS which provides global position measurements at a low frequency, dead reckoning is the technique of integrating velocity estimates over time to deduce the position of a vehicle [17]. This can for example be done using a speedometer/odometer or by double integrating accelerometer measurements. Likewise, a vehicle's attitude can be determined by integrating its angular velocity which can be measured by e.g. a gyroscope. The upside of dead reckoning is that it can be done at a high rate, however, the integration of relative measurements will lead to drift, and absolute corrections from for instance GNSS or PARS are needed, leading to an aided inertial navigation system (INS). The field of combining such measurements in an optimal way is commonly known as sensor fusion where the Kalman filter and its derivatives are the standard and most common techniques, [17, 18]. However, in recent years other approaches like factor graphs have entered the scene [19] which provide a set of benefits that makes it a contender to classical methods.

1.3 Related work

There has been considerable effort to explore phased-array radio systems with respect to UAV navigation at the NTNU UAV Lab. In [20], the authors showed that PARS-aided inertial navigation was possible using a nonlinear observer. The PARS system used was a Radionor CRE2 189, providing both bearing and range. Due to signal reflections causing noise in the PARS measurements a barometer and a compass were used as well. They achieved a root mean square error (RMSE) of 26.3m compared to real-time kinematic (RTK) GNSS. In [21] this work is extended to provide a secure solution that can detect GNSS-spoofing and switch to PARS measurements should it be deemed necessary.

In the doctoral thesis [14] a significant amount of research on PARS navigation for UAVs was conducted. A multiplicative extended Kalman filter was developed to be able to estimate heading without the use of a compass and to tackle the considerable outliers

present in PARS measurements. Safety mechanisms that can switch from PARS to GNSS or opposite, were added to increase the robustness of the navigation solution. An RMSE of 24.99m was achieved using PARS only on a beyond visual line of sight fixed wing campaign. The author concludes that the main probable cause of the relatively large errors is most likely because of the misalignment of the phased-array ground station antenna.

In [16] a solution to the ground PARS mounting orientation problem was presented. They developed an online calibration algorithm using a multiplicative extended Kalman filter that can adjust the PARS alignment in real-time. The solution presented uses GNSS measurements to do the calibration.

Aided inertial navigation using factor graphs has been explored in [19], where a fixed lag smoother capable of incorporating delayed measurements was developed. In [22], a bearing-only tracker was developed using factor graphs and the `miniSAM` framework, [23]. The authors suggest switching to `GTSAM` because of consistency issues.

1.4 Problem description, research question and main contributions

The main problem tackled in this thesis is to fuse bearing measurements from PARS and inertial measurements from an IMU using factor graph optimization. More formally this can be stated as the following. Let $\mathcal{Z} = \{\alpha_0, \psi_0, \ldots, \alpha_n \psi_n\}$ be a set of noisy PARS elevation α and azimuth ψ measurements given in the radio frame that arrive incrementally. And let $\mathcal{Y} = \{\boldsymbol{f}_0 \boldsymbol{\omega}_0, \ldots, \boldsymbol{f}_m \boldsymbol{\omega}_m\}$ be set of noisy angular velocity and specific force measurements in the body frame, that arrives at a different frequency. The goal is to estimate the position $\boldsymbol{p}$, velocity $\boldsymbol{v}$, and attitude $\boldsymbol{q}$ of the UAV in real-time by a factor graph based algorithm with $\mathcal{Z}$ and $\mathcal{Y}$ as input parameters. In light of this, the main research question of this thesis is formulated as such: *Can a factor graph based approach to a PARS-aided INS provide higher accuracy and more robustness to outliers than the Kalman filter?* To answer this question the following subgoals are defined:

- Review the theory behind IMU and PARS sensor models and errors, as well as factors graphs and error state Kalman filter.
- Implement a simulator capable of generating IMU, GNSS and PARS measurements with noise.
- Implement an error state Kalman filter for benchmarking.
- Implement a factor graph based estimator.
- Test and analyze estimators in simulation and on data from field experiments.
- Test and analyze the estimator's ability to handle multipath errors using outlier rejection.
- Implement online mounting calibration of PARS in the factor-graph-bases estimator.

The main contributions of this thesis are:

- The implementation of a factor graph based PARS-aided INS.

- Analysis of FGO compared to the ESKF as PARS-aided INS.
- The implementation of FGO-based online calibration of PARS mounting pose.

1.5 Delimitations

This thesis will assume that PARS bearing measurements are given. That is, there will not be any focus on the underlying functional aspects associated with PARS, such as the algorithms that produce the measurements, electronics, antenna setup, etc. The exception to this is the PARS noise characteristics that will be explored.

1.6 Structure of the thesis

The rest of this thesis is divided up into six parts. In Chapter 2, a brief overview of the relevant Lie theory needed for the factor graph framework will be given. Then a short description of the error state Kalman filter is performed before factor graphs are presented. In Chapter 3 the simulator implementation is detailed. In Chapter 4 the design of the error state Kalman filter is briefly detailed before a deeper dive into the factor graph implementation is carried out. After this the estimators are tested in simulation and the results for this are presented and discussed in Chapter 5. Then there is a transition from simulation to data from field experiments. Chapter 6 details the experimental setup. To handle the intricacies of real PARS and flight data, outlier rejection is added to both estimators and online calibration to the factor graph. Lastly, the results from running the estimators on the field data are presented and discussed in Chapter 7.

Chapter 2

Theoretical Background

2.1 Coordinate frames

A coordinate frame can be interpreted in two ways. Firstly, a coordinate frame can be seen as a reference for motion that provides an origin and three axes. Secondly it can represent the six degrees of freedom pose of an object [17]. For describing position, velocity, acceleration and angular velocity, the same notation as Groves [17] is used: For a given quantity $\boldsymbol{x}^{\gamma}_{\beta\alpha}$, α is the *object frame* that that is moving with respect to the *reference frame* β, and γ is the frame where this motion is described in; which is known as the *resolving frame*. For a rotation there is no resolving frame, and here the notation which used is $\boldsymbol{R}^{\alpha}_{\beta}$, which acts on a point by rotating it from frame β to α. To refer to a specific frame curly brackets are often used $\{\cdot\}$. Bold symbols are used for vectors and matrices, and matrices are capitalized.

Common coordinate frames include the *earth centered inertial* (ECI) $\{i\}$, the *Earth-centered Earth-fixed* (ECEF) $\{e\}$, *North East Down* (NED) $\{n\}$ frame and *body* $\{b\}$ [17]. As the name implies ECI is an inertial frame, meaning that it does not experience any accelerations; This simplifies expressions. ECI has its origin at the center of mass of the Earth, the z-axis aligns with the Earth's spin axis, and the x-axis is fixed using stars. ECEF is similar to ECI, but ECEF rotates with the Earth; the x-axis points towards 0° longitude. Also, its origin is defined to be in the center of a *reference ellipsoid* like WGS84 that models the Earth's shape. NED is a local tangent navigation frame with the x-axis pointing north, the y-axis east and the z-axis down to the center of the reference ellipsoid. A comparison between ECEF and NED is shown in Figure 2.1. In this work NED mostly will be used. The assumption that NED is inertial will also be used, i.e. Earth's rotation will not be corrected for in the navigation equations and the the NED frame is stationary. The reasoning behind this is that the distances traveled, and time scales are small, so the errors from this assumption will be reasonably small as well. Lastly, the body frame is attached to the vehicle and has its x-axis forward, y-axis right and the z-axis pointing down.

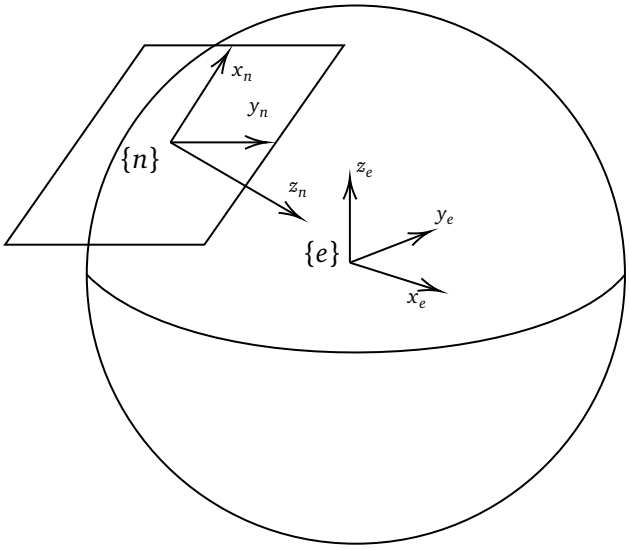


Figure 2.1: Coordinate frames NED $\{n\}$ and ECEF $\{e\}$. In NED the x-axis is pointing north, the y-axis east and the z-axis down to the center of the reference ellipsoid. For ECEF the z-axis aligns with the Earth's rotational axis, and the x-axis points towards 0° longitude.

2.2 Basic Lie theory

In state estimation and optimization, it is often desirable to take small steps along a certain direction of the state space. This is easy for positions and linear velocities as they belong to a vector space. However, for orientations, one cannot assume that a rotation plus a small increment will still be a valid rotation. Rotations and poses are unfortunately not part of a vector space, making it necessary to utilize a bit more sophisticated machinery to do state estimation. The following section will be a short reiteration of some of the relevant theory in found in "*A micro Lie theory* [24]" by Sola, and "*A handbook in visual SLAM* [25]" by Håvardsholm.

Rotations and poses are so-called *Lie Groups*, which are groups as well as smooth manifolds. A group is a set and a composition operation, denoted with "$\circ$", that satisfies the group axioms: closure property, associativity, identity element, and inverse element [24]. That Lie Groups are smooth manifolds means that at every point along the manifold, there is a well-defined tangent space. As done in [25], the tangent space of a manifold $\mathcal{M}$ at a point $\mathcal{X}$ will be denoted $\mathcal{TM}_{\mathcal{X}}$. The usefulness of Lie theory comes from the fact that operations on the manifold can be carried out in the tangent vector space meaning that standard linear algebra and calculus can be used [25]. That is, one can define increments in the tangent space which map back to the manifold, making optimization that includes orientations possible. Further, Lie groups have a *group action* which acts on other sets. Actions are denoted with a dot "$\cdot$".

The *Lie algebra* $\mathfrak{m}$ of a manifold $\mathcal{M}$ is the tangent vector space at identity $\mathcal{TM}_{\mathcal{E}}$, where $\mathcal{E}$ denotes identity. Elements of a Lie algebra are denoted with a *hat* i.e. $\boldsymbol{\tau}^{\wedge} \in \mathfrak{m}$. The Lie algebra can be expressed as a linear combination of *generators* [24]. The

coefficients of this expression make up a vector $\boldsymbol{\tau} \in \mathbb{R}^m$ with m being the degrees of freedom of the manifold. The linear maps *hat* and *vee* enable transition between the Lie algebra and the vector space [25]. That is $(\boldsymbol{\tau})^\wedge : \mathbb{R}^m \to \mathfrak{m}$ and $\left(\boldsymbol{\tau}^\wedge\right)^\vee : \mathfrak{m} \to \mathbb{R}^m$

2.2.1 The exponential and logarithmic maps

The exponential map transforms elements in the Lie algebra to the manifold. And the logarithmic map does the inverse transformation [25]

$$\exp\left(\boldsymbol{\tau}^\wedge\right) : \mathfrak{m} \to \mathcal{M} \tag{2.1}$$
$$\log\left(\mathcal{X}\right) : \mathcal{M} \to \mathfrak{m} \tag{2.2}$$

To circumvent the *vee* and *hat* operations, the capitalized maps are defined as

$$\mathrm{Exp}\left(\boldsymbol{\tau}\right) = \exp\left(\boldsymbol{\tau}^\wedge\right) : \mathbb{R}^m \to \mathcal{M} \tag{2.3}$$
$$\mathrm{Log}\left(\mathcal{X}\right) = \left(\log\left(\mathcal{X}\right)\right)^\vee : \mathcal{M} \to \mathbb{R}^m \tag{2.4}$$

The exponential map can be approximated to first order as

$$\mathrm{Exp}\left(\boldsymbol{\tau}\right) \approx \mathcal{E} + \boldsymbol{\tau}^\wedge \tag{2.5}$$

For instance, for rotation matrices, $\mathcal{E}$ is the identity matrix, and the *hat* operation gives the skew-symmetric matrix, leading to the well-known approximation. Plus is defined as increments in the tangent space mapped onto the manifold. While minus gives the tangent space difference between two manifolds .

$$\mathcal{Y} = \mathcal{X} \oplus {}^{\mathcal{X}}\boldsymbol{\tau} = \mathcal{X} \circ \mathrm{Exp}\left({}^{\mathcal{E}}\boldsymbol{\tau}\right) \in \mathcal{M} \tag{2.6}$$
$${}^{\mathcal{X}}\boldsymbol{\tau} = \mathcal{Y} \ominus \mathcal{X} = \mathrm{Log}\left(\mathcal{X}^{-1} \circ \mathcal{Y}\right) \in \mathcal{TM}_{\mathcal{X}} \tag{2.7}$$

[25]. The composition does not commute; in the above equation, plus and minus are defined on the right, which means that the perturbations are performed locally. The operations can also be carried out globally on the left side. The adjoint $\mathrm{Ad}_{\mathcal{X}}(\boldsymbol{\tau}^\wedge) = \mathcal{X}\boldsymbol{\tau}^\wedge\mathcal{X}^{-1}$ gives the relationship between these local and global tangent space vectors [25]. For matrix Lie groups the adjoint matrix can be found and used to transform vectors at the tangent space of the manifold at $\mathcal{X}$ to the origin $\mathcal{E}$, that is ${}^{\mathcal{E}}\boldsymbol{\tau} = \boldsymbol{Ad}_{\mathcal{X}}\,{}^{\mathcal{X}}\boldsymbol{\tau}$ [24].

2.2.2 Rotation matrices

There are multiple ways to represent orientations, one of them is the rotation group SO(3) consisting of matrices $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ that have the following two properties

$$\boldsymbol{R}\boldsymbol{R}^\top = \boldsymbol{I}, \quad \det\left(\boldsymbol{R}\right) = 1 \tag{2.8}$$

By time differentiating the orthogonality constraint, as in [24], it can be shown that the Lie algebra is given by the skew-symmetric matrix. That is, the *hat* operation map the vector of angular velocities to the corresponding skew-symmetric matrix, while the *vee* operation does the opposite. The skew-symmetric matrix is denoted $[\boldsymbol{\omega}]_\times$, and is given by

$$\boldsymbol{\omega}^\wedge = [\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.9}$$

In light of this, the differential equation that governs how the rotation matrix change with time, is given by $\dot{\boldsymbol{R}} = \boldsymbol{R}[\boldsymbol{\omega}]_\times$. Using the first-order approximation of exponential in (2.5) gives that the solution for a small time increment T_s is

$$\Delta\boldsymbol{R} \approx \boldsymbol{I} + [\boldsymbol{\omega} T_s]_\times \tag{2.10}$$

Rotations act on vectors by rotating them from one resolving frame to another. That is, given a rotation matrix $\boldsymbol{R}_\beta^\alpha$, then

$$\boldsymbol{x}^\beta = \boldsymbol{R}_\alpha^\beta \cdot \boldsymbol{x}^\alpha = \boldsymbol{R}_\alpha^\beta \boldsymbol{x}^\alpha \tag{2.11}$$

From the orthogonal property, it is clear that the inverse rotation is given by transposition. That is

$$\left(\boldsymbol{R}_\alpha^\beta\right)^{-1} = \left(\boldsymbol{R}_\alpha^\beta\right)^\top = \boldsymbol{R}_\beta^\alpha \tag{2.12}$$

The composition of two rotations is carried out by matrix multiplication

$$\boldsymbol{R}_\alpha^\beta = \boldsymbol{R}_\gamma^\beta \boldsymbol{R}_\alpha^\gamma \tag{2.13}$$

2.2.3 Rigid motion

The rigid motion group SE(3) consists of matrices $\boldsymbol{T} \in \mathbb{R}^{4\times4}$ that can be written as

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^\top & 1 \end{bmatrix} \tag{2.14}$$

where $\boldsymbol{R} \in \mathrm{SO}(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$. It can be shown that the hat operator gives

$$\boldsymbol{\tau}^\wedge = \begin{bmatrix} [\boldsymbol{\theta}]_\times & \boldsymbol{\rho} \\ \boldsymbol{0}^\top & 0 \end{bmatrix} \in \mathfrak{se}(3) \tag{2.15}$$

where $\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\rho} & \boldsymbol{\theta} \end{bmatrix}^\top \in \mathbb{R}^6$ is the tangent vector with the translation associated part $\boldsymbol{\rho}$ and rotation associated part $\boldsymbol{\theta}$ [24]. The SE(3) group action on vectors rotates and translates them, and is given by

$$\boldsymbol{p}^\beta = \boldsymbol{T}_\alpha^\beta \cdot \boldsymbol{p}^\alpha = \boldsymbol{R}_\alpha^\beta \boldsymbol{p}^\alpha + \boldsymbol{t}^\beta \tag{2.16}$$

As rigid motions incorporate both rotation and translation they will sometimes be referred to as a *pose*.

2.2.4 Unit quaternions

A more compact representation of rotation can be expressed using the unit quaternion. Quaternions consist of a sum of one real part and three so called "hyper-imaginary" parts. Following the notation in [26], the unit quaternion $\boldsymbol{q}$ is written as

$$\boldsymbol{q} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \tag{2.17}$$

and has the property that $\|\boldsymbol{q}\|_2 = 1$ where $\|\cdot\|_2$ represents the Euclidean norm. Composition of two rotations represented with unit quaternions is performed using multiplication which is denoted with $\otimes$.

$$\boldsymbol{q}_\alpha^\beta = \boldsymbol{q}_\gamma^\beta \otimes \boldsymbol{q}_\alpha^\gamma = \begin{bmatrix} \eta_\gamma^\beta \eta_\alpha^\gamma - \left(\boldsymbol{\epsilon}_\gamma^\beta\right)^\top \boldsymbol{\epsilon}_\alpha^\gamma \\ \eta_\gamma^\beta \boldsymbol{\epsilon}_\alpha^\gamma + \eta_\alpha^\gamma \boldsymbol{\epsilon}_\gamma^\beta + \boldsymbol{\epsilon}_\gamma^\beta \times \boldsymbol{\epsilon}_\alpha^\gamma \end{bmatrix} \tag{2.18}$$

For unit quaternions the hat operation gives the *pure quaternion* $\boldsymbol{\omega}^\wedge = \begin{bmatrix} 0 & \frac{\boldsymbol{\omega}}{2} \end{bmatrix}^\top$. The differential equation is

$$\dot{\boldsymbol{q}} = \boldsymbol{q} \otimes \boldsymbol{\omega}^\wedge = \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}$$

Again using (2.5), for a small time increment T_s, the resulting relative rotation is given by

$$\Delta\boldsymbol{q} \approx \begin{bmatrix} 1 \\ \frac{\boldsymbol{\omega} T_s}{2} \end{bmatrix} \tag{2.19}$$

where the identiy element is $\begin{bmatrix} 1 & \mathbf{0}^\top \end{bmatrix}^\top$. This is in turn an approximation of

$$\Delta\boldsymbol{q} = \begin{bmatrix} \cos(T_s/2\|\boldsymbol{\omega}\|) \\ \sin(T_s/2\|\boldsymbol{\omega}\|_2)\frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|_2} \end{bmatrix} \tag{2.20}$$

as given by [17, 27].

2.2.5 Jacobians

Having defined the right side plus and minus operations, it is now possible to define Jacobians of Lie groups as

$$\boldsymbol{J}_{\mathcal{X}}^{f(\mathcal{X})} = \lim_{\boldsymbol{\tau}\to\mathbf{0}} \frac{f(\mathcal{X} \oplus \boldsymbol{\tau}) \ominus f(\mathcal{X})}{\boldsymbol{\tau}} \tag{2.21}$$

$$\boldsymbol{J}_{\mathcal{Y}}^{\mathcal{X}} = \boldsymbol{J}_{\mathcal{Z}}^{\mathcal{X}} \boldsymbol{J}_{\mathcal{Y}}^{\mathcal{Z}} \tag{2.22}$$

By utilizing the chain rule and some common Jacobian blocks, we can compute derivatives of functions with Lie groups as inputs. The following is a list of some Jacobian blocks, compiled from [24] that will be used.

- Composition:

$$\boldsymbol{J}_{\mathcal{X}}^{\mathcal{X}\circ\mathcal{Y}} = \boldsymbol{Ad}_{\mathcal{Y}}^{-1} \tag{2.23}$$
$$\boldsymbol{J}_{\mathcal{Y}}^{\mathcal{X}\circ\mathcal{Y}} = \boldsymbol{I} \tag{2.24}$$

- Inversion

$$\boldsymbol{J}_{\mathcal{X}}^{\mathcal{X}^{-1}} = -\boldsymbol{Ad}_{\mathcal{X}} \tag{2.25}$$

- SO3 rotation action:

$$\boldsymbol{J}_{\boldsymbol{R}}^{\boldsymbol{R}\cdot\boldsymbol{x}} = -\boldsymbol{R}[\boldsymbol{x}]_{\times} \tag{2.26}$$
$$\boldsymbol{J}_{\boldsymbol{x}}^{\boldsymbol{R}\cdot\boldsymbol{x}} = \boldsymbol{R} \tag{2.27}$$

- SE3 rigid motion action:

$$\boldsymbol{J}_{\boldsymbol{T}}^{\boldsymbol{T}\cdot\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{R} & -\boldsymbol{R}[\boldsymbol{x}]_{\times} \end{bmatrix} \tag{2.28}$$
$$\boldsymbol{J}_{\boldsymbol{x}}^{\boldsymbol{T}\cdot\boldsymbol{x}} = \boldsymbol{R} \tag{2.29}$$

See [24] for derivation.

2.3 Inertial navigation

An inertial navigation system consists of an inertial measurement unit along with some algorithm that calculates estimates of the position, velocity, and attitude based on the IMU outputs in a dead reckoning fashion [17].

2.3.1 IMU

The IMU usually consists of three accelerometers and three gyroscopes mounted orthogonally to each other. The accelerometers measures the specific force that is acting on the sensor. It does not measure the acceleration as one might expect, e.g., by laying the IMU still on the ground the specific force output will be approximately 9.8 $[\mathrm{m\,s^{-2}}]$ upwards away from the Earth's center. The units reveal that this is in fact an acceleration and not a force, which adds to the confusion. However, one can think of specific force as the acceleration the sensor experiences minus gravity [17]. The gyroscopes measure angular rates, which are used to calculate the vehicle's attitude. The measurements are thus also affected by the Earth's rotation which often needs compensation. However, in this work, the effect from the Earth's rotation will be neglected. The accelerometer and gyroscope outputs might be given as samples at a specific time instance or as integrated increments; here, samples will be of concern. Combining the three accelerometers and gyroscopes with a processor, controller, and a clock in a compact package and the result is an IMU. Some IMUs also include three magnetometers to measure heading, a barometer to get altitude, or both. Depending on the placement of the IMU in relation to the body frame, adjustments need to be made. In the following section, it is assumed that the IMU coincides with the body frame. The IMU measurements are affected by errors such as scale factors, cross-coupling, and noise. To achieve the best possible performance, these quantities need to be modeled and calibrated for.

2.3.2 Inertial integration

To integrate the IMU measurements into estimates for position, velocity and attitude, the strap-down navigation equations are used. Recall that in this work, NED is treated as an inertial frame. The navigation equations are given by

$$\begin{aligned} \dot{\boldsymbol{p}}_{nb}^{n} &= \boldsymbol{v}_{nb}^{n} \\ \dot{\boldsymbol{v}}_{nb}^{n} &= \boldsymbol{R}_{b}^{n}\boldsymbol{f}_{nb}^{b} + \boldsymbol{g}_{b}^{n} \\ \dot{\boldsymbol{R}}_{b}^{n} &= \boldsymbol{R}_{b}^{n}\,\mathrm{Exp}([\boldsymbol{\omega}_{nb}^{b}]_{\times}) \end{aligned} \tag{2.30}$$

Where $\boldsymbol{p}_{nb}^{n}$ is the vehicle position, $\boldsymbol{v}_{nb}^{n}$ velocity, and $\boldsymbol{R}_{b}^{n}$ is the attitude here represented by a rotation matrix. For implementation the equations need to be discretized. With a sampling interval of T_s the attitude update is given by

$$\boldsymbol{R}_{b,k}^{n} = \boldsymbol{R}_{b,k-1}^{n}\,\mathrm{Exp}([\boldsymbol{\omega}_{nb,k-1}^{b}]_{\times} T_s) \tag{2.31}$$

The exponential can be approximated to any order depending on what accuracy is sought after. In this work, the exact exponential of a skew-symmetric matrix is used; that is

$$\mathrm{Exp}([\boldsymbol{\omega}]_{\times}) = \boldsymbol{I} + \frac{\sin(||\boldsymbol{\omega}||_2)}{||\boldsymbol{\omega}||_2}[\boldsymbol{\omega}]_{\times} + \frac{1-\cos(||\boldsymbol{\omega}||_2)}{||\boldsymbol{\omega}||_2^2}[\boldsymbol{\omega}]_{\times}^{2} \tag{2.32}$$

which is taken from equation 5.63 in [17]. For the velocity update, the average rotation $\bar{\boldsymbol{R}}_{b,k}^{n}$ between $k-1$ and k is needed, which can be approximated by simply averaging and then renormalizing the rotation. However, in this work, spherical linear interpolation is used. The average acceleration in the inertial frame is then given by

$$\bar{\boldsymbol{a}}_{nb}^{n} = \bar{\boldsymbol{R}}_{b,k}^{n}\boldsymbol{f}_{nb,k-1}^{b} + \boldsymbol{g}_{b}^{n} \tag{2.33}$$

The velocity update can then be calculated as

$$\boldsymbol{v}_{nb,k}^{n} = \boldsymbol{v}_{nb,k-1}^{n} + \bar{\boldsymbol{a}}_{nb}^{n} T_s \tag{2.34}$$

Finally the position can be integrated as

$$\boldsymbol{p}_{nb,k}^{n} = \boldsymbol{p}_{nb,k-1}^{n} + \boldsymbol{v}_{nb,k}^{n} T_s - \frac{\bar{\boldsymbol{a}}_{nb}^{n} T_s^2}{2} \tag{2.35}$$

which concludes the derivation of the inertial equations based on [17].

2.3.3 IMU errors

There are different ways to model the stochastic errors affecting an IMU, and the most suitable model is probably dependent on the IMU. In this work, the measurements are assumed to be affected by a zero mean white noise term as well as a slowly varying bias

term. As done in [26], the IMU outputs of the measured specific force $\boldsymbol{f}^b_{nb,m}$ and angular rate $\boldsymbol{\omega}^b_{nb,m}$ can be modelled as

$$\boldsymbol{f}^b_{nb,m} = \boldsymbol{R}^n_b\left(\boldsymbol{a}^b_{nb} - \boldsymbol{g}\right) + \boldsymbol{b}_f + \boldsymbol{w}_f \tag{2.36}$$

$$\boldsymbol{\omega}^b_{nb,m} = \boldsymbol{\omega}^b_{nb} + \boldsymbol{b}_\omega + \boldsymbol{w}_\omega \tag{2.37}$$

where $\boldsymbol{a}^b_{nb}$ and $\boldsymbol{\omega}^b_{nb}$ are the true acceleration and angular velocity. The biases are denoted with $\boldsymbol{b}$, and $\boldsymbol{w}$ corresponds to the white noise affecting the measurements. Again following the model in [26], the white noise processes are assumed Gaussian and their power spectral densities (PSD) are constant and given by $\boldsymbol{Q}_f$ and $\boldsymbol{Q}_\omega$. That is,

$$\boldsymbol{w}_f(t) \sim \mathcal{N}(0, \boldsymbol{Q}_v\delta(t-\tau)) \tag{2.38}$$

$$\boldsymbol{w}_\omega(t) \sim \mathcal{N}(0, \boldsymbol{Q}_w\delta(t-\tau)) \tag{2.39}$$

The values of $\boldsymbol{Q}_f$ and $\boldsymbol{Q}_\omega$ can be found in the datasheet by looking up the velocity random walk (VRW) for the accelerometer and angular random walk (ARW) for the gyroscopes. These are often given in units $[\mathrm{m\,s^{-1}\,h^{-1/2}}]$ and $[^\circ\,\mathrm{h^{-1/2}}]$ correspondingly, and should then be converted to SI-units. That is $[\mathrm{m\,s^{-2}\,Hz^{-1/2}}]$ and $[\mathrm{rad\,s^{-1}\,Hz^{1/2}}]$. If these values are not specified directly in the datasheet, they can be found by the typically provided Allan Standard Deviation (ASD) plot [28].

Unlike the uncorrelated white noise, the bias for the accelerometer $\boldsymbol{b}_f$ and the gyroscope $\boldsymbol{b}_\omega$ describes the correlated noise. The biases are often the largest source of error. They have a static component called the run-to-run bias which is constant and changes when the IMU is powered on or off, as well as in-run bias stability –sometimes named bias instability– which varies every time the IMU is powered [17] chapter 4.4.1. In the design of navigation filters, different stochastic processes can be used to model the IMU bias [18]. The simplest one is the constant bias, which in the scalar case is given by

$$\dot{b} = 0 \tag{2.40}$$

This is not recommended as it causes the navigation filters' state covariance to go to zero as time tends towards infinity [18] p. 42. Another approach is the Wiener process

$$\dot{b} = w_b \quad w_b \sim \mathcal{N}(0, \sigma^2_{w_b}\delta(t-\tau)) \tag{2.41}$$

However, this approach leads to an infinite covariance as time tends towards infinity. The last bias model that will be mentioned here is a first-order Gauss-Markov process, which is given as

$$\dot{b} = -\frac{1}{T}b + w_b\,, \quad w_b \sim \mathcal{N}(0, \sigma^2_{w_b}\delta(t-\tau)) \tag{2.42}$$

where T is the time constant. In *Navigation Filter Best Practices* [18], the Gauss-Markov process is recommended as it has a bounded covariance. To determine the value $\sigma^2_{w_b}$ for the Gauss-Markov process from the data sheet, one must look up the bias instability b_{ins}, convert it to SI-units, and then use

$$\sigma^2_{w_b} = \frac{2}{T}b^2_{ins} \tag{2.43}$$

Other sources of errors in an IMU are scale factors errors and cross-coupling errors [17] chapter 4.4.2. The scale factor error is when the output of the accelerometer or the gyroscope is proportional to the true specific force or angular velocity. Cross-coupling errors arise when there is a slight misalignment of the axes of IMU, which may cause the measurements in one axis to be sensitive to motion about, or in the direction of, one of the two orthogonal axes. In [29], Gade does a thorough analysis of scale factor and cross-coupling errors and concludes that these errors are largely dominated by the bias error. Therefore the sum of these errors can be modeled as an aggregate bias process [29] encapsulating the aforementioned error sources.

2.4 PARS noise

As mentioned there has been various tests of PARS systems in the field [12, 30]. From these works it is concluded that the main systematic source of error is multipath, which predominantly affects elevation. Multipath is when the signal reflects off some other surface before reaching the receiver causing a situation where the signal directly from the transmitter is ambiguous [12]. Further, the elevation angle is mostly affected by this error when the transmitter is at a low angle relative to the horizon; this may cause the signal to be reflected off the surface below as depicted in Figure 2.2. Because of this systematic source of error, it is not appropriate to consider the noise affecting the AoA measurements white. Instead, the noise is a form of colored noise which is correlated in time. An example of this is shown in Figure 2.3, which is an extract from the dataset that will be used and is described in Chapter 6. The noise affecting the elevation measurements is clearly correlated.

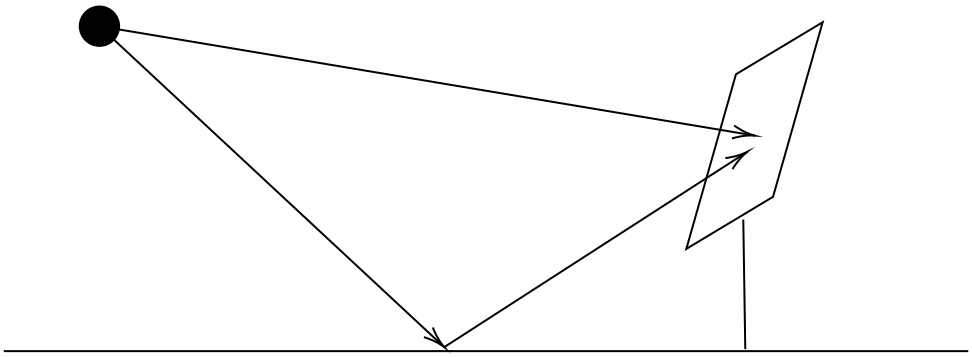


Figure 2.2: An example of multipath error. The signal from the black dot transmitter is reflected off the surface causing an ambiguity in the measured elevation angle.

The fact that the AoA measurement noise is correlated is a problem because it breaks with the assumption that the measurements are affected by white noise in the Kalman filter. Groves [17] presents a couple of ways to deal with this problem. The first is to add the the correlated noise to the state vector, as done with the biases. However, this would require the noise to be observable and adds considerable additional complexity to the estimation algorithms. Another way to deal with this problem, without explicitly

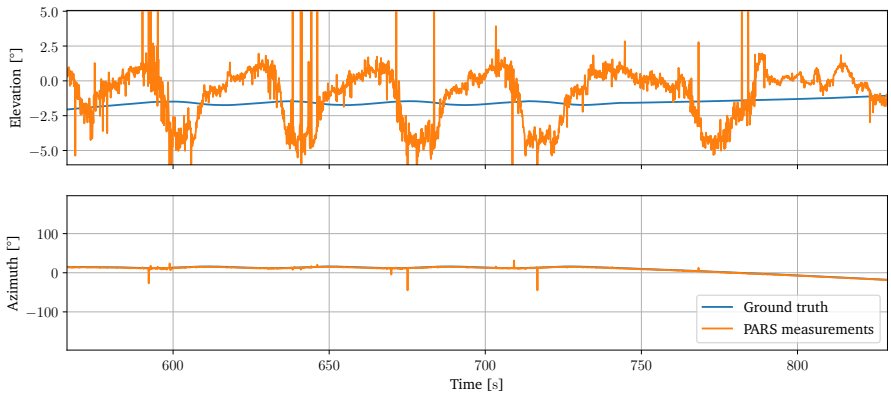


Figure 2.3: Multipath is present in the measured AoA elevation angle.

modeling it, is to simply increase the white noise covariance used in the estimators assuming that this will capture the effects of the correlated noise. This can either be done by reducing the Kalman gain or by increasing the terms in the measurement noise covariance matrix. As the latter approach is the simplest, this technique will be used in the design of the estimators in this work.

2.5 Bayesian filtering

This section comprises of the relevant theory regarding Bayesian filtering techniques which are common in inertial navigation problems. To keep the notation simpler, the coordinate super- and subscripts are dropped in this section. The theory and notation are based on "Fundamentals of Sensor Fusion" by Brekke [26].

The Bayesian filtering problem can be formulated as such: Given a state vector $\boldsymbol{x}$ of a dynamical system and a measurement vector $\boldsymbol{z}$ that are affected by noise, the goal is to combine the measurements with prior information about the system to create an accurate estimate of $\boldsymbol{x}$. The process model is written as $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})$ and describes how the state evolves in time. Here the Markov property is applied; given $\boldsymbol{x}_{k-1}$, $\boldsymbol{x}_k$ is independent of all previous states and measurements. Similarly, the measurement model is $p(\boldsymbol{z}_k \mid \boldsymbol{x}_k)$ where it also assumed that conditional on the newest state, the measurement is independent of all previous states before that and the other measurements [26]. The filtering approach is typically divided into an update step and a prediction step. The prediction provides the density of $\boldsymbol{x}_k$ conditional on all the measurements before k i.e. $\boldsymbol{z}_{1:k-1}$. It is given by the total probability theorem

$$p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k-1}) = \int p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})\, p(\boldsymbol{x}_{k-1} \mid \boldsymbol{z}_{1:k-1})\, d\boldsymbol{x}_{k-1} \tag{2.44}$$

Where $p(\boldsymbol{x}_{k-1} \mid \boldsymbol{z}_{1:k-1})$ is the previous posterior. The prediction step uses the process model to propagate the state forward in time. The update step incorporates the newest

measurements using Bayes rule to give the full posterior

$$p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k}) \propto p(\boldsymbol{z}_k \mid \boldsymbol{x}_k)\, p(\boldsymbol{x}_k \mid \boldsymbol{z}_{1:k-1}) \tag{2.45}$$

[26]. In some special cases these equations have a closed-form solution.

2.5.1 Kalman filter

Under the assumptions that mentioned densities are Gaussian and linear, the equations above simplify to what is known as the Kalman filter [26]. That is, the states and measurements are on the form

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{w}_{x,k} \\ \boldsymbol{z}_k &= \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{w}_{z,k} \end{aligned} \tag{2.46}$$

where $\boldsymbol{F}$ gives the state transition and $\boldsymbol{H}$ the measurement prediction. The noise $\boldsymbol{w}_{x,k}$ and $\boldsymbol{w}_{z,k}$ are zero-mean Gaussian with covariance $\boldsymbol{Q}$ and $\boldsymbol{R}$. Also, it assumed that $\boldsymbol{x}_0 \sim \mathcal{N}(\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0)$ Under these assumptions, it can be shown that, (2.44) and (2.45) reduces to

$$\begin{aligned} p(\boldsymbol{x}_k \mid \mathbf{z}_{1:k-1}) &= \mathcal{N}\left(\hat{\boldsymbol{x}}_k^-, \boldsymbol{P}_k^-\right) \\ p(\boldsymbol{x}_k \mid \mathbf{z}_{1:k}) &= \mathcal{N}\left(\hat{\boldsymbol{x}}_k^+, \boldsymbol{P}_k^+\right) \end{aligned} \tag{2.47}$$

where the predicted distribution has mean and covariance

$$\begin{aligned} \hat{\boldsymbol{x}}_k^- &= \boldsymbol{F}\hat{\boldsymbol{x}}_{k-1} \\ \boldsymbol{P}_k^- &= \boldsymbol{H}\boldsymbol{P}_{k-1}\boldsymbol{H}^\top + \boldsymbol{Q} \end{aligned} \tag{2.48}$$

Similarly, the update step gives the posterior mean and covariance

$$\begin{aligned} \hat{\boldsymbol{x}}_k^+ &= \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\left(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^-\right) \\ \boldsymbol{P}_k^+ &= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\,\boldsymbol{P}_k^- \end{aligned} \tag{2.49}$$

where $\boldsymbol{K}_k$ is the so-called *Kalman gain* and is given by

$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}^\top\left(\boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^\top + \boldsymbol{R}\right)^{-1} \tag{2.50}$$

which gives the basic Kalman filter equations [26].

In the case where we have a nonlinear process model $\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1})$ or measurement model $\boldsymbol{z}_k = h(x_k)$, the nonlinear function can be linearized using the Jacobians

$$\begin{aligned} \boldsymbol{F}_k &= \left.\frac{\partial f(\boldsymbol{x}_{k-1})}{\partial \boldsymbol{x}_{k-1}}\right|_{\boldsymbol{x}_{k-1}=\hat{\boldsymbol{x}}_{k-1}^+} \\ \boldsymbol{H}_k &= \left.\frac{\partial h(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k}\right|_{\boldsymbol{x}_k=\hat{\boldsymbol{x}}_k^-} \end{aligned} \tag{2.51}$$

The only change from the standard Kalman filter is to calculate these Jacobians $\boldsymbol{A}$ and $\boldsymbol{H}$ and to predict states and measurements using the nonlinear functions. This is called the *extended Kalman filter* (EKF) as it is extended to also work on nonlinear models.

2.5.2 Error state Kalman filter

The following will be a short summarization of the error state Kalman filter as presented by Brekke [26] and Sola [27]. In inertial navigation, an estimate of position, velocity, and attitude is sought after. As seen, singularity-free representations of attitude include unit quaternions and rotation matrices. However, the problem with these types of representations is that they are overparameterized; an orientation has three degrees of freedom while these representations have four and nine dimensions. This would in turn give a 4x4 or 9x9 covariance matrix, which is troublesome [26]. Because of this, a three-dimensional attitude error is used instead. The error state is always small meaning that the singularities won't be an issue [27].

The Error State Kalman Filter keeps track of a nominal state $\boldsymbol{x}$ and an error state $\delta\boldsymbol{x}$ [27]. The true state is given as $\boldsymbol{x}_t = \boldsymbol{x} + \delta\boldsymbol{x}$. The covariance of the error state and the mean of the nominal state is propagated for each new IMU measurement; they are treated as control input. When a new sensor measurement arrives the nominal state is updated using the error state covariance and injected into the true state and then reset. In this work, the same formulation as in [26] is used. That is, the state vector is defined as 16-dimensional consisting of position $\boldsymbol{p}$, velocity $\boldsymbol{v}$, attitude $\boldsymbol{q}$ – represented by a unit quaternion –, accelerometer bias $\boldsymbol{b}_f$ and gyroscope bias $\boldsymbol{b}_\omega$:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{v} \\ \boldsymbol{q} \\ \boldsymbol{b}_f \\ \boldsymbol{b}_\omega \end{bmatrix} \tag{2.52}$$

Modifying (2.30) to include the IMU biases modeled as Gauss-Markov processes, adding noise terms, and approximating the attitude exponential to first order gives the following equations for the true state kinematics

$$\dot{\boldsymbol{x}}_t = \begin{bmatrix} \boldsymbol{v}_t \\ \boldsymbol{R}(\boldsymbol{q}_t)\left(\boldsymbol{f} - \boldsymbol{b}_{f,t} - \boldsymbol{w}_f\right) + \boldsymbol{g} \\ \boldsymbol{q}_t \otimes \left(\boldsymbol{\omega} - \boldsymbol{b}_{\omega,t} - \boldsymbol{w}_\omega\right)^\wedge \\ -\dfrac{1}{T_f}\boldsymbol{b}_{f,t} + \boldsymbol{w}_{b_f} \\ -\dfrac{1}{T_\omega}\boldsymbol{b}_{\omega,t} + \boldsymbol{w}_{b_\omega} \end{bmatrix} \tag{2.53}$$

The coordinate frames have been removed for simplicity. The presence of the quaternion product is the reason why this filter is sometimes called the multiplicative Kalman filter

(MKF). The nominal state is the same but without the noise terms.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{R}(\boldsymbol{q})\left(\boldsymbol{f} - \boldsymbol{b}_f\right) + \boldsymbol{g} \\ \boldsymbol{q} \otimes (\boldsymbol{\omega} - \boldsymbol{b}_\omega)^\wedge \\ -\frac{1}{T_f}\boldsymbol{b}_f \\ -\frac{1}{T_\omega}\boldsymbol{b}_\omega \end{bmatrix} \tag{2.54}$$

The error state $\delta\boldsymbol{x}$ is the difference between the true state and the nominal state. Approximations based on neglecting small values and first order expansion of the attitude increment as in [27] and [26] can be performed to arrive at linear error kinematics

$$\delta\dot{\boldsymbol{x}} = \begin{bmatrix} \delta\boldsymbol{v} \\ -\boldsymbol{R}(\boldsymbol{q})\left[\boldsymbol{f} - \boldsymbol{b}_f\right]_\times \delta\boldsymbol{\theta} - \boldsymbol{R}(\boldsymbol{q})\delta\boldsymbol{b}_f - \boldsymbol{R}(\boldsymbol{q})\boldsymbol{w}_f \\ -[\boldsymbol{\omega} - \boldsymbol{b}_\omega]\,\delta\boldsymbol{\theta} - \delta\boldsymbol{b}_\omega - \boldsymbol{w}_\omega \\ -\frac{1}{T_f}\delta\boldsymbol{b}_f + \boldsymbol{w}_{b_f} \\ -\frac{1}{T_\omega}\delta\boldsymbol{b}_\omega + \boldsymbol{w}_{b_\omega} \end{bmatrix} \tag{2.55}$$

It's important to note that $\delta\boldsymbol{x}$ is 15-dimensional because the full quaternion $\boldsymbol{q}$ is replaced by the 3-dimensional error rotation vector $\delta\boldsymbol{\theta}$. The prediction phase of the ESKF consists of propagating the nominal state and the error state covariance.

The update step assumes that the sensor used has a measurement model on the form

$$\mathbf{z} = \mathbf{h}(\boldsymbol{x}_t) + \mathbf{w} \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{2.56}$$

The corrected estimates for the error state mean and covariance are given by

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}^\top \left(\boldsymbol{H}\boldsymbol{P}_k^- \boldsymbol{H}^\top + \boldsymbol{R}\right)^{-1} \tag{2.57}$$

$$\delta\hat{\boldsymbol{x}}_k = \boldsymbol{K}_k\left(\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_k)\right) \tag{2.58}$$

$$\boldsymbol{P}_k^+ = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\,\boldsymbol{P}_k^- \tag{2.59}$$

The $\boldsymbol{H}$ is the Jacobian of $\boldsymbol{h}$ with respect to the error state evaluated at the nominal state. Recall that $\boldsymbol{x}_t$ is the summation of the nominal state with the error state which contains a quaternion multiplication. From the chain rule

$$\boldsymbol{H} = \left.\frac{\partial h}{\partial \boldsymbol{x}_t}\right|_{\boldsymbol{x}} \left.\frac{\partial \boldsymbol{x}_t}{\partial \delta\boldsymbol{x}}\right|_{\boldsymbol{x}} \tag{2.60}$$

Where $\boldsymbol{H}_{\boldsymbol{x}}$ is the same Jacobian as calculated in the EKF, and it is shown in [27] that

$$\left.\frac{\partial \boldsymbol{x}_t}{\partial \delta\boldsymbol{x}}\right|_{\boldsymbol{x}} = \begin{bmatrix} \mathbf{I} & 0 & 0 \\ 0 & \boldsymbol{Q}_{\delta\boldsymbol{\theta}} & 0 \\ 0 & 0 & \mathbf{I} \end{bmatrix} \tag{2.61}$$

with

$$\boldsymbol{Q}_{\delta\boldsymbol{\theta}} = \frac{1}{2}\begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \eta & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix} \tag{2.62}$$

When the error state is estimated it needs to be injected into the nominal state. Which, as mentioned, is just an addition for all the vectors and multiplication for the quaternion. Lastly, the error state mean estimate is reset to zero, and the covariance need to be updated like this

$$\boldsymbol{P}_k^+ = \boldsymbol{G}\boldsymbol{P}_k^+\boldsymbol{G}^\top \tag{2.63}$$

where

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{I} & 0 & 0 \\ 0 & \boldsymbol{I} - \left[\frac{1}{2}\delta\hat{\boldsymbol{\theta}}\right]_\times & 0 \\ 0 & 0 & \boldsymbol{I} \end{bmatrix} \tag{2.64}$$

which completes the short recap of the ESKF as presented by Sola [27] and Brekke [26].

2.6 Factor graphs

The following is a brief reiteration of some theory on MAP inference and factor graphs as presented in *Factor graphs for robot perception* [31] by Dellaert. As a supplementary source [25] has also been used. The factor graph framework that will be explained shortly provide an intuitive way of modelling sensor fusion problems and has associated with it a set of algorithms that provide a general way to exploit the sparsity typically found in these problems. In the context of SLAM, factor graphs have become a standard tool with the development of frameworks such as `GTSAM` [32], `miniSAM` [23], `Symforce` [33] and `WOLF` [34].

2.6.1 A brief introduction to factor graphs

Instead of computing the entire posterior distribution, it is possible to find only the state that maximizes the full posterior, this is called *maximum a posteriori* (MAP) estimation [35]. In other words, with a set of states X and measurement Z, one can calculate the maximum of the posterior density $p(X \mid Z)$. The MAP estimator is given by

$$\hat{X} = \underset{X}{\operatorname{argmax}}\, p(X \mid Z) \tag{2.65}$$

One way to encode the factorization of the posterior is to use a factor graph. A factor graph is a graph which consists of two types of nodes: factors and variables [31]. A simple factor graph example with only one measurement is shown in Figure 2.4. The factors that represent the dependencies between variables are denoted with ϕ, and the variables are $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{p}_1\}$. A factor graph can encode the factorization the conditional

density in a way that makes it clear which variables are independent. In Figure 2.4 the factorization

$$p(X \mid Z) \propto p(\boldsymbol{x}_1)p(\boldsymbol{x}_2 \mid \boldsymbol{x}_1)p(\boldsymbol{x}_3 \mid \boldsymbol{x}_2)p(\boldsymbol{p}_1)l(\boldsymbol{x}_1, \boldsymbol{p}_1; \boldsymbol{z}) \tag{2.66}$$

can be represented by the factor graph factorization

$$\phi(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{p}_1) = \phi_1(\boldsymbol{x}_1)\phi_2(\boldsymbol{x}_1, \boldsymbol{x}_2)\phi_3(\boldsymbol{x}_2, \boldsymbol{x}_3)\phi_4(\boldsymbol{p}_1)\phi_5(\boldsymbol{x}_2, \boldsymbol{p}1) \tag{2.67}$$

Here the $l(\boldsymbol{x}_1, \boldsymbol{p}_1; \boldsymbol{z})$ refers to the likelihood of $\boldsymbol{x}_1$ and $\boldsymbol{p}_1$ given $\boldsymbol{z}$, which is in general not a Gaussain density. The ability of the factor graph to utilize these likelihoods is one of their main benefits compared to other graphical models like the Bayes net. In fact, factor graphs can be used to represent any factored function, not just probability densities [31].

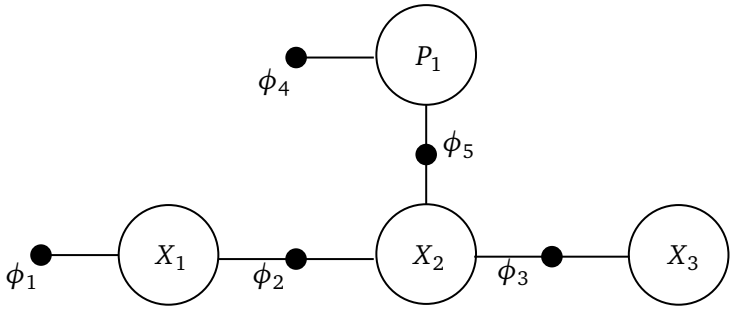


Figure 2.4: Factor graph representation of an estimation problem with three vehicles poses $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$ and one PARS with pose $\boldsymbol{p}_1$. The factors are corresponds to the black circles denoted with ϕ.

To show how one can obtain the MAP estimate let us consider a scenario with only measurements. Under the assumption that the state $\boldsymbol{x}_k \in X$ is related to the measurement $\boldsymbol{z}_k \in Z$ in the following way

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k \tag{2.68}$$

where $\boldsymbol{w}_k$ is Gaussian white noise with covariance $\boldsymbol{\Sigma}$, it can be shown that the MAP problem reduces to finding the minimum of the sum of residuals

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_{k=1}^{n} ||\boldsymbol{h}(\boldsymbol{x}_k) - \boldsymbol{z}_k||^2_{\boldsymbol{\Sigma}} \tag{2.69}$$

where the notation $||\cdot||^2_{\boldsymbol{\Sigma}}$ denotes the *Mahalanobis distance* squared, which is

$$\begin{aligned} ||\boldsymbol{h}(\boldsymbol{x}_k) - \boldsymbol{z}_k||^2_{\boldsymbol{\Sigma}} &= (\boldsymbol{h}(\boldsymbol{x}_k) - \boldsymbol{z}_k)^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{h}(\boldsymbol{x}_k) - \boldsymbol{z}_k) \\ &= \left(\boldsymbol{\Sigma}^{-\frac{1}{2}} (\boldsymbol{h}(\boldsymbol{x}_k) - \boldsymbol{z}_k)\right)^\top \left(\boldsymbol{\Sigma}^{-\frac{1}{2}} (h(\boldsymbol{x}_k) - \boldsymbol{z}_k)\right) \\ &= ||\boldsymbol{\Sigma}^{-\frac{1}{2}} (\boldsymbol{h}(\boldsymbol{x}_k) - \boldsymbol{z}_k)||^2_2 \end{aligned} \tag{2.70}$$

[25, 31]. By linearizing the measurement function $\boldsymbol{h}$ as

$$\boldsymbol{h}(\bar{\boldsymbol{x}}_k + \boldsymbol{\Delta x}) \approx \boldsymbol{h}(\bar{\boldsymbol{x}}_k) + \boldsymbol{H}_k \boldsymbol{\Delta x} \tag{2.71}$$

where $\boldsymbol{H}_k$ is the Jacobian of $\boldsymbol{h}$ evaluated at $\bar{\boldsymbol{x}}_k$, the problem can be recast as a linear least squares problem

$$\begin{aligned}\boldsymbol{\Delta x}^* &= \underset{\boldsymbol{\Delta}}{\operatorname{argmin}} \sum_{k=1}^{n} ||\boldsymbol{\Sigma}_i^{-\frac{1}{2}} \left(\boldsymbol{h}(\bar{\boldsymbol{x}}_k) + \boldsymbol{H}_k \boldsymbol{\Delta} - \boldsymbol{z}_k\right)||_2^2 \\ &= \underset{\boldsymbol{\Delta x}}{\operatorname{argmin}} \sum_{k=1}^{n} ||\boldsymbol{A}_k \boldsymbol{\Delta x} - \boldsymbol{b}_k||_2^2\end{aligned} \tag{2.72}$$

where

$$\boldsymbol{A}_k = \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{H}_k \tag{2.73}$$

and

$$\boldsymbol{b}_k = \boldsymbol{\Sigma}^{-\frac{1}{2}} \left(\boldsymbol{z}_k - \boldsymbol{h}(\bar{\boldsymbol{x}}_k)\right) \tag{2.74}$$

[25, 31]. This also works with any Lie group not just vectors, but the linearization is given as

$$\boldsymbol{h}(\bar{\mathcal{X}}_k \oplus \boldsymbol{\tau}_k) \approx \boldsymbol{h}(\bar{\mathcal{X}}_k) + \boldsymbol{J}^{\boldsymbol{h}}_{\bar{\mathcal{X}}_k} \boldsymbol{\tau}_k \tag{2.75}$$

where the Jacobian $\boldsymbol{J}^{\boldsymbol{h}}_{\bar{\mathcal{X}}_k}$ and the tangent space vector $\boldsymbol{\tau}_k$ are defined as in Section 2.2.5. Once the problem is linearized, the least squares solution is found by solving the *normal equations*

$$\left(\boldsymbol{A}_k^\top \boldsymbol{A}_k\right) \boldsymbol{\tau}_k = \boldsymbol{A}_k^\top \boldsymbol{b}_k \tag{2.76}$$

This can in turn be solved using for instance Cholesky or QR factorization [31]. After the step $\boldsymbol{\tau}_k$ is taken, the measurement function is relinearized and the process is repeated until convergence. This is called the *Gauss-Newton* method. A problem with this method is that $\left(\boldsymbol{A}_k^\top \boldsymbol{A}_k\right)$ might not be positive definite, resulting in the possibility of the cost actually increasing for some steps [25]. A solution to this is to add a small constant λ to this term such that the normal equation becomes

$$\left(\boldsymbol{A}_k^\top \boldsymbol{A}_k + \lambda \boldsymbol{I}\right) \boldsymbol{\tau}_k = \boldsymbol{A}_k^\top \boldsymbol{b}_k \tag{2.77}$$

This modification leads to the *Levenberg-Marquardt* algorithm where every step is guaranteed not to increase the cost. It is also possible to combine the different $\boldsymbol{A}_k$ and $\boldsymbol{b}_k$ into one block matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b}$. The block structure of $\boldsymbol{A}$ is closely related to the factor graph representation.

Another way to perform inference on a factor graph, which exploits the graph structure, is to use the *elimination algorithm*. This algorithm turns the posterior into a form that makes it easy to compute the MAP solution. It provides a way to factorize the $\boldsymbol{A}$ matrix into a triangular form, and QR and Cholesky can be considered specific cases. Once the system is in this form, it can easily be solved for each variable using back-substitution [31]. Importantly, the elimination algorithm is a way to convert the factor graph into a *Bayes* net, but to keep this section concise, the reader is referred to [31], which makes this connection clear.

This far, only batch estimation has been discussed, while typically in navigation problems, the measurements arrive incrementally, and it would be beneficial to reuse previous computations as well as bounding the growth of the problem. These are the problems

addressed by the *incremental smoothing and mapping* (ISAM) algorithm [36] and its successor ISAM2 [37]. As opposed to filtering which marginalizes out the variables on each iteration, a fixed lag smoother sustain the densities of a subset of historical historical states and marginalizes out those variables not in the subset [31].

The original ISAM algorithm works by incrementally updating the factorization of the linear system $(\boldsymbol{A}, \boldsymbol{b})$ using QR-updating [31] when a new measurement arrives. In this way, previous calculations are utilized. However, one problem with this approach is that it requires the relinerization to be done on the full graph, and since this as an expensive operation the relinerization step is only performed periodically. ISAM 2 deals with this shortcoming by introducing a Bayes tree data structure which makes it possible to selectively relinearize factors in the graph whose estimate deviate from the linearization point more than a selected threshold [31].

2.6.2 IMU factors

In inertial navigation IMU measurements arrive at a high rate, typically 100 Hz or higher. Creating a factor for each of these measurements would probably be infeasible in the long run, due to to computational constraints. This problem is tackled by the so-called inertial preintegration [38, 39]. This technique allows for integrating IMU measurements into relative motion constraints. In this way, all IMU measurements are summarized into one factor between the navigation states. This approach was first proposed by [38] and extended by Forster in [39] where the integration is performed on the SO(3) manifold. Another consideration that needs to be addressed is that by just integrating the IMU measurements, the integration has to be repeated whenever the linearization point changes [39]. Therefore, in the method of Forster, the relative pose and velocity increments are computed independently of a specific linearization point, meaning that the motion constraints do not have to be recomputed during the optimization. All the derivations presented by Forster are too comprehensive to be repeated in detail here. However, it can be said that Forster starts by defining the following relative motion increments

$$\Delta \boldsymbol{R}_{ij} = \prod_{k=i}^{j-1} \mathrm{Exp}((\boldsymbol{\omega}_k - \boldsymbol{b}_{\omega,k} - \boldsymbol{w}_\omega)T_s) \tag{2.78}$$

$$\Delta \boldsymbol{v}_{ij} = \sum_{k=i}^{j-1} \Delta \boldsymbol{R}_{ik}((\boldsymbol{f}_k - \boldsymbol{b}_{f,k} - \boldsymbol{w}_f)T_s) \tag{2.79}$$

$$\Delta \boldsymbol{p}_{ij} = \sum_{k=i}^{j-1} \left(\Delta \boldsymbol{v}_{ij} T_s + \frac{1}{2} \Delta \boldsymbol{R}_{ik}(\boldsymbol{f}_k - \boldsymbol{b}_{f,k} - \boldsymbol{w}_f)T_s^2 \right) \tag{2.80}$$

where T_s is the sampling interval. That is, the relative rotation from time i to j is equal to the product of the individual rotations yielded by the rotation prediction function defined on the angular velocities in (2.31). The same is true for velocity and position, but here the integration is done with the sum over the rotated specific force measurements as in (2.34) and (2.35). Forster then shows how to isolate the noise terms by doing a first-order approximation, and assuming the bias is constant between time i and j, which

then yields the final preintegrated measurement model on the form

$$\Delta\tilde{\boldsymbol{R}}_{ij} = \boldsymbol{R}_i^\top \boldsymbol{R}_j \operatorname{Exp}(\delta\boldsymbol{\phi}_{ij}) \tag{2.81}$$

$$\Delta\tilde{\boldsymbol{v}}_{ij} = \boldsymbol{R}_i^\top \left(\boldsymbol{v}_j - \boldsymbol{v}_i - \boldsymbol{g}\, T_{ij}\right) + \delta\boldsymbol{v}_{ij} \tag{2.82}$$

$$\Delta\tilde{\boldsymbol{p}}_{ij} = \boldsymbol{R}_i^\top \left(\boldsymbol{p}_j - \boldsymbol{p}_i - \boldsymbol{v}_i T_{ij} - \frac{1}{2}\boldsymbol{g}\, T_{ij}\right) + \delta\boldsymbol{p}_{ij} \tag{2.83}$$

where the quantities $\delta\boldsymbol{\phi}_{ij}$, $\delta\boldsymbol{v}_{ij}$ and $\delta\boldsymbol{p}_{ij}$ are noise terms. Further, the noise affecting the rotation $\delta\boldsymbol{\phi}_{ij}$, is defined in the tangent space of the SO(3) manifold. For derivations on noise propagation, bias update, and Jacobians the reader is referred to [39].

2.6.3 GNSS factor

Compared to the IMU factor, the GNSS factor is substantially more simple. However, if it is assumed that the vehicle state is $\boldsymbol{T}_b^n$, the derivation is a little less intuitive than expected. The measurement function is then

$$\boldsymbol{h}(\boldsymbol{T}_b^n) = \boldsymbol{p}_{nb}^n \tag{2.84}$$

where $\boldsymbol{p}_{nb}^n$ is the translation part of the rigid transformation. One way to find the Jacobian, is by using the definition

$$\begin{aligned}
\boldsymbol{J}_{\boldsymbol{T}_b^n}^{\boldsymbol{h}(\boldsymbol{T}_b^n)} &= \lim_{\tau\to 0} \frac{\boldsymbol{h}(\boldsymbol{T}_b^n \oplus \boldsymbol{\tau}) \ominus \boldsymbol{h}(\boldsymbol{T}_b^n)}{\boldsymbol{\tau}} \\
&= \lim_{\tau\to 0} \frac{\boldsymbol{h}(\boldsymbol{T}_b^n \operatorname{Exp}(\tau)) - \boldsymbol{h}(\boldsymbol{T}_b^n)}{\boldsymbol{\tau}} \\
&= \lim_{\tau\to 0} \frac{\boldsymbol{h}(\boldsymbol{T}_b^n (\boldsymbol{I} + \boldsymbol{\tau}^\wedge) - \boldsymbol{h}(\boldsymbol{T}_b^n)}{\boldsymbol{\tau}} \\
&= \lim_{\tau\to 0} \frac{\boldsymbol{h}\left(\begin{bmatrix} \boldsymbol{R}_b^n(\boldsymbol{I} + [\boldsymbol{\theta}]_\times) & \boldsymbol{R}_b^n\boldsymbol{\rho} + \boldsymbol{p}_{nb}^n \\ \boldsymbol{0} & 1 \end{bmatrix}\right) - \boldsymbol{h}(\boldsymbol{T}_b^n)}{\boldsymbol{\tau}} \\
&= \lim_{\tau\to 0} \frac{\boldsymbol{R}_b^n\boldsymbol{\rho} + \boldsymbol{p}_{nb}^n - \boldsymbol{p}_{nb}^n}{\boldsymbol{\tau}} \\
&= \lim_{\tau\to 0} \frac{\begin{bmatrix} \boldsymbol{R}_b^n & \boldsymbol{0}^\top \end{bmatrix}\boldsymbol{\tau}}{\boldsymbol{\tau}} \\
&= \begin{bmatrix} \boldsymbol{R}_b^n & \boldsymbol{0}^\top \end{bmatrix}
\end{aligned} \tag{2.85}$$

where $\boldsymbol{\rho}$ and $\boldsymbol{\theta}$ are the translation and rotation associated components of the tangent space vector, that is $\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\rho} & \boldsymbol{\theta} \end{bmatrix}^\top$. This Jacobian might seem strange as one would perhaps expect the Jacobian to be identity. However, recall that the Jacobian was defined using perturbation on the right side, where $\boldsymbol{\tau}$ is in local coordinates.

Chapter 3

Simulator Design

To aid the development of the filters, a simulator was created capable of generating IMU accelerometer and gyroscope measurements as well as measurements for any sensor model.

3.1 IMU measurements

To generate the IMU measurements the technique described in Appendix J of Groves [17] was used. The IMU simulator takes in a motion profile consisting of position $\boldsymbol{v}^n_{nb}$ and attitude $\boldsymbol{R}^n_b$ motions as well as IMU parameters such as noise and sampling interval T_{IMU} and generates velocity $\boldsymbol{v}^n_{nb}$, specific force $\boldsymbol{f}^b_{nb}$ and angular velocity $\boldsymbol{\omega}^b_{nb}$. The velocity is first calculated in the inertial frame as

$$\boldsymbol{v}^n_{nb,k+1} = 2\frac{\boldsymbol{p}^n_{nb,k+1} - \boldsymbol{p}^n_{nb,k}}{T_{mp}} - \boldsymbol{v}^n_{nb,k} \tag{3.1}$$

where T_{mp} is the step length in the provided motion profile. The initial velocity $\boldsymbol{v}_0$ must also be provided. Then the position, velocity, and attitude are resampled at T_{IMU} using a linear interpolator. Proper rotational interpolations like spherical linear interpolation (SLERP), [27] should be used for attitude, but good enough results were achieved by doing the linear interpolation and renormalizing the rotation. Attempts were made to interpolate prior to the velocity calculation, but this resulted in the accumulation of large errors. After this, the average acceleration is calculated by

$$\bar{\boldsymbol{a}}^n_{nb} = \frac{\boldsymbol{v}^n_{nb,k+1} - \boldsymbol{v}^n_{nb,k}}{T_{\text{IMU}}} \tag{3.2}$$

which is made into a specific force by

$$\boldsymbol{f}^n_{nb,k} = \bar{\boldsymbol{a}}^n_{nb} - \boldsymbol{g} \tag{3.3}$$

Proper calculation of gravity at a specific latitude and height was done based on derivations in [17]. The script was given the latitude of Raudstein at Agdenes in Trøndelag,

Norway and the gravity vector was set to

$$\boldsymbol{g} = \begin{bmatrix} -2.8710430 \times 10^{-}7 \\ 0 \\ 9.8217694 \end{bmatrix} \tag{3.4}$$

This is the gravity vector that is used throughout this whole project. The acceleration must be converter to the body frame so the average rotation $\bar{\boldsymbol{R}}_b^n$ is needed. For this the SLERP implementation in `GTSAM` was used

$$\bar{\boldsymbol{R}}_b^n = \texttt{slerp}\left(\boldsymbol{R}_{b,k}^n, \boldsymbol{R}_{b,k+1}^n, \texttt{0.5}\right) \tag{3.5}$$

Finally, the specific force in the body frame can be recovered as

$$\boldsymbol{f}_{nb,k}^b = \bar{\boldsymbol{R}}_b^n \boldsymbol{f}_{nb,k}^n \tag{3.6}$$

To generate gyroscope angular velocity measurements, the rotational difference between the two steps is first found

$$\Delta\boldsymbol{R} = (\boldsymbol{R}_{b,k+1}^n)^\top \boldsymbol{R}_{b,k}^n \tag{3.7}$$

And the attitude increment is calculated using [17, Eq. (5.64)]

$$\Delta\boldsymbol{\theta} = \frac{\theta}{2} \begin{bmatrix} \Delta\boldsymbol{R}[2,1] - \Delta\boldsymbol{R}[1,2] \\ \Delta\boldsymbol{R}[0,2] - \Delta\boldsymbol{R}[2,0] \\ \Delta\boldsymbol{R}[1,0] - \Delta\boldsymbol{R}[0,1] \end{bmatrix} \tag{3.8}$$

where

$$\theta = \frac{\mu}{\sin(\mu)} \qquad \mu = \cos^{-1}\left(\frac{\text{tr}(\Delta\boldsymbol{R}) - 1}{2}\right) \tag{3.9}$$

If $\mu < \epsilon$ then $\theta \leftarrow 1$. Here, ϵ was set to 2×10^{-5} as in [17]. The notation $\text{tr}(\cdot)$ is used for the matrix trace. This gives the attitude increment, and the angular velocity can be recovered by simply dividing by the sample time, i.e.,

$$\boldsymbol{\omega}_{nb,k}^n = \frac{\Delta\boldsymbol{\theta}}{T_{\text{IMU}}} \tag{3.10}$$

To test the simulator, specific force and angular velocity measurements were generated for a 15-minute flight, and the strapdown equations were used to recover the flight path. The result showed an error in the millimeter range for position. The flight path used in the simulations is taken from a dataset [40] used as part of the course TTK4250 Sensor Fusion.

3.2 IMU noise

The IMU noise model detailed in Subsection 2.3.3 was used. Noise is added by specifying an IMU by covariances of the noise that directly affect the accelerometer and gyroscope measurements: $\boldsymbol{Q}_f$ and $\boldsymbol{Q}_\omega$. The noise is assumed uncorrelated and equal for each axis in both sensors, so

$$\boldsymbol{Q}_f = \sigma_f^2 \boldsymbol{I}, \qquad \boldsymbol{Q}_\omega = \sigma_\omega^2 \boldsymbol{I} \tag{3.11}$$

is used. For axis $a \in \{x,\ y,\ z\}$ the accelerometer and gyroscope noise was created as

$$w_{f_a,k} \sim \mathcal{N}\left(0, \frac{\sigma_f^2}{T_{\text{IMU}}}\right) \qquad w_{\omega_a,k} \sim \mathcal{N}\left(0, \frac{\sigma_\omega^2}{T_{\text{IMU}}}\right) \tag{3.12}$$

The drifting bias of the IMU was simulated using a Gauss-Markov process. Like before, the bias noise covariances are written as

$$\boldsymbol{Q}_{f_b} = \sigma_{f_b}^2 \boldsymbol{I}, \qquad \boldsymbol{Q}_{\omega_b} = \sigma_{\omega_b}^2 \boldsymbol{I} \tag{3.13}$$

The time constants T_f and T_ω must also be specified. Another technique from Appendix J in Groves [17] was used to simulate the Gauss-Markov process. The accelerometer biases are calculated as

$$b_{f_a,k} = \exp\left(\frac{-T_{\text{IMU}}}{T_f}\right) b_{f_a,k-1} + w_{f_a,k} \tag{3.14}$$

with

$$w_{f_a,k} \sim \mathcal{N}\left(0, \sigma_{f_b}^2 \left(1 - \exp\left(\frac{-2T_{\text{IMU}}}{T_f}\right)\right)\right) \tag{3.15}$$

The same thing is done for the gyroscope biases $b_{\omega_a,k}$. Finally, the noise and biases are added to get the noisy measurements

$$\begin{aligned} \tilde{\boldsymbol{f}}_k &= \boldsymbol{f}_k + \boldsymbol{w}_{f,k} + \boldsymbol{b}_{f,k} \\ \tilde{\boldsymbol{\omega}}_k &= \boldsymbol{\omega}_k + \boldsymbol{w}_{\omega,k} + \boldsymbol{b}_{\omega,k} \end{aligned} \tag{3.16}$$

To make the simulation realistic, there was a need for some real-world IMU noise statistics, so the IMU simulated in the experiments was chosen to be the tactical grade STIM300 by sensonor as this is used with the PARS dataset. The noise values were acquired from the datasheet, and corresponding SI-units conversion are displayed in Table 3.1.

3.3 Sensor measurements

There was a need to generate measurements from other sensors such as GNSS and PARS. To do this a sensor time interval T_{sens} is specified as well as a measurement model on the form

$$\boldsymbol{z} = h(\boldsymbol{x}) + \boldsymbol{w}_z \qquad \boldsymbol{w}_z \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_z) \tag{3.17}$$

Table 3.1: IMU parameters used in the simulation. The relation between bias noise variance, the time constant, and the bias instability is $\sigma_b^2 = 2b_{ins}^2/T$. Further, The units [mg] refer to milli-standard-gravity i.e $9.80665 \cdot 10^{-3}$.

	Data sheet	SI-Units
Accelerometer		
Velocity Random Walk σ_f	$0.07\,[\mathrm{m\,s^{-1}\,h^{-1/2}}]$	$1.2 \times 10^{-3}\,[\mathrm{m\,s^{-2}\,Hz^{-1/2}}]$
Bias Instability b_{ins_f}	$0.05\,[\mathrm{mg}]$	$4.9 \times 10^{-4}\,[\mathrm{m\,s^{-2}}]$
Time constant T_f	—	$3.6 \times 10^{2}\,[\mathrm{s}]$
Bias noise std. σ_{b_f}	—	$1.2 \times 10^{-5}\,[\mathrm{m\,s^{-2}\,Hz^{1/2}}]$
Gyroscope		
Angular Random Walk σ_ω	$0.15\,[^\circ\,\mathrm{h^{-1/2}}]$	$4.4 \times 10^{-5}\,[\mathrm{rad\,s^{-1}\,Hz^{-1/2}}]$
Bias Instability b_{ins_ω}	$0.50\,[^\circ\,\mathrm{h^{-1}}]$	$2.4 \times 10^{-6}\,[\mathrm{rad\,s^{-1}}]$
Time constant T_ω	—	$3.6 \times 10^{2}\,[\mathrm{s}]$
Bias noise std. σ_{b_ω}	—	$5.7 \times 10^{-8}\,[\mathrm{rad\,s^{-1}\,Hz^{1/2}}]$

where $\boldsymbol{R}_z$ is the measurement covariance. To generate such measurements, the full state vector $\boldsymbol{x}$ is again interpolated and resampled at T_{sens}, then the function $\boldsymbol{h}$ is applied to the every step in the whole resampeled trajectory before noise is applied. For the simulation experiments, PARS and GNSS were generated at 5Hz. The noise added to the GNSS measurements had a standard deviation of 1m in all axes. For PARS measurements the noise on elevation and azimuth had a standard deviation of 0.5°.

Chapter 4

Estimator Design

This chapter detail the design of the two estimators. Firstly a PARS sensor model is derived with accompanying Jacobians then the ESKF implementation is briefly introduced. Finally, more detail will be given in regard to the design of the FGO approach as well as some initial testing.

4.1 PARS measurement model

In both the ESKF and factor graph implementation a measurement model that predicts the measurements $\boldsymbol{z}$ from the system state $\boldsymbol{x}$, as well as its Jacobian, is needed. The phased-array radio systems that are considered in this thesis provide elevation α and azimuth ψ angle measurements of a vehicle with position $\boldsymbol{p}_{nb}^{n}$ as shown in Figure 4.1. Assuming the radio frame is aligned with NED, the measurement model is simply the

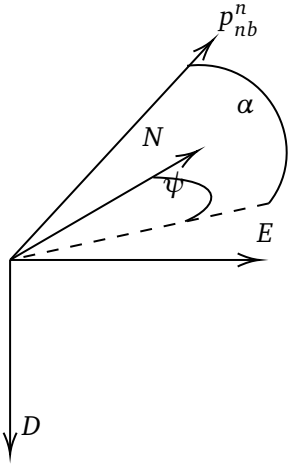


Figure 4.1: PARS measurements are given in spherical coordinates where α denote the elevation angle and ψ is the azimuth.

cartesian to spherical coordinate transform denoted with a bold $\boldsymbol{\alpha}$

$$\begin{bmatrix} \alpha^n \\ \psi^n \end{bmatrix} = \boldsymbol{\alpha}\left(\boldsymbol{p}^n_{nb}\right) = \begin{bmatrix} \operatorname{atan2}\left(\boldsymbol{p}^n_{nb,z}, \sqrt{(\boldsymbol{p}^n_{nb,x})^2 + (\boldsymbol{p}^n_{nb,y})^2}\right) \\ \operatorname{atan2}\left(\boldsymbol{p}^n_{nb,y}, \boldsymbol{p}^n_{nb,x}\right) \end{bmatrix} \tag{4.1}$$

The Jacobian of (4.1) with respect to the position is

$$\boldsymbol{J}^{\boldsymbol{\alpha}(\boldsymbol{p}^n_{nb})}_{\boldsymbol{p}^n_{nb}} = \begin{bmatrix} \frac{-\boldsymbol{p}^n_{nb,x}\boldsymbol{p}^n_{nb,y}}{r^2\sqrt{(\boldsymbol{p}^n_{nb,x})^2+(\boldsymbol{p}^n_{nb,y})^2}} & \frac{-\boldsymbol{p}^n_{nb,y}\boldsymbol{p}^n_{nb,z}}{r^2\sqrt{(\boldsymbol{p}^n_{nb,x})^2+(\boldsymbol{p}^n_{nb,y})^2}} & \frac{\sqrt{(\boldsymbol{p}^n_{nb,x})^2+(\boldsymbol{p}^n_{nb,y})^2}}{r^2} \\ \frac{-\boldsymbol{p}^n_{nb,y}}{(\boldsymbol{p}^n_{nb,x})^2+(\boldsymbol{p}^n_{nb,y})^2} & \frac{\boldsymbol{p}^n_{nb,x}}{(\boldsymbol{p}^n_{nb,x})^2+(\boldsymbol{p}^n_{nb,y})^2} & 0 \end{bmatrix} \tag{4.2}$$

where $r = \sqrt{(\boldsymbol{p}^n_{nb,x})^2 + (\boldsymbol{p}^n_{nb,y})^2 + (\boldsymbol{p}^n_{nb,z})^2}$ is the range. The measurements are in the radio frame $\{r\}$. However, it cannot be assumed that the PARS frame is aligned with NED in general so the measurement function and its Jacobian has to be corrected for this. That is, the measurements need to be converted from radio frame $\{r\}$ to the local navigation frame $\{n\}$ before they can be used in the filters. Assuming that the PARS pose $\boldsymbol{T}^r_n$ is known, the measurement model is then

$$\begin{bmatrix} \alpha^r \\ \psi^r \end{bmatrix} = \boldsymbol{h}(\boldsymbol{p}^n_{nb}) = \boldsymbol{\alpha}\left(\boldsymbol{T}^r_n \cdot \boldsymbol{p}^n_{nb}\right) = \boldsymbol{\alpha}\left(\boldsymbol{p}^r_{rb}\right) \tag{4.3}$$

The measurement model Jacobian is

$$\boldsymbol{H}(\boldsymbol{p}^n_{nb}) = \boldsymbol{J}^{\boldsymbol{\alpha}(\boldsymbol{T}^r_n \cdot \boldsymbol{p}^n_{nb})}_{\boldsymbol{p}^n_{nb}} \boldsymbol{R}^r_n \tag{4.4}$$

This is the Jacobian used in the ESKF implementation.

To create a custom factor in `GTSAM` the measurement Jacobian is needed. The intuitive approach might be to try to use the Jacobian in (4.4), however this does not work as the Jacobian with respect to the whole manifold $\boldsymbol{T}^n_b$ is needed when using `GTSAM`. To do this it is first noted that the position vector can be extracted from the matrix representation in the same way as done in (2.84). This operation will be denoted as a function $s(\cdot)$

$$\boldsymbol{p}^r_{rb} = s(\boldsymbol{T}^r_b) \tag{4.5}$$

which has the Jacobian as in (2.85). The Jacobian of rigid action with respect to the left term, as well as the Jacobian of the composition with respect to the right term is found in Subsection 2.2.5. The latter is just identity. The Jacobian of (4.5) with respect to the vehicle pose $\boldsymbol{T}^b_n$ is then found using the chain rule

$$\begin{aligned} \boldsymbol{J}^{\boldsymbol{h}}_{\boldsymbol{T}^n_b} &= \boldsymbol{J}^{\boldsymbol{\alpha}(\boldsymbol{p}^r_{rb})}_{\boldsymbol{T}^n_b} \\ &= \boldsymbol{J}^{\boldsymbol{\alpha}\left(s\left(\boldsymbol{T}^r_n \boldsymbol{T}^n_b\right)\right)}_{s\left(\boldsymbol{T}^r_n \boldsymbol{T}^n_b\right)} \boldsymbol{J}^{s\left(\boldsymbol{T}^r_n \boldsymbol{T}^n_b\right)}_{\boldsymbol{T}^r_n \boldsymbol{T}^n_b} \boldsymbol{J}^{\boldsymbol{T}^r_n \boldsymbol{T}^n_b}_{\boldsymbol{T}^n_b} \\ &= \boldsymbol{J}^{\boldsymbol{\alpha}(\boldsymbol{p}^r_{rb})}_{\boldsymbol{p}^r_{rb}} \begin{bmatrix} \boldsymbol{R}^r_b & \boldsymbol{0} \end{bmatrix} \end{aligned} \tag{4.6}$$

This is the same on SE(2) and on SE(3) only the dimensions change. Of course, for SE(2) the function $\boldsymbol{\alpha}$ has to be modified to only give azimuth.

In the case that the PARS pose $\boldsymbol{T}_r^n$ is not known accurately it could be beneficial to include it as a variable in the factor graph. To do this the following Jacobian is needed

$$\begin{aligned}
\boldsymbol{J}_{\boldsymbol{T}_r^n}^{\boldsymbol{h}} &= \boldsymbol{J}_{\boldsymbol{T}_r^n}^{\boldsymbol{\alpha}(\boldsymbol{p}_{rb}^r)} \\
&= \boldsymbol{J}_{\boldsymbol{T}_n^r \cdot \boldsymbol{p}_{nb}^n}^{\boldsymbol{\alpha}(\boldsymbol{T}_n^r \cdot \boldsymbol{p}_{nb}^n)} \boldsymbol{J}_{\boldsymbol{T}_n^r}^{\boldsymbol{T}_n^r \cdot \boldsymbol{p}_{nb}^n} \boldsymbol{J}_{\boldsymbol{T}_r^n}^{(\boldsymbol{T}_r^n)^{-1}} \\
&= \boldsymbol{J}_{\boldsymbol{p}_{rb}^r}^{\boldsymbol{\alpha}(\boldsymbol{p}_{rb}^r)} \begin{bmatrix} \boldsymbol{R}_n^r & -\boldsymbol{R}_n^r \left[\boldsymbol{p}_{nb}^n\right]_\times \end{bmatrix} \left(-\boldsymbol{Ad}_{\boldsymbol{T}_r^n}\right) \\
&= \boldsymbol{J}_{\boldsymbol{p}_{rb}^r}^{\boldsymbol{\alpha}(\boldsymbol{p}_{rb}^r)} \begin{bmatrix} \boldsymbol{R}_n^r & -\boldsymbol{R}_n^r \left[\boldsymbol{p}_{nb}^n\right]_\times \end{bmatrix} \begin{bmatrix} -\boldsymbol{R}_r^n & -\left[\boldsymbol{p}_{nr}^n\right]_\times \boldsymbol{R}_r^n \\ \boldsymbol{0}^\top & -\boldsymbol{R}_r^n \end{bmatrix} \\
&= \boldsymbol{J}_{\boldsymbol{p}_{rb}^r}^{\boldsymbol{\alpha}(\boldsymbol{p}_{rb}^r)} \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{R}_n^r \left[\boldsymbol{p}_{nb}^n - \boldsymbol{p}_{nr}^n\right]_\times \boldsymbol{R}_n^r \end{bmatrix} \\
&= \boldsymbol{J}_{\boldsymbol{p}_{rb}^r}^{\boldsymbol{\alpha}(\boldsymbol{p}_{rb}^r)} \begin{bmatrix} -\boldsymbol{I} & \left[\boldsymbol{R}_n^r \boldsymbol{p}_{rb}^n\right]_\times \end{bmatrix} \\
&= \boldsymbol{J}_{\boldsymbol{p}_{rb}^r}^{\boldsymbol{\alpha}(\boldsymbol{p}_{rb}^r)} \begin{bmatrix} -\boldsymbol{I} & \left[\boldsymbol{p}_{rb}^r\right]_\times \end{bmatrix}
\end{aligned} \tag{4.7}$$

where the identity

$$\boldsymbol{R}\left[\boldsymbol{p}\right]_\times \boldsymbol{R}^\top = \left[\boldsymbol{R}\boldsymbol{p}\right]_\times \tag{4.8}$$

was used. Also, the Jacobian of the action with respect to the pose, as well as the Jacobian with respect to the inverse was used. With these Jacobians, it enables the creation of a custom PARS factor in `GTSAM`. The Jacobian calculations are error-prone and so checking them numerically was deemed necessary. To do this, a snippet similar to the one found in Code listing 4.1 was used. The main difference between this and standard finite differences is that the exponential map has to be used, instead of ordinary addition, when perturbing the function argument.

Code listing 4.1: Function of SE(3) finite differences Jacobian. The output of $\boldsymbol{h}$ is expected to be a vector.

```
import numpy as np
import gtsam
from typing import Callable

def se3_finite_differences(X: gtsam.Pose3,
                           h: Callable,
                           z_dim: int):
    x_dim = 6
    d = 1e-7
    J = np.zeros((z_dim, x_dim))
    for i in range(x_dim):
        t = np.zeros(x_dim)
        t[i] = 1e-7
        J[:,i] = (h(X*X.Expmap(t)) - h(X)) / d
    return J
```

4.2 Error state Kalman filter implementation

The goal of this thesis is mainly to explore factor graph based approaches for navigation using PARS, so this section will be brief. To establish a baseline in performance, an error state Kalman filter was developed. The ESKF implementation is inspired by previous work [41], and provided materials from the sensor fusion course TTK4250 [42]. The relevant theoretical background regarding the ESKF is found in Subsection 2.5.2. The main point to note, that is not mentioned in the theory section is that the system matrices need to be discretized. This is done using Van Loan's formula as described in [26]. This is then used to propagate the error state covariance. The prediction of the nominal state is done using strapdown equations in (2.30). Other than that, the equations in Subsection 2.5.2 are implemented more or less directly as stated. The prediction function takes in an `IMU` object which contains the noise standard deviations discussed in Subsection 2.3.3. The update function takes in a `Sensor` object that needs to contain the measurement prediction function $\boldsymbol{h}(\boldsymbol{x})$ as well as its Jacobian $\boldsymbol{H}(\boldsymbol{x})$. Given a list of IMU measurements `z_acc` and `z_gyr`, sensor measurements `z_sens` from e.g. the simulator and some other basic parameters, the ESKF can be run as in Code listing 4.2.

Code listing 4.2: An example of using the ESKF prediction and update function on simulated data.

```
import numpy as np
from parnav.eskf import predict, update

def run_eskf(N, x0, cov0, imu, z_acc, z_gyr, imu_time, imu_dt,
             sensor, z_sensor, sensor_time, gravity):

    x = np.zeros((16,N))
    x[:,0] = x0

    P_err = np.zeros((15,15, N))
    P_err[:,:,0] = cov0

    last_update_indx_sens  = 0
    for k in range(N-1):
        # predict with IMU
        dt = imu_dt if k == 0 else imu_time[k] - imu_time[k-1]
        x[:, k+1], P_err[:,:,k+1] = predict(
            x[:,k], P_err[:,:,k], z_acc[:,k], z_gyr[:,k], dt, gravity, imu
            )

        # update with sensor
        if last_update_indx_sens < len(sensor_time):
            if imu_time[k+1] >= sensor_time[last_update_indx_sens]:
                x[:, k+1], P_err[:,:,k+1] = update(
                    x[:,k+1], P_err[:,:,k+1],
                    z_sensor[:, last_update_indx_sens], sensor)
                last_update_indx_sens += 1

    return x, P_err
```

4.3 Factor graph filter implementation

4.3.1 GTSAM

There are multiple open source factor graph framework freely available for use, like Symforce [33], `WOLF` [34], `g2o` [43] and `GTSAM` [32]. These have different advantages and disadvantages. In the end, `GTSAM` was chosen for this project because of its large community and wide use over time which has made it well-tested and robust Futher, as well as its large collection of factors including the IMU factor discussed in Subsection 2.6.2. `GTSAM` is an abbreviation for Georgia Tech Smoothing and Mapping and is a framework written in `C++` that allows for factor graph optimization [44]. It has mainly been used to solve SLAM and structure from motion (SFM) problems, but it can also be used in navigation problems by using for instance the GPS-factor. Another great attribute of `GTSAM` is that the developers have created `Python` and `MATLAB` wrappers for much of the functionally contained within the library. In this work, the `Python` wrappers have been used.

Code listing 4.3: The custom factor function in the PARS sensor object.

```
# ...
# custom factor error function inside PARS sensor class
def J_h_X_man(self, X: gtsam.Pose3):
    R_rn = self.P.rotation().matrix().T
    R_nb = X.rotation().matrix()
    p_rb_r = self.P.transformTo(X.translation())
    J = np.zeros((2,6))
    J[:,3:] = self.J_alpha_p_rb_r(p_rb_r)@(R_rn@R_nb)
    return J

def cf_error_par(self,
                 z: np.ndarray,
                 this: gtsam.CustomFactor,
                 values: gtsam.Values,
                 jacobs: Optional[List[np.ndarray]]):
    X = values.atPose3(this.keys()[0]) #T_nb
    p_nb_n = X.translation()
    error = np.array([ssa(self.h(p_nb_n)[0] - z[0]),
                      ssa(self.h(p_nb_n)[1] - z[1])])
    if jacobs is not None:
        jacobs[0] = self.J_h_X_man(X)
    return error
# ...


# ...
# useage inside factor graph creation
pars_factor = gtsam.CustomFactor(pars_noise_model,
                                 [X(i)],
                                 partial(pars.cf_error_pars, z_pars[:,z_idx]))
factor_graph.push_back(pars_factor)
# ...
```

4.3.2 PARS factor

To create a custom factor in `GTSAM` one needs to define an error function and the measurement prediction Jacobian. The error function is usually just $\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x})$ if $\boldsymbol{z}$, but for bearing measurements it is slightly different, as the difference needs to be wrapped around a shortest signed distance (SSA) function such that the difference in angles close to $\pm 180°$ is handled properly. Sensor object that have defined a measurement model can then create a custom factor error function such as the PARS one displayed in Code listing 4.3. This was created following the guide and examples in `Python` folder of [32]. The custom factor in Code listing 4.3 is only on the SE(3) pose variable and the Jacobian `J_h_X_man` is the one defined in (4.6). The measurement model $\boldsymbol{h}$ is a bit different from the one defined in (4.5) as it takes in the position of the vehicle in NED, and rotates it to the radio frame using the pose stored internally in the PARS sensor object. In later sections, the online calibration of the PARS pose is also added by simply extending the `jacobian` list with (4.7) as well as adding the PARS pose to the `values` list.

4.3.3 PARS factor initial testing on SE2

Once the PARS factor was created, some initial testing was done by using a small factor graph with only three variables $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3 \in \mathrm{SE}(2)$. Firstly three factors were placed on the variables. A `BetweenFactorPose2` was added on $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as well as one on $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ with an odometry vector $\boldsymbol{\tau}$ and noise model $\boldsymbol{\sigma}_{\boldsymbol{\tau}}$ as

$$\boldsymbol{\tau} = \begin{bmatrix} 4 & 0 & 0 \end{bmatrix}^\top$$

$$\boldsymbol{\sigma}_{\boldsymbol{\tau}} = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

where the first two terms correspond to translation and the last to the rotation. Additionally, a `PriorFactorPose2` was added on $\boldsymbol{x}_1$ with zero translation and zero rotation. The noise model on the prior was

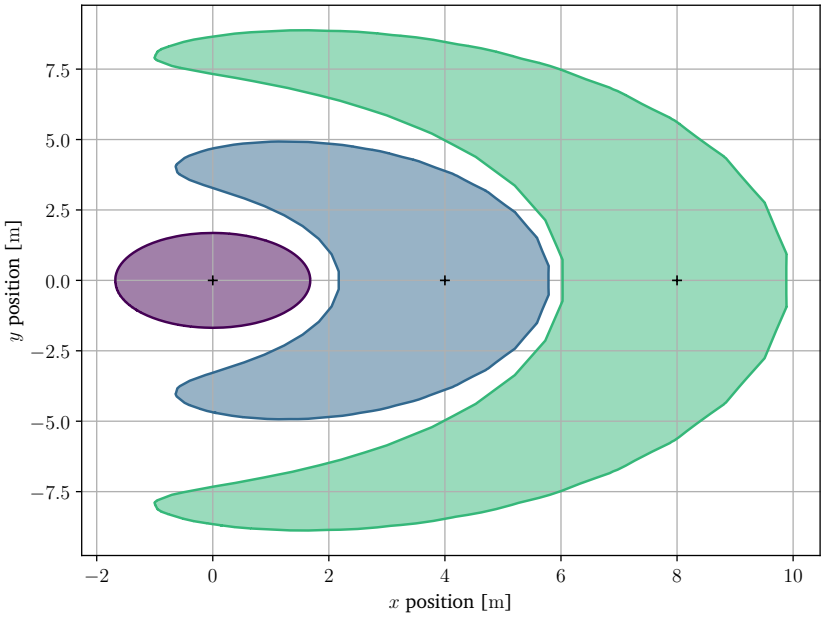
$$\boldsymbol{\sigma}_{\boldsymbol{x}_1} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$

The initial value for the variables was given as ground truth plus a small perturbation. Then the factor graph was solved using the `LevenbergMarquardtOptimizer`. See e.g. [31] for details. The marginals of the translation part of the $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ are shown in Figure 4.2a. The "banana" shape comes from the fact that the covariance is defined Gaussian in tangent space; when this is projected onto SE(2), the following shapes of the covariance are observed.

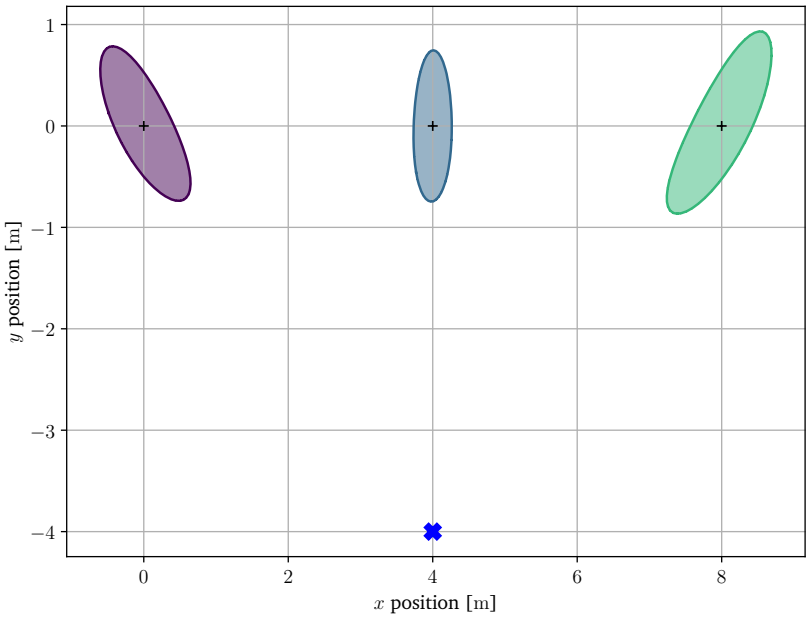
After testing with only prior and odomtery between factors, a PARS factor was added on each variable. The three factors were created with the ground truth angles as measurements. The noise model used on each PARS factor was

$$\boldsymbol{\sigma}_z = 0.02$$

Then the factor graph was again solved for the marginals using `LevenbergMarquardt`, which gave the result in Figure 4.2b. When adding the PARS factors, the covariance is smaller and is skewed towards the PARS position, which is as expected; as there are no range measurements, the factor constrains the position along a straight line from the PARS position to the vehicle.



(a)



(b)

Figure 4.2: Factor graph on SE(2). Top figure: Marginal variables after optimization without PARS factor. Bottom figure: Marginal variables after graph optimization with added PARS factor on the poses. The blue '×' at(4,-4) is the position of the PARS. For references on how these figures were made see Section 4.4.

4.3.4 Factor graph

Similar to the ESKF implementation, a factor graph optimization class was implemented that can operate on IMU measurements as well as sensors with a corresponding error function and Jacobian. The factor graphs consists of three types of variables: pose $\boldsymbol{T}_{b,i}^{n} \in \mathrm{SE}(3)$, velocity $\boldsymbol{v}_{nb,i}^{n} \in \mathbb{R}^3$ and biases $\boldsymbol{b}_i \in \mathbb{R}^6$. Notice that the bias for the accelerometer and gyroscope is now combined in one vector. The first factors added to the graph are the prior factors `PriorFactorPose3` on $\boldsymbol{T}_{b,0}^{n}$, `PriorFactorVector` on $\boldsymbol{v}_{nb,0}^{n}$ and `PriorFactorConstantBias` on $\boldsymbol{b}_0$. Other factors that are added to the graph are IMU-factors, bias factors and measurement factors such as GPS-factors or PARS-factors as detailed in Subsection 4.3.2. These are added simultaneously on a new measurement's arrival.

Preintegrated IMU measurements

To handle the IMU measurements the `PreintegratedImuMeasurements` (PIM) class in `GTSAM` based on the IMU on manifold pre-integration theory mentioned in Subsection 2.6.2 was used. The PIM object is initialized with the continuous-time IMU noise covariance $\sigma_f^2 \boldsymbol{I}$ and $\sigma_\omega^2 \boldsymbol{I}$ using the functions `setGyroscopeCovariance` and `setAccelerometerCovariance`. The IMU covariances were based on the Table 3.1, but were subject to some tuning, which will be detailed later. There is also a parameter to the PIM object called the `integrationCovariance`, which is supposed to model approximation errors in the integration of the IMU measurements. Some experimentation showed that if this was set lower than `10e-10`, ill-posed errors appeared. When a new measurement from the sensor arrived, the PIM object is used to create an `IMUFactor` on $\boldsymbol{T}_{b,i-1}^{n}$, $\boldsymbol{v}_{nb,i-1}^{n}$, $\boldsymbol{b}_{i-1}$ and $\boldsymbol{T}_{b,i}^{n}$, $\boldsymbol{v}_{nb,i}^{n}$, which is inserted into the graph along with the corresponding PARS factor on $\boldsymbol{T}_{b,i}^{n}$, and then the integration is reset. Prior to this, the optimizer needs an initial estimate of each variable added to the graph, and to get this for the pose and velocity variables the `predict` function in PIM is used. This predicts those states from the previous states and the integrated IMU measurements. This is done for each new IMU measurement, such that the predicted state is available between updates. This is done inside the `predict` function of the FGO class, which is essentially just a wrapper around PIM `predict`. A final thing to note is that `GTSAM` comes by default with a new version preintegration called tangent preintegration. For reasons that are unclear, after some testing with this, it was found to be unreliable, yielding large errors that were not present using the strapdown equations. Therefore the tangen preintegration was turned off, and `GTSAM` was rebuilt with the original preintegration. This yielded performance commensurate with the strap-down equations.

Bias factor

The PIM object used here does not handle the biases. There exists a similar class called the `PreintegratedCombinedMeasurements` that takes care of the biases. However, there seems to be an issue in the `Python` wrappers of this functionality, or some unknown-to-the-author configuration is needed. In any case, experiments with the `Preintegrat-`

edCombinedMeasurements was not successful, and so the biases were modeled using the BetweenFactorConstantBias factor on $\boldsymbol{b}_i$ instead. For PIM to take the estimated bias into account the function bias inside the object is set when the integration is set. The difficult part of getting this right is that in contrast to the PIM object that takes the continuous time IMU noise sigmas, the bias noise in the between factor needs to be appropriately scaled when a new factor is inserted. In this implementation, a bias factor is inserted into the graph only when a new sensor measurement arrives, so the discrete-time bias noise standard deviation is given as

$$\boldsymbol{\sigma}_{b,d} = \sqrt{T_{ij}}\boldsymbol{\sigma}_b \tag{4.9}$$

where T_{ij} is the time difference between bias factor i and j. Every time an IMU measurement is added to PIM the time increment is also stored. In this sense, PIM can easily be used to keep track of the time since the last update and is therefore used to fetch T_{ij} on a new insert. The initial biases in the optimization are just set to be equal to the previous biases. To summarize: prior factors are placed on the first variables. Each time a new PARS measurement arrives, bias, IMU and PARS factors are added, and the graph is optimized. The structure of the factor graph is illustrated in Figure 4.3. Example usage with one sensor is shown in Code listing 4.4.

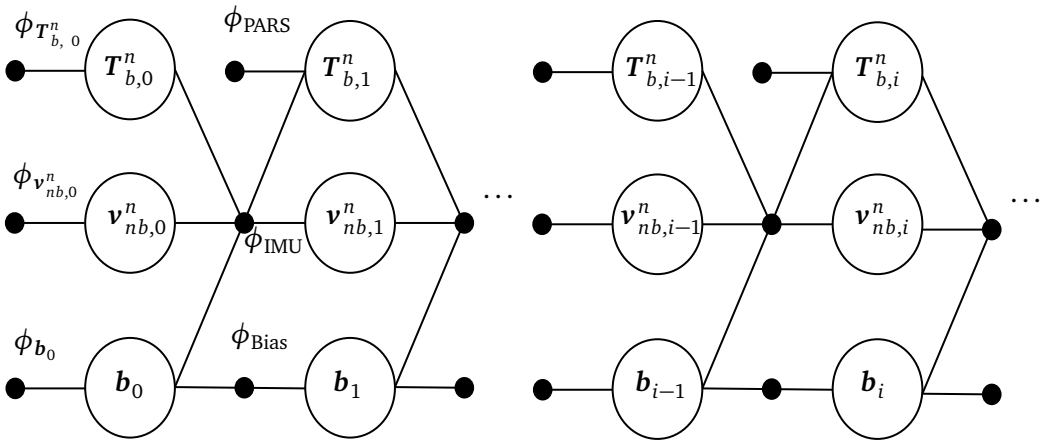


Figure 4.3: Factor graph. Here only the first factors are labeled.

4.3.5 Optimizer

The initial plan was to use the ISAM2 optimizer, however the runtime with this became infeasible for datasets longer than 10 - 15 minutes. Even with experimentation with parameters such as relinearizeThreshold and relinearizeSkip, which makes ISAM2 skip relinearization for a given number of updates, the graph became too large for sufficient performance. Instead, the IncrementalFixedLagSmoother from was used. As the name implies, this is a fixed lag smoother that only considers the variables within the fixed lag when performing the optimization which leads to constant computational complexity, regardless of the graph size. The variables that fall out of the lag are marginal-

ized out. In the end, the relinearization threshold was set to 0.001 and relinearize skip to 1, such that variables are likely relinearized on every update. Additionally, two extra update calls are performed, to increase the likelihood of convergence. The lag interval was subject to a lot of experimentation; a discussion regarding this parameter will be of concern later.

Code listing 4.4: Example usage of the factor graph class

```
from functools import partial
import numpy as np
import gtsam
from gtsam.symbol_shorthand import X
from parnav.fgo import FgoFixedLag, x_to_pvb, pvb_to_x

def run_fgo(N, x0, cov0, imu, z_acc, z_gyr, imu_time, imu_dt,
            sensor, z_sensor, sensor_time, sensor_dt):

    x = np.zeros((16, N))
    x[:10,0] = x0[:10]
    pose0, vel0, bias0 = x_to_pvb(x0)

    fgo = FgoFixedLag(x0, cov0, imu, sensor_dt, lag=30)
    sens_noise = gtsam.noiseModel.Diagonal.Sigmas(sensor.std)

    last_nav = gtsam.NavState(pose0, vel0)
    last_bias = bias0
    last_update_indx_pars  = 0
    for k in range(N-1):
        # predict with IMU
        dt = imu_dt if k == 0 else imu_time[k] - imu_time[k-1]
        pose, vel, bias = fgo.predict(
            z_acc[:,k], z_gyr[:,k], dt, last_nav, last_bias)
        x[:,k+1] = pvb_to_x(pose, vel, bias)

        # Update with sensor
        factors = []
        if last_update_indx_pars < len(sensor_time):
            if imu_time[k+1] >= sensor_time[last_update_indx_pars]:
                factors.append(gtsam.CustomFactor(sens_noise,
                    [X(fgo.update_indx+1)],
                    partial(sensor.cf_error_par, z_sensor[:,last_update_indx_pars])))
                z_time = sensor_time[last_update_indx_pars-1]
                last_update_indx_pars += 1

        if factors:
            pose, vel, bias = fgo.update(factors, pose, vel, bias, z_time)
            last_nav   = gtsam.NavState(pose, vel)
            last_bias  = bias
            x[:,k+1]   = pvb_to_x(pose, vel, bias)

    return x
```

4.4 References and other resources used

Other than `GTSAM [32]`, this work uses multiple `Python` [45] packages, including `Numpy` [46] and `Scipy` [47] for calculations and basic functionality. For coordinate frame transformations like ECEF to NED the package `PyMap3d` [48] was used. `Matplotlib` [49] is used to generate the plots. The SSA function is inspired by the `MATLAB` implementation in [50]. Furthermore, the example in Subsection 4.3.3 is inspired by [51]. To create the visualization in Figure 4.2, `plot_utils.py` in [51] was used. This also uses the library `Visgeom` [52]. To calculate the Jacobians in Section 4.1 material from the course

TTK21 Introduction to Visual Simultaneous Localization and Mapping - VSLAM was used, particularly the exercise sheet [53]. It would have been difficult to create the factor graph implementation without the inspiration and knowledge that was gathered from the various examples in the `Python` folder of [32]. The references used to make the custom factors were mainly the guide `CustomFactors.md` and the examples in `CustomFactorExample.py` in `GTSAM` [32]. Finally, the finite differences method was inspired by the implementation in [54].

Chapter 5

Simulation Results and Discussion

Three simulation experiments were run to verify the filters before testing on data from flight test. Firstly they were run using GNSS measurements to establish a baseline in performance. Two PARS simulation experiments are also showed here; firstly a run using only one PARS ground system positioned at the origin, and then a run with three PARS ground systems distributed evenly along the flight path. The experiment utilizing just one PARS was done to see if it is possible to navigate while getting measurements that constrain the vehicle to a line. The three PARS experiments contain enough information to constrain the vehicle's position to a point in space and should be more accurate. The flight path used in the simulations is taken from a dataset [40] used in TTK4250 Sensor Fusion, and contains strong dynamics that should benefit the estimators. The subsequent subsections will describe and discuss all three experiments in more detail. However, some assumptions made in all experiments should be mentioned. They are:

- The PARS poses are assumed to be known with perfect accuracy.
- The measurements contain no outliers.
- The measurements are affected by white noise.

Firstly, the assumption that all the PARS ground poses are assumed to be known with perfect accuracy is quite strong. In reality, this would be difficult to achieve without some initial surveying and calibration. It could be possible to survey the position with GNSS and the orientation calibration using a compass and a level. However, as will be a topic in later sections, this is hard to get right and calibration is a problem that needs to be addressed. Secondly, the PARS and GNSS measurements contain to outliers; this might be realistic for a UAV using GNSS in some cases, but as seen, the PARS measurements contain a considerable amount of outliers caused by multipath, especially in the elevation angle. As a result of this assumption, the estimators in these experiments are run without outlier rejection active. Outlier rejection as well will be a topic later. Lastly, the assumption that the noise affecting the PARS and GNSS measurements is assumed white might be remedied by increasing the noise to a point at which it can capture other effects as discussed in Section 2.4.

The parameters used to simulate the measurements and tune the estimators are found in Table 5.1. For the estimators, the IMU measurement noise and bias noise stan-

Table 5.1: Simulator and estimator tuning paramaters

Type	Value	Unit	Type	Value	Unit
GNSS sim std.	1	[m]	GNSS std.	2	[m]
PARS sim elev std.	0.5	[°]	PARS Elev std.	1	[°]
PARS sim azi std.	0.5	[°]	PARS Azi std.	1	[°]
IMU sim	Table 3.1		Acc. / Acc. b scaling	10 / 50	—
IMU frequency	200	[Hz]	Gyr. / Gyr. b scaling	10 / 10	—
GNSS frequency	5	[Hz]	$p^n_{nb,0}$ std.	5	[m]
PARS frequency	5	[Hz]	$v^n_{nb,0}$ std.	2	$[\mathrm{m\,s^{-1}}]$
$p^n_{nb,0}$ perturbation	±2	[m]	$q^n_{b,0}$ std.	10	[°]
$q^n_{b,0}$ perturbation	±5	[°]	$b_{f,0}$ std.	0.1	$[\mathrm{m\,s^{-2}}]$
Smoother fixed lag	15	[s]	$b_{\omega,0}$ std.	0.002	$[\mathrm{rad\,s^{-1}}]$

Table 5.2: Simulation results RMSE relative the ground truth. Comparing errors from ESKF and FGO on three experiments.

	GNSS		1 PARS radio		3 PARS radios		
	ESFK	FGO	ESFK	FGO	ESKF	FGO	Unit
Position	0.28	0.29	34.2	35.1	3.18	2.89	[m]
Velocity	0.086	0.089	0.73	0.70	0.36	0.21	$[\mathrm{m\,s^{-1}}]$
Attitude	0.35	0.35	0.81	0.35	0.79	0.33	[°]
Gyro. Bias	0.012	0.012	0.043	0.015	0.039	0.13	$[^\circ\,\mathrm{s^{-1}}]$
Accel. Bias	0.011	0.011	0.035	0.0087	0.034	0.0086	$[\mathrm{m\,s^{-2}}]$

dard deviations are the ones used in the simulation but scaled by a factor as shown in the table. It is important to note that the tuning and initial values are the same for the two estimators. The IMU tuning was determined by trial and error; a higher scaling was needed in the accelerometer bias noise to make it track the ground truth bias as it drifts. The FGO window length was set to 15s. The lag was determined by trial and error; interestingly, a higher fixed lag resulted in worse performance, and so did a shorter lag. A final thing to note regarding the parameters of these experiments is that the initial position and orientation were perturbed from ground truth, to test the estimators' convergence properties.

The RMSE between all states and ground truth is found in Table 5.2. More in-depth results from the three simulations now follow consecutively in Sections 5.1 to 5.3. Plots of all states with ground truth as well as error plots are found in Appendix A.

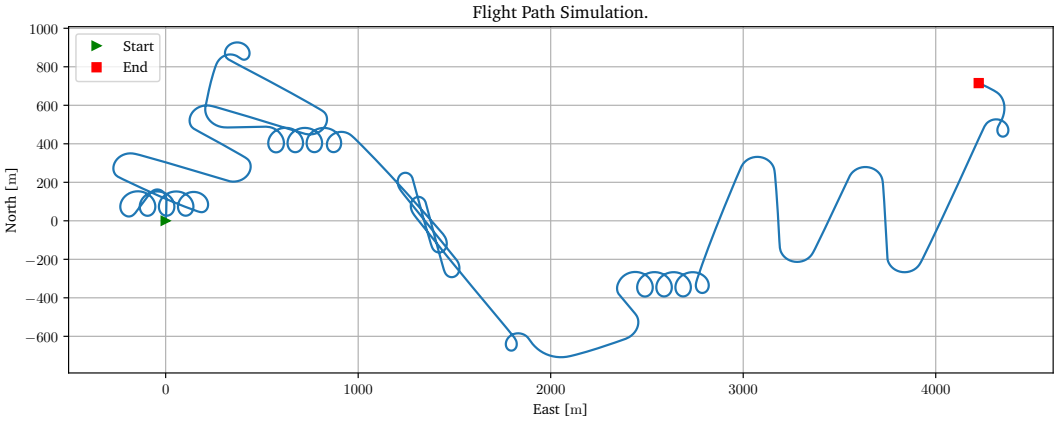


Figure 5.1: Simulation flight path

5.1 GNSS-aided INS

The first experiment carried out was with GNSS, this was done to establish a baseline to compare the performance of the PARS measurements. The flight path taken is pictured in Figure 5.1. The flight path contains quite strong accelerations which helps the biases converge. Error plots for position, attitude, and biases are displayed in Figure 5.2, the rest can be found in Appendix A. From Figure 5.2 it is clear that the position error is commensurate with the GNSS noise of 1m standard deviation, with RMSE of about 0.3m it seems like both estimators work as expected. Further, the estimators are able to quickly recover from the initialization errors in position and attitude. Interestingly, both estimators exhibit very similar error plots, which is a bit unexpected as the methods differ in such a large fashion. One might expect that the FGO smoother would indeed smooth out the errors more than what is the case. For the FGO estimator, different fixed lag intervals was experimented with, but a higher fixed lag did not decrease the RMSE or smooth out peak errors; for longer than 15s no additional performance gain was noted.

Another thing to note is that both estimators are able to converge on the correct yaw, even though no compass is used. Both the ESKF and FGO couple the error dynamics between the position and attitude and the estimation error via the cross covariances. Hence, this is as expected given the non-stationary flight path of Figure 5.1. See Appendix A, Figures A.1 to A.2 for details of the UAV motion.

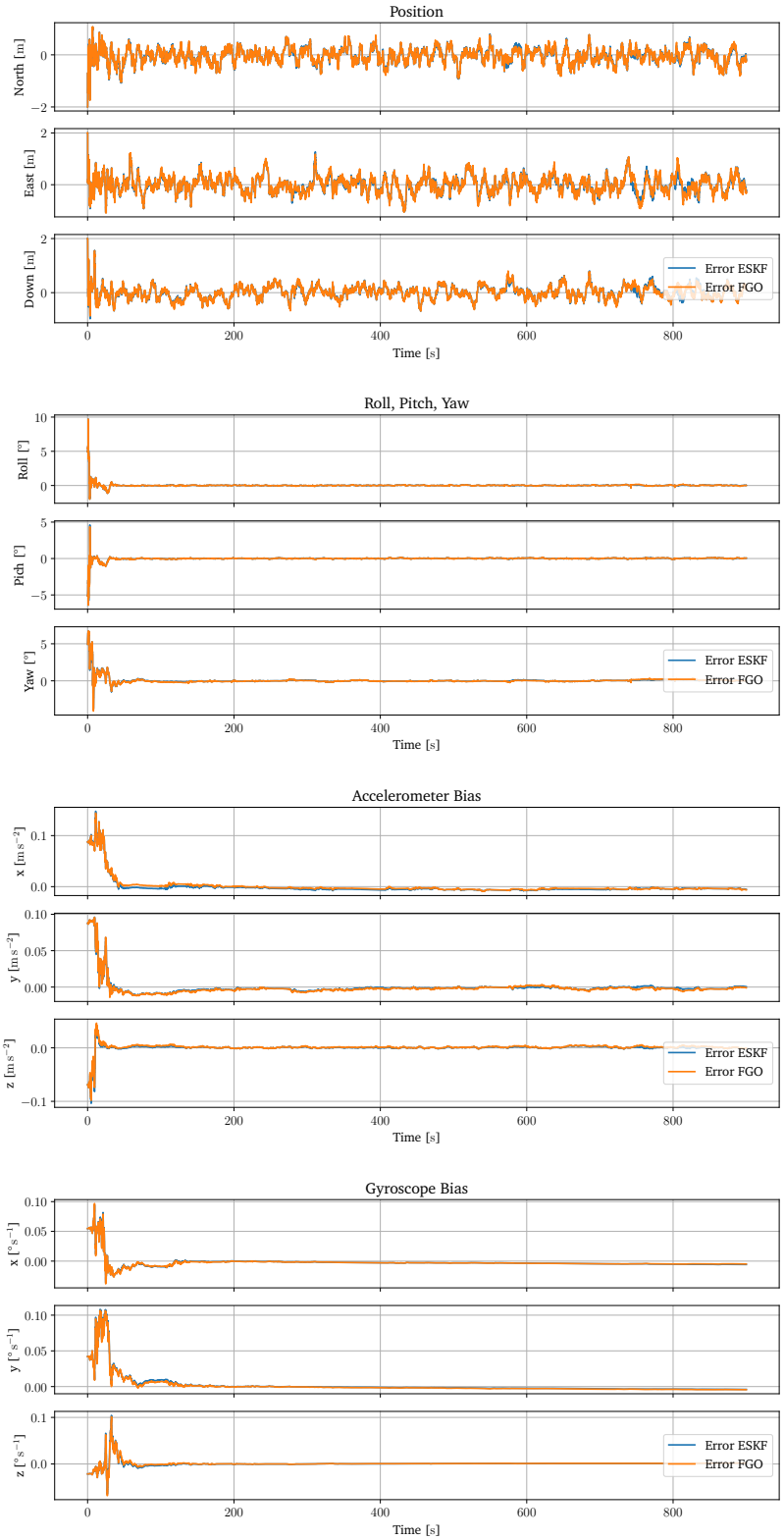


Figure 5.2: Simulation results with GNSS sensor. Error from ground truth. Top to bottom: position, attitude, accelerometer bias, gyroscope bias.

5.2 PARS-aided INS using one ground radio

The simulation was repeated but now instead of GNSS the PARS system was simulated. In this experiment, only one ground radio was used, and it was centered at the origin as shown in Figure 5.3. The noise added to the simulated elevation and azimuth measurements had a standard deviation of 0.5°. In Figure 5.5 the position, attitude, and biases converge appropriately as the vehicle is close to the radio. However, as the vehicle moves further away from the PARS, the position error starts to increase rapidly. When the vehicle is at a distance of about 3000m east of the radio the east position error reaches a maximum of over 200m for both the ESKF and FGO filters. This is expected as the measured angle differences become very small at such distances and even though the added noise of just 0.5°, it is enough to disrupt the position estimates. At 3000m a small change in position will result in a very small change in elevation and azimuth. However, the fact that the estimators are able to perform at all with just one PARS ground radio is unexpected; with one PARS the position of the vehicle is constrained to just a line extending from the PARS though the vehicle. One would need at least two PARS to constrain the position to a point. This becomes evident at 300s marked by an orange "×" in Figure 5.3, where the UAV is moving along the line extending through the PARS, which leads to a spike in the North position error as seen in Figure 5.5. Despite this, the performance of one ground PARS is considerably worse than for GNSS and with three ground PARS as seen in Table 5.2. The fact that it works at all is probably due to the high degree of dynamic excitation present in this motion profile which causes the biases to converge early such that the IMU integration is accurate. Had there been more stationary dynamics, the errors would probably be higher.

The ESKF have a slightly lower RMSE of 34.2m compared to FGO with 35.1m this is probably because of the erratic behavior experienced at the end when the vehicle is too far away from the PARS for accurate estimation, driving the RMSE up high. Calculating the position RMSE of the first 500s the ESKF RMSE is 6.52m and the FGO is 5.44m.

Another interesting thing to note is that the convergence of the states for the ESKF exhibits a longer transient time than for the FGO as seen in Figure 5.5. This seems to be because of the initial perturbation from ground truth in position. In Figure 5.4, the same run is performed only, but the initial position is set to ground truth. As seen, the biases for the two estimators have the same transient behavior in this case. The other states in Figure 5.5 also move more robustly towards their steady state, indicating that FGO is more robust to poor initial conditions.

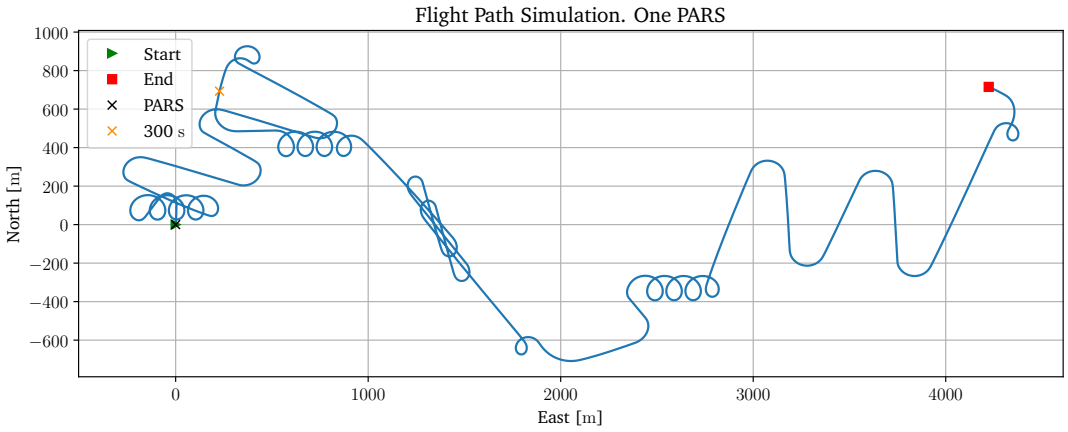


Figure 5.3: Simulation flight path with the PARS at the origin. At 300s, the UAV is moving straight toward the PARS, resulting in an error spike.

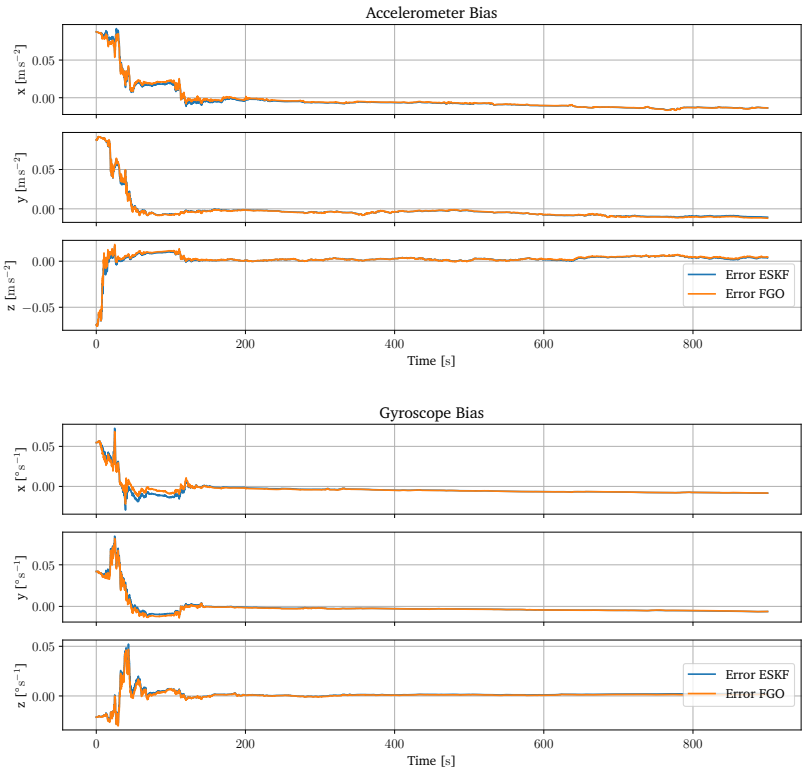


Figure 5.4: Simulation results one PARS. Bias errors with initial position set to ground truth. In this case the two estimators produce the same transient behavior, which is not the case when the initial position is perturbed.

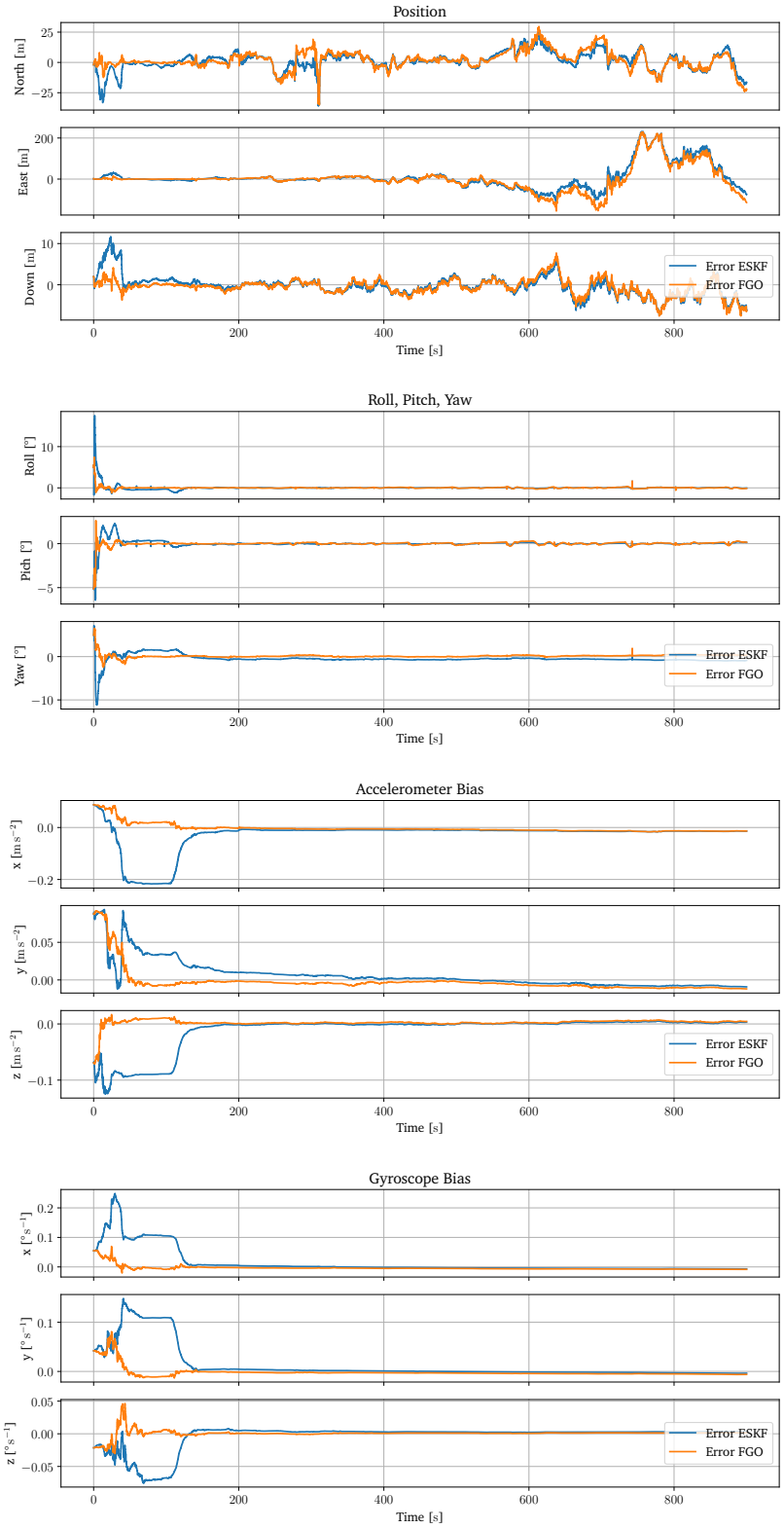


Figure 5.5: Simulation results with one PARS ground radio sensor. Error from ground truth. Top to bottom: position, attitude, accelerometer bias, gyroscope bias.

5.3 PARS-aided INS using three ground radios

The estimators were also tested with three PARS ground radios deployed around the flight path. This setup is depicted in Figure 5.6. The radios are placed at

$$\boldsymbol{p}^n_{nr_1} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top$$
$$\boldsymbol{p}^n_{nr_2} = \begin{bmatrix} -500 & 1500 & 0 \end{bmatrix}^\top$$
$$\boldsymbol{p}^n_{nr_3} = \begin{bmatrix} 0 & 4500 & 50 \end{bmatrix}^\top$$

and the orientation is perfectly aligned with NED. The filters seem to handle multiple measurements quite well, achieving better performance than with a single radio as expected. That is, an RMSE of 3.18m for the ESFK and 2.89m for the FGO was achieved. From Figure 5.7 the position error for both estimators reaches a maximum at about 20m in the east direction when the vehicle is the furthest away from one of the radios. It is worth mentioning that the noise added to the PARS measurements has a standard deviation of 0.5° which translates to large differences in position when the UAV is far from the PARS ground station. The attitude error is also substantially smaller for the FGO estimator than for the ESKF with errors of 0.33° and 0.79° respectively. The probable cause of this error difference is because the ESKF biases take longer to converge; namely about 150s while the FGO biases converge much faster. This again results in inaccurate integration of the IMU measurements hence the discrepancy is observed. As was the case with one PARS ground radio, FGO seems to be more robust to the initialization errors that were added.

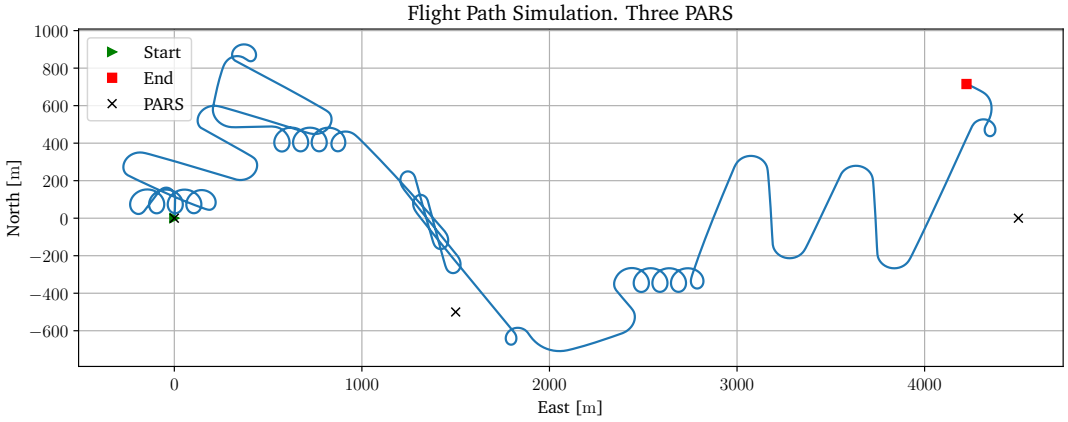


Figure 5.6: Simulation flight path with the three PARS ground radios. Flight path marked in blue. The start and end points are marked with an array and box, respectivly. The PARS ground systems are marked with ×-es.

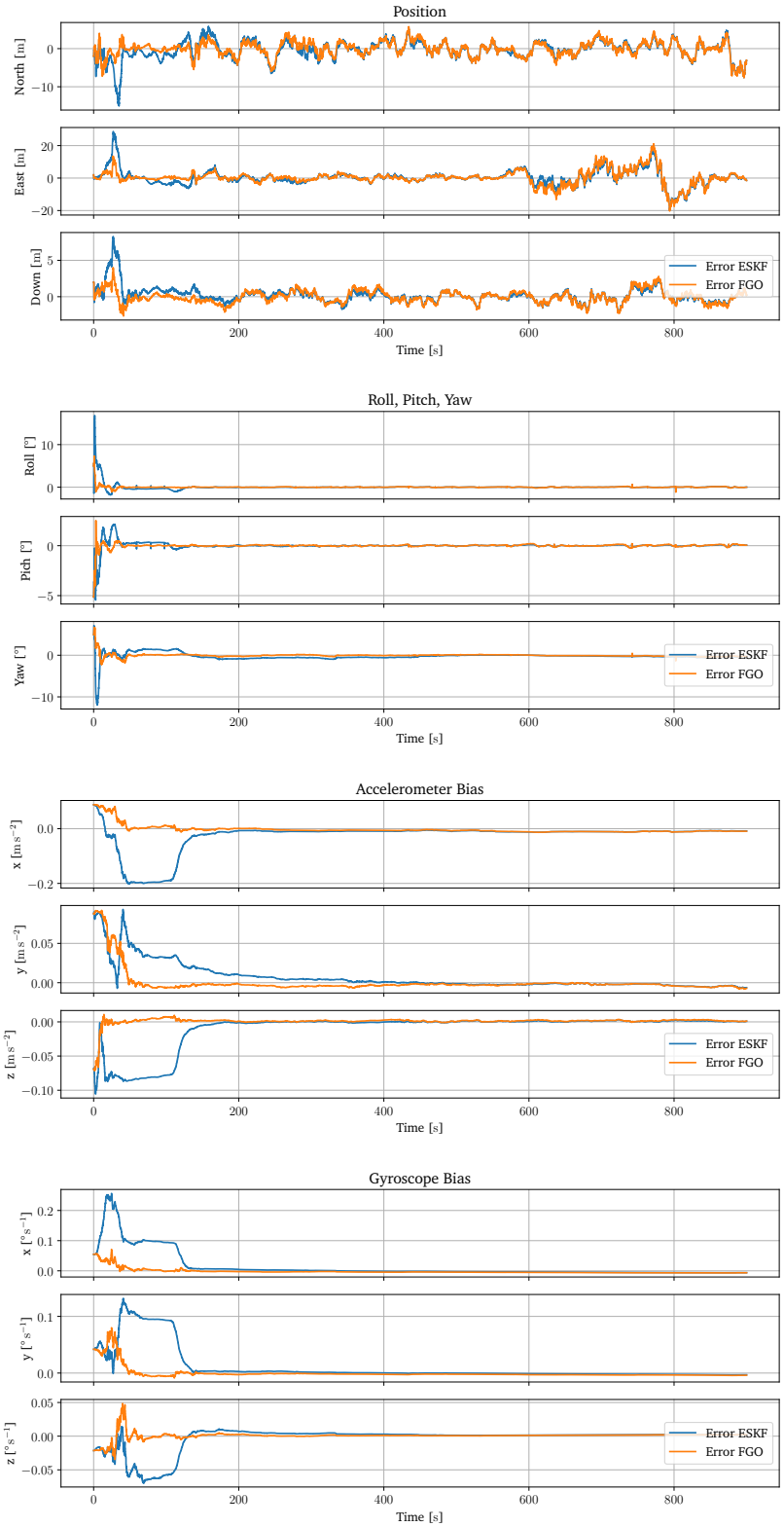


Figure 5.7: Simulation results with three PARS ground radios sensor. Error from ground truth. Top to bottom: position, attitude, accelerometer bias, gyroscope bias.

Chapter 6

Experimental Setup and Calibration

After initial development and experimentation in simulation, the filters were tested on experimental data to assure that the implementation would work in a real-world scenario. The data set used [55] was collected by researchers at NTNU UAV Lab in the fall of 2020 and has been used in [14, 16]. The data set has not been published but was made available for this project by the UAV Lab. [1] Three UAV flights were launched from Raustein and carried out over Trondheimsfjorden. Data from IMU, an AHRS, RTK-assisted GNSS as well as two PARS ground systems deployed at different locations were gathered. In this work the data from flight three is used. Moreover, some explanation about the hardware and software systems involved follows.

6.1 Hardware

The hardware used in the test flight is listed in Table 6.1. The UAV used is a Skywalker X8 fixed wing with a payload depicted in Figure 6.1. The payload consists of a Pixhawk, which runs the ArduPlane autopilot software. The Pixhawk has an internal IMU and a compass that is used by the autopilot software to form an attitude and heading reference system (AHRS) that was used to benchmark the attitude calculated by the estimators. A Sensonor STIM300 IMU is also part of the payload, which provides the IMU measurements that were used to test the algorithms. Some post-processing was done on these IMU measurements to downsample them to 200Hz. Additionally, the UAV carries a Ublox F9P-ZED receiver, capable of handling RTK GNSS measurements, and a CRE2-144-LW PARS by Radionor, used for communications. Two CRE2-189 PARS by Radionor were deployed on the ground and are pictured in Figure 6.2. In Figure 6.3, PARS 1 is depicted along with the ground computer, the UAV, and the launcher used to propel the aircraft initially. The detail of this hardware and software stack is taken from [16] and [14], which the reader is encouraged to read if further details are warranted.

[1] It is important to stress that the work done to gather this data set was not carried out as part of this project. The UAV Lab provided the dataset, for which this author is grateful.

Table 6.1: Hardware used in test flight.

Type	Name
UAV	Skywalker X8 fixed wing
Autopilot	Holybro Pixhawk 4
IMU	Sensonor STIM 300
GNSS Receiver	Ublox F9P-ZED
On-board PARS	CRE2-144-LW
Ground station PARS	Radionor CRE2-189

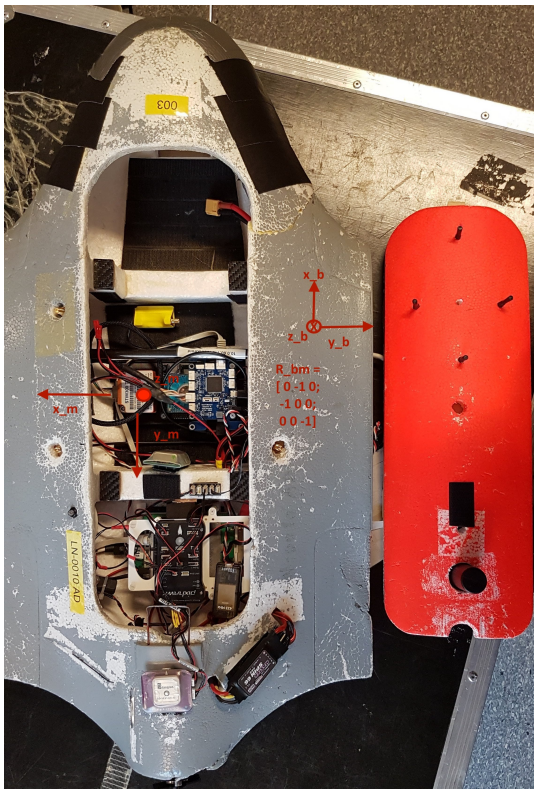


Figure 6.1: Payload. The STIM300 IMU is mounted as shown in relation to the body frame. Photo source: *NTNU UAV Lab* [55].

Figure 6.2: PARS 1 and 2 as well as RTK base stations mounted on top. Photo source: *NTNU UAV Lab* [55].

Figure 6.3: PARS, UAV and launcher. Photo source: *NTNU UAV Lab* [55]

6.2 Flight

Of the three flights carried out during the field experiment, flight three was selected to test the navigation algorithms. PARS 1 is the radio closest to the takeoff and landing spot and PARS 2 is the radio a couple of kilometers down the road as seen in Figure 6.4.

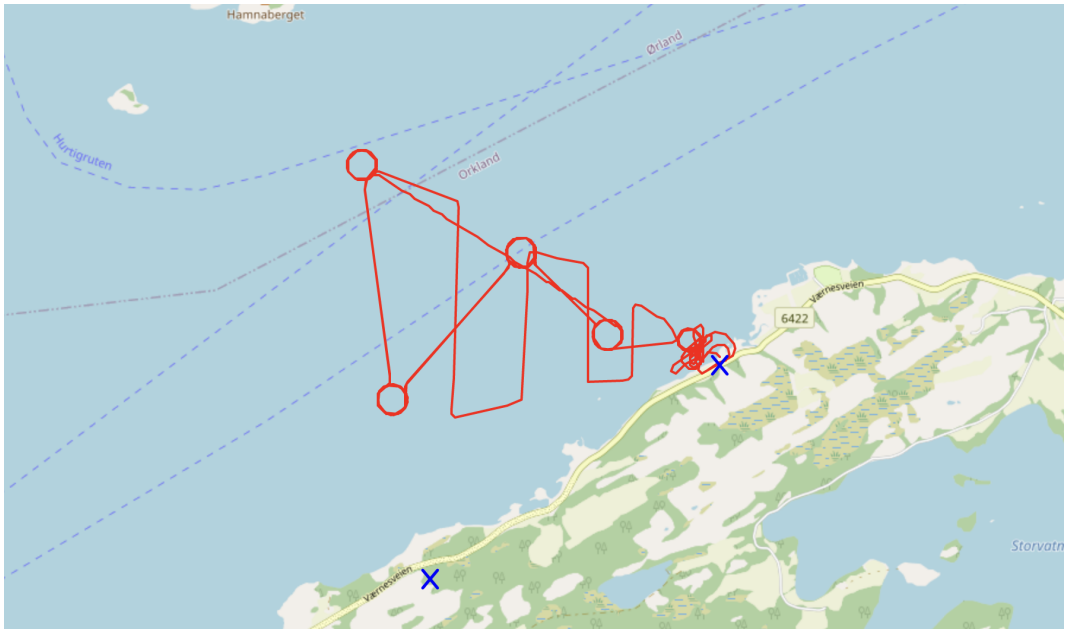


Figure 6.4: Flight path from Raustein, Agdenes over Trondheimsfjorden. The blue "×-es" correspond to the PARS locations. PARS 1 is located where the UAV starts and stops. PARS 2 is at the lower left side. Map data from OpenStreetMap. ©OpenStreetMap contributors [ODbL]

6.3 Coordinate frames

The local NED navigation frame $\{n\}$ is defined at the center of the GNSS measured position of PARS 1 which is at $\{63.61552°,\ 9.59161°,\ 44.6\text{m}\}$. The PARS frames, where the elevation and azimuth measurements are defined, are denoted $\{r_1\}$ and $\{r_2\}$. The position of PARS 2 in $\{n\}$ is approximately

$$\boldsymbol{p}^n_{nr_2} = \begin{bmatrix} -1525.4 & -2082.9 & -37.9 \end{bmatrix}^\top \text{m}$$

Further, it is necessary to obtain the rotations $\boldsymbol{R}^n_{r_1}$ and $\boldsymbol{R}^n_{r_2}$ that aligns the measurements in unit vector form to NED.

6.4 Offline calibration of PARS pose

Little post-processing was done to the data set, including no filtering or any form of outlier removal, as its necessary to gauge how the algorithms can handle these. However, even though there were made some measurements of the radio poses during the test with a compass and GNSS, these were not of sufficient accuracy to achieve accurate estimates using the two PARS systems deployed in the flight, see Figure 6.5. Therefore, the radio poses needed to be calibrated, and the measurements transformed before running the estimation algorithms. The problem of estimating PARS angles on this data set online using a multiplicative extended Kalman filter was tackled in [16]. However, the angles for flight three were not published, and the radios might have been moved between flights. Here an offline approach is taken before the filters are run, which also finds the translation in the case that the radios were moved.

Even though this work only uses bearing measurements, the PARS system used during the test flight actually also produces range measurements. These are of course not used in the estimation algorithms in this work. However, they were used to calibrate the PARS poses before testing the algorithms. To do this offline calibration, the point cloud alignment technique based on the Singular Value Decomposition (SVD) described in [56] was used. Two sets of points of GNSS measurements transformed to NED and PARS measurements in $\{r_1\}$ frame were interpolated and sampled at the same time instance. These are denoted $\boldsymbol{p}^n_{nb,i}$ and $\boldsymbol{p}^{r_1}_{r_1 b}$ for point $i \in \{1, 2, \ldots, N\}$. The rotation $\boldsymbol{R}^n_{r_1}$ and position of the radio $\boldsymbol{p}^n_{nr_1}$ that minimizes the distances between these corresponding point pairs was then found. As described in [56], this was done by first moving all points to the center by subtracting the mean of each set

$$\begin{aligned} {}'\boldsymbol{p}^n_{nb,i} &= \boldsymbol{p}^n_{nb,i} - \bar{\boldsymbol{p}}^n_{nb} \\ {}'\boldsymbol{p}^{r_1}_{r_1 b,i} &= \boldsymbol{p}^{r1}_{r_1 b,i} - \bar{\boldsymbol{p}}^{r_1}_{r_1 b} \end{aligned} \tag{6.1}$$

with

$$\begin{aligned} \bar{\boldsymbol{p}}^n_{nb} &= \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{p}^n_{nb,i} \\ \bar{\boldsymbol{p}}^{r_1}_{r_1 b} &= \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{p}^{r_1}_{r_1 b,i} \end{aligned} \tag{6.2}$$

Then the so-called "cross covariance matrix" was computed as

$$\boldsymbol{W} = \sum_{i=1}^{N} {}'\boldsymbol{p}^n_{nb,i} \left({}'\boldsymbol{p}^{r_1}_{r_1 b,i} \right)^\top \tag{6.3}$$

Finally by taking the SVD of $\boldsymbol{W}$, that is $\boldsymbol{U}$, $\boldsymbol{D}$, $\boldsymbol{V}^\top = \text{svd}(\boldsymbol{W})$ the rotation and translation was recovered as

$$\begin{aligned} \boldsymbol{R}^n_{r_1} &= \boldsymbol{U}\boldsymbol{V}^\top \\ \boldsymbol{p}^n_{nr_1} &= \bar{\boldsymbol{p}}^n_{nb} - \boldsymbol{R}^n_{r_1} \bar{\boldsymbol{p}}^{r_1}_{r_1 b} \end{aligned} \tag{6.4}$$

which gives the rigid transformation sought after.

Table 6.2: The calibration parameters found using compass and ground GNSS, and using point cloud alignment.

	Compass calibration		Offline calibration	
	PARS 1	PARS 2	PARS 1	PARS 2
North [m]	0.0	-1525.4	5.1	-1526.7
East [m]	0.0	-2082.9	0.3	-2086.3
Down [m]	0.0	-37.9	-0.9	-13.9
Roll [°]	0.0	0.0	1.9	2.6
Pitch [°]	0.0	0.0	2.1	0.3
Yaw [°]	-65.5	26.7	-75.0	16.4

Table 6.3: RMSE between GNSS and PARS data with manual compass calibration and offline calibration.

	Compass calibration		Offline calibration	
	PARS 1	PARS 2	PARS 1	PARS 2
North [m]	245	199	66.0	13.0
East [m]	111	353	17.7	26.0
Down [m]	84.3	67.8	62.2	42.1

This was also done for the second PARS to recover a better estimate of the pose. The resulting pose parameters are found in Table 6.2, which also displays the parameters found using only a compass and GNSS. The result of applying the rigid transformation from compass measurements to the PARS points are found in Figure 6.5. Likewise, the result of applying the rigid transformation from point alignment is found in Figure 6.6. The RMSE between the range added PARS measurements and GNSS is shown in Table 6.3. Note that the presence of outliers drives the RMSE up quite high, but still, the RMSE is up to an order of magnitude lower for the aligned parameters. A thing to note is the difference in position for the aligned parameters; there are some differences here and between the GNSS-measured position of the radios which are probably more correct. However, a test was run using the angle parameters from the alignment as well as the measured position using GNSS and this resulted in poorer performance. The discrepancy could be explained by that the ground radios were moved slightly before flight three or that the alignment procedure is distorted by outliers.

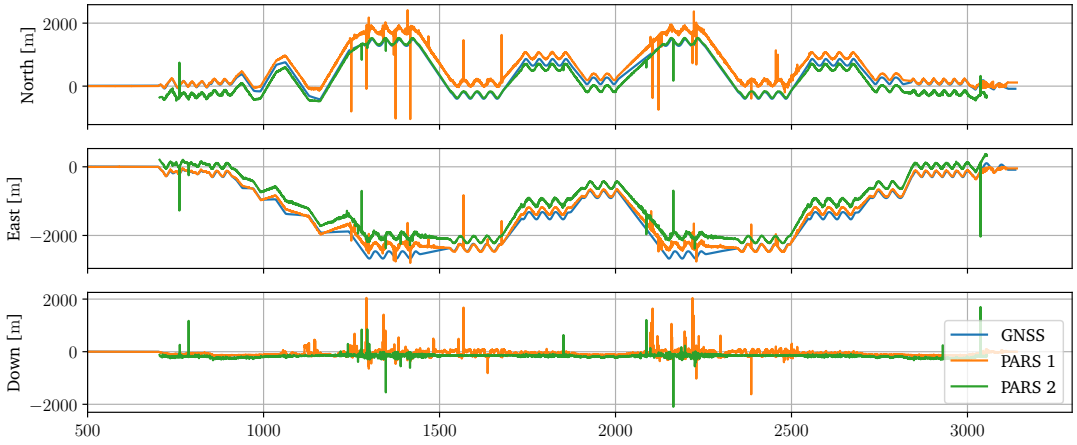


Figure 6.5: Compass calibration. PARS measurements multiplied by range along with GNSS data. Here the measurements are transformed to NED using manual compass measurements of −65.5° for PARS 1, and 26.7° for PARS 2.

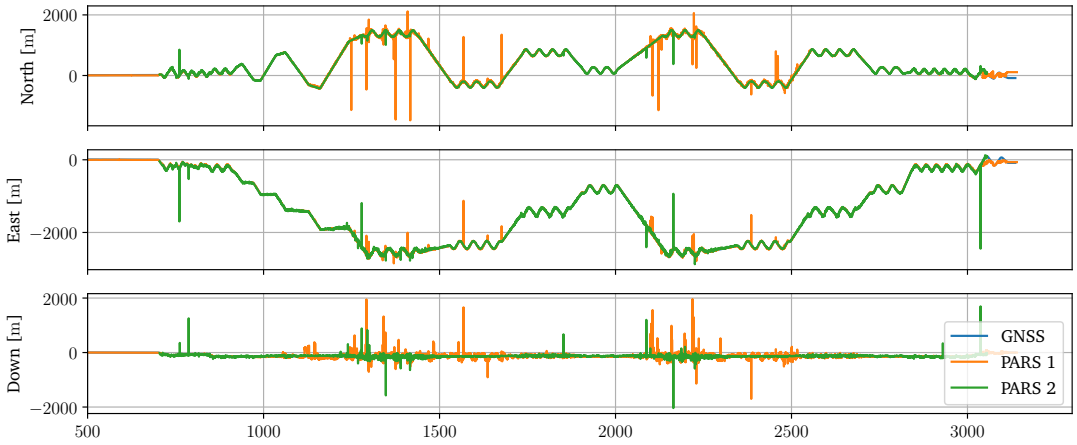


Figure 6.6: Offline calibration. PARS measurements multiplied by range along with GNSS data. Here the measurements are transformed to NED using a point cloud alignment technique.

6.5 Outlier rejection

As is clear from Figure 6.6, the PARS measurements contain a substantial amount of outliers. This is primarily dominant in the elevation measurements which are considerably affected by multipath. To mitigate this, it was deemed necessary to implement outlier rejection in both the error state Kalman filter and the factor graph smoother. The approaches used are completely different; in the ESKF the innovation covariance is utilized, while in the FGO smoother a *robust error model* was used. For ESKF an elevation and azimuth measurement pair is accepted if

$$(\boldsymbol{z}-\boldsymbol{h}(\boldsymbol{x}))\left(\boldsymbol{H}\boldsymbol{P}^{-}\boldsymbol{H}^{\top}+\boldsymbol{R}_{z}\right)^{-1}(\boldsymbol{z}-\boldsymbol{h}(\boldsymbol{x}))^{\top}<\gamma_{2}(r) \tag{6.5}$$

as given in in e.g. [57, Ch. 7.6], where $\gamma_2(r)$ denotes the χ^2 *percent point function* which is the inverse of the cdf with two degrees of freedom in this case. The parameter to this function is such that $0 \leq r \leq 1$. If a measurement is rejected the update step is cancelled. A potential problem with this approach is that both measurements are rejected when it might be the case that only one measurement is an outlier and need to be discarded.

For the FGO smoother, measurements are not rejected, but down-weighted inside the optimization, using the Huber robust error model. The idea being that instead of just squaring the error as in (2.72) one can use different function ρ of the error e that reduces the effects of outliers [58, 59]. For the Huber estimator, the ρ function is

$$\rho(e)=\begin{cases}\frac{1}{2}e^{2} & \text{if } |e|<k\\ k(|e|-\frac{1}{2}k) & \text{else}\end{cases} \tag{6.6}$$

which it can be shown that leads to a weight function $w(e)$

$$w(e)=\begin{cases}1 & \text{if } |e|<k\\ \frac{k}{|e|} & \text{else}\end{cases} \tag{6.7}$$

that can be multiplied with the error in (2.72). In `GTSAM` this is implemented in the `mEstimator` noise model class [32], and utilizing it is as simple as replacing the noise model used this far with this robust version. Then the measurement errors are automatically reweighed and no other code needs to be written to do outlier rejection. The choice of the parameters r and k was done by simple trial and error; the tuning used is found in Table 7.1.

6.6 Online calibration of PARS pose

For accurate navigation using PARS it is crucial the poses of the radios are known accurately. Although the PARS pose calibration point cloud alignment technique used in Section 6.4 seem to work, it has the major limitation that it cannot be used online when the UAV is flying, as well as the fact that the range measurements were used which is not supposed to be within the scope of this project. To remedy this, an online calibration algorithm was developed for the factor graph optimization smoother. The PARS factor

was modified to take in a PARS ground pose variable $\boldsymbol{T}_r^n$ as well as the vehicle pose $\boldsymbol{T}_b^n$. The Jacobian in (4.7) is also needed. The modified PARS factor is displayed in Code listing 6.1

Code listing 6.1: PARS factor that also includes the PARS pose.

```
# ...
# custom factor error function inside PARS sensor class
def J_h_P_man(self, X, P):
    p_nb_n = X.translation()
    p_rb_r = P.transformTo(p_nb_n)
    J = self.J_alpha_p_rb_r(p_rb_r)@np.block([skew(p_rb_r), -np.eye(3)])
    return J

def J_h_X_man(self, X: gtsam.Pose3):
    R_rn = self.P.rotation().matrix().T
    R_nb = X.rotation().matrix()
    p_rb_r = self.P.transformTo(X.translation())
    J = np.zeros((2,6))
    J[:,3:] = self.J_alpha_p_rb_r(p_rb_r)@(R_rn@R_nb)
    return J

def cf_error_pars_calib(self,
                z: np.ndarray,
                this: gtsam.CustomFactor,
                values: gtsam.Values,
                jacobs: Optional[List[np.ndarray]]):
    X = values.atPose3(this.keys()[0]) #T_nb
    P = values.atPose3(this.keys()[1]) #T_nr
    p_nb_n = X.translation()
    error = np.array([ssa(self.h(p_nb_n)[0] - z[0]),
                      ssa(self.h(p_nb_n)[1] - z[1])])
    if jacobs is not None:
        jacobs[0] = self.J_h_X_man(X)
        jacobs[1] = self.J_h_P_man(X, P)
    return error
# ...
```

In contrast to the vehicle pose variables that are created on each new measurement, there is only one variable for each PARS pose as illustrated in Figure 6.7. There might be one or more measurement factors on each $\boldsymbol{T}_b^n$ variable. In addition, for this setup GNSS factors are also placed on the pose variables. Because of this, the online calibration has its own result section and was not used to generate the results in Section 7.2. The PARS pose priors were set as the manual calibration values in Table 6.2.

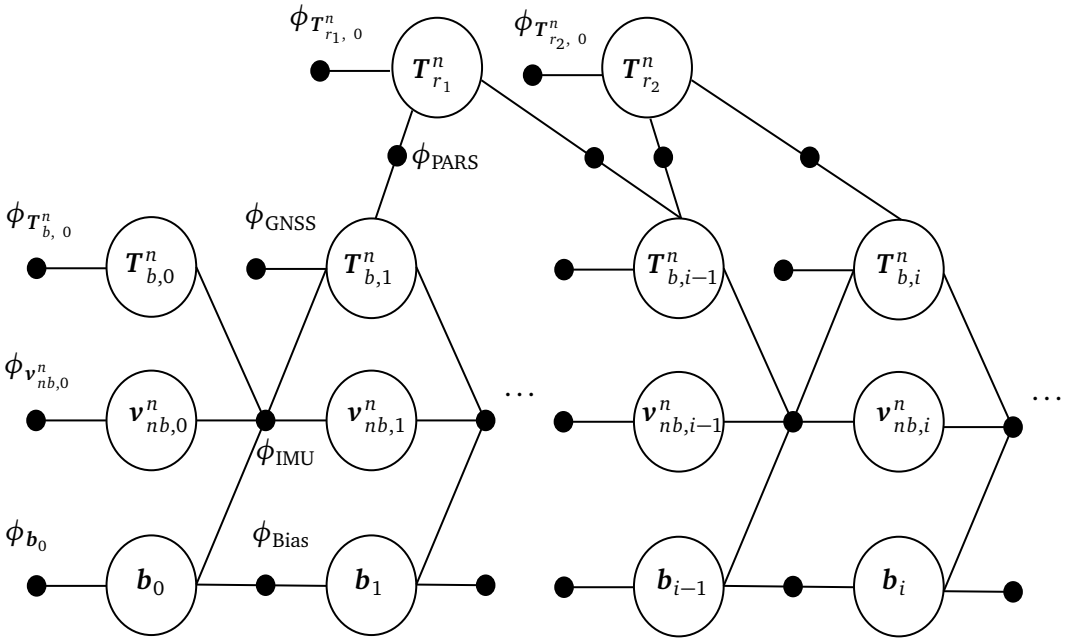


Figure 6.7: Factor graph used for online calibration

Chapter 7

Experimental Results and Discussion

Two sets of experiments were performed on the Raustein data set: firstly a GNSS-only run to establish a baseline in expected performance and then a run using the two PARS. The flight path as well as the PARS positions are displayed in Figure 6.4. The hardware used and the experimental setup is described in Chapter 6. As mentioned there, the PARS poses need accurate calibration. In the two experiments the offline calibration from Section 6.4 is used. The last section in this chapter will detail the online calibration results from applying the factor graph in Section 6.6

A summary of the results are found in Table 7.2 which displays the RMSE between RTK GNSS and position estimates, as well as between the onboard AHRS and the attitude estimates for both runs with the ESKF and FGO filters. It is important to note that there are errors in both the GNSS measurements and the AHRS system, so this can not be considered ground truth. Further, the RTK GNSS measurements are used on the GNSS run as well as to calculate the RMSE on position which is problematic for a performance baseline. However, due to the lack of any better reference signals, there is no other way to compare performance. Nevertheless, the information available is sufficient to show that the fusion of PARS with IMU works.

The tuning parameters used in both experiments are displayed in Table 7.1. The PARS pose calibration used was the one from the offline method shown in Table 6.2. The starting point for tuning the elevation and azimuth measurements was done as a result of the outcome in Table 6.3 where PARS 2 has a higher RMSE on the measurements compared to PARS 2, so the standard deviation was set higher for this radio.

Table 7.1: Tuning parameters used for the estimators. Acc. / Acc. b scaling refers to how the noise is scaled in relation to the values in Table 3.1.

Type	Value	Unit
GNSS std.	2	[m]
PARS 1 Elev std.	15	[°]
PARS 1 Azi std.	10	[°]
PARS 2 Elev std.	10	[°]
PARS 2 Azi std.	5	[°]
Acc. / Acc. b scaling	22 / 1	—
Gyr. / Gyr. b scaling	22 / 1	—
Smoother fixed lag	1, 30	[s]
ESKF χ^2 ppf, r	0.95	—
Huber cost param. k	0.15	—

Type	Value	Unit
$p^n_{nb,0}$	11.0, 8.0, 0.9	[m]
$v^n_{nb,0}$	**0**	$[\mathrm{m\,s^{-1}}]$
$q^n_{b,0}$	-1.5, 11.7, -54,6	[°]
$b_{f,0}$	**0**	$[\mathrm{m\,s^{-2}}]$
$b_{\omega,0}$	**0**	$[\mathrm{rad\,s^{-1}}]$
$p^n_{nb,0}$ std.	5	[m]
$v^n_{nb,0}$ std.	2	$[\mathrm{m\,s^{-1}}]$
$q^n_{b,0}$ std.	10	[°]
$b_{f,0}$ std.	0.0686	$[\mathrm{m\,s^{-2}}]$
$b_{\omega,0}$ std.	0.0017	$[\mathrm{rad\,s^{-1}}]$

Table 7.2: RMSE experimental results comparison between position and attitude estimates with RTK GNSS and AHRS. The FGO estimator is run with a fixed lag of 1s, and 30s.

		GNSS			PARS		
Type	Unit	ESKF	FGO 1[s]	FGO 30[s]	ESKF	FGO 1[s]	FGO 30[s]
North	[m]	0.056	0.056	0.056	9.9	10.1	10.2
East	[m]	0.055	0.054	0.054	14.1	14.2	14.0
Down	[m]	0.033	0.033	0.033	13.0	11.2	11.1
Roll	[°]	2.4	2.4	2.4	2.3	2.3	2.3
Pitch	[°]	2.9	2.9	2.9	2.9	2.9	2.9
Yaw	[°]	18.1	18.1	18.1	18.1	18.2	18.2

7.1 GNSS-aided INS

A RTK GNSS only run was carried out to establish a baseline in expected performance before using PARS measurements. Error plots are found in Figure 7.1 and position, attitude, and biases are shown in Figure 7.2. The position estimate errors are in the centimeter range and the roll and pitch closely agree with the AHRS having errors of about $2-3°$. Yaw has a higher error, but this is expected due to the lack of a compass, so the heading has to be estimated through the coupling of the dynamics. This is also true for the UAV autopilot's AHRS since the magnetometer can be disabled to avoid magnetic-field disturbances of the yaw estimate. The fact that the heading angle is difficult for the AHRS on the autopilot to estimate, makes it hard to conclude with certainty that the heading estimate is accurate. Further, there is uncertainty connected to the relative mounting of the STIM300 and the autopilot. However, the fact that the ESKF and FGO yield similar results makes it likely that the heading is as accurate as possible without

compensating for more errors like the IMU mounting angle.

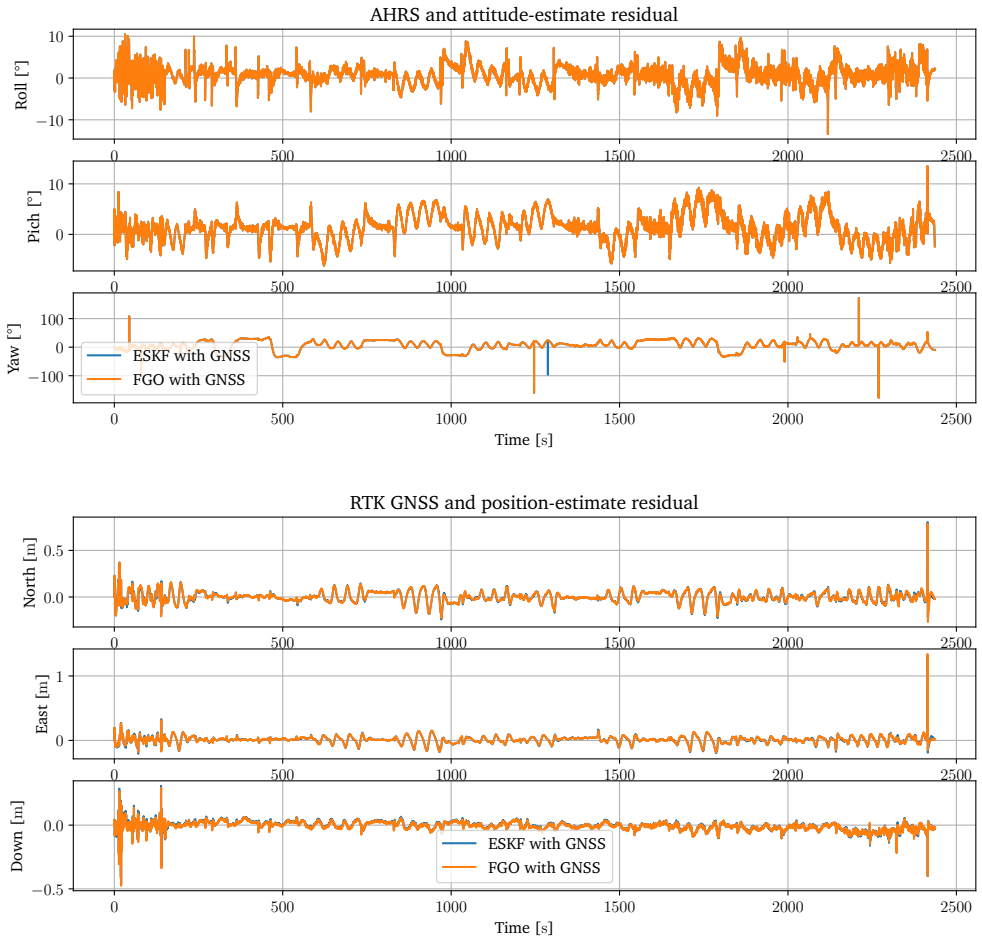


Figure 7.1: Experimental results with GNSS sensor. Error from onboard AHRS and GNSS. Comparing the position estimates from the filters that are run with GNSS measurements with the corresponding GNSS measurements is not very insightful. However, it shows that the filters work.

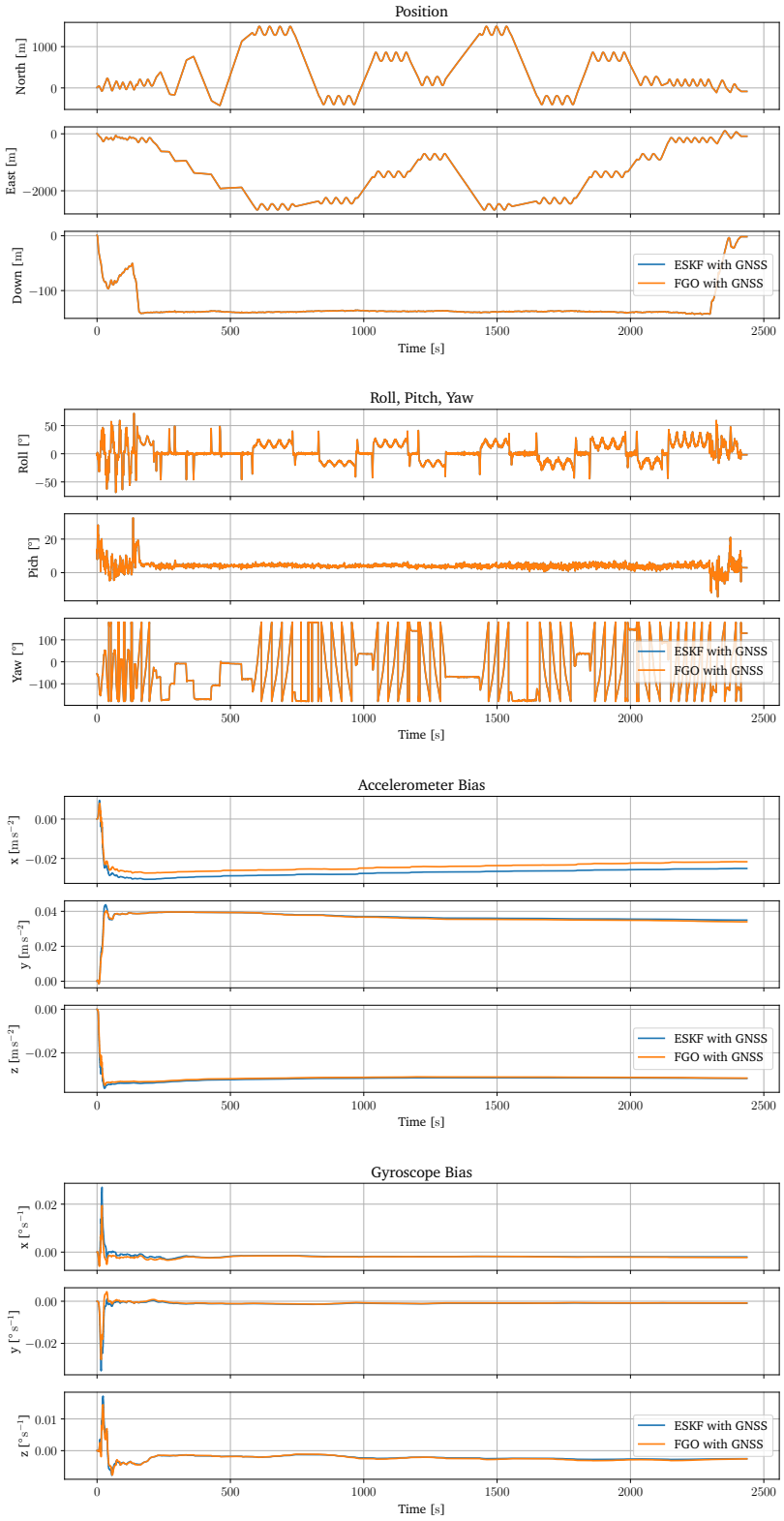


Figure 7.2: Experimental results with GNSS. Top to bottom: position, attitude, accelerometer bias, gyroscope bias.

7.2 PARS-aided INS using two ground radios

This run uses both PARS located as displayed in Figure 6.4. As the UAV comes back to land the radios drop out, and therefore the last minute and a half is cut off from the plots and not used in the RMSE calculations. Figure 7.3 shows the error between the position and attitude estimates and GNSS and AHRS. Note that the position errors at times get quite high – up to 50m for ESKF – probably due to the high number of outliers caused by multipath. As Table 7.2 shows, most states do not have considerable performance differences between the ESKF and FGO estimators. However, the FGO smoother gives a lower RMSE in the down position direction. As seen in Figure 7.3 there is a considerable peak in the ESKF down-position estimate that is not seen in the FGO errors.

As mentioned earlier, the elevation angle measurements are those that are the most affected by multipath, and therefore the down component might be seen as the hardest to estimate. The fact that the factor graph approach gives a lower RMSE in down, could be an effect caused by the FGO smoothing or from the different outlier rejection schemes, or a combination. Seeing as increasing the window length from 1s to 30s does not give a performance increase alludes to the possibility that the outlier rejection is more prominent. However, as we shall see in Section 7.2, removing the outlier scheme on both estimators makes the FGO perform better in all position components. In any case, for this experiment, the factor graph approach seems to be more robust towards outliers.

The attitude errors are approximately equal to those using GNSS. Even the heading angle is commensurate to the run using GNSS only, which is promising. The biases still converge to approximately the same values as in the GNSS run except for the accelerometer x-axis which converges to a value slightly higher than with GNSS. This could be explained by the fact that the same initial covariance is used on GNSS and FGO runs; it is clear that with PARS, the biases need more time to settle, so increasing the initial covariance may lead to the same bias for the accelerometer x-axis with ESKF.

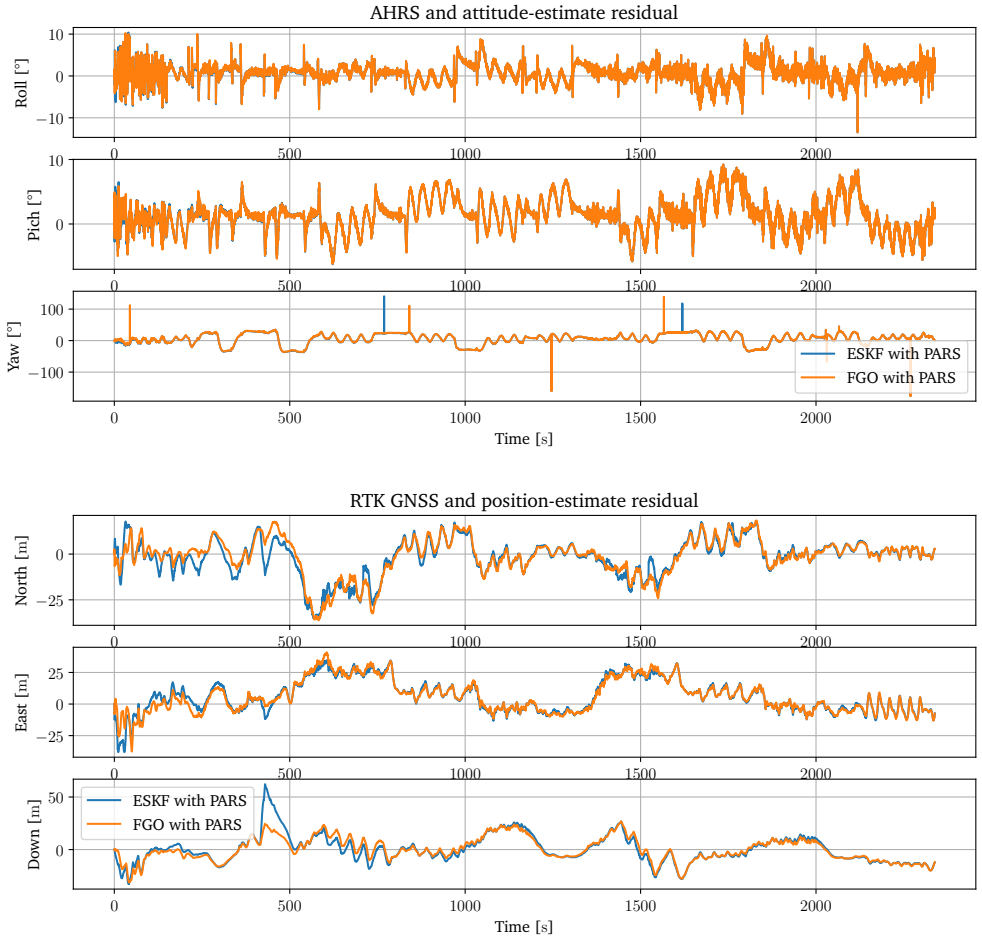


Figure 7.3: Experimental results with PARS. Error from onboard AHRS and RTK GNSS.

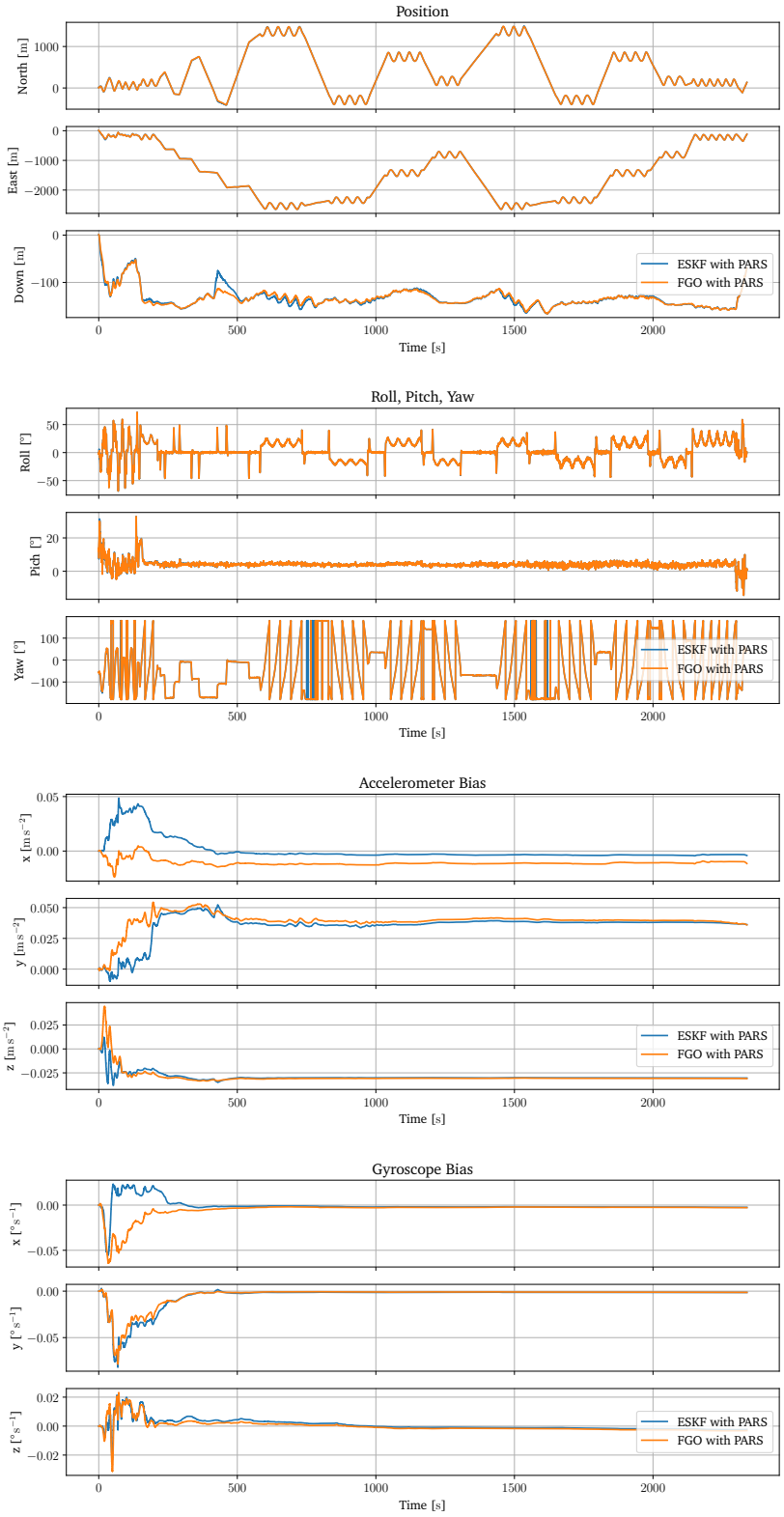


Figure 7.4: Experimental results with PARS. Top to bottom: position, attitude, accelerometer bias, gyroscope bias.

The effect of translation calibration

The PARS pose calibration used was the one from the offline method, as in Table 6.2. One issue with these parameters is that there was as mentioned some discrepancy between the position of the PARS measured position with RTK GNSS and the one found using the offline calibration technique. As the RTK GNSS measurements of the PARS position is assumed to be very accurate, an experiment similar to the one described in this section was performed, with the only difference being that the RTK GNSS measurements of the PARS positions were used instead of the position achieved through alignment calibration. The orientation was kept the same as in the alignment calibration. The result of this is showed in table Table 7.3, as seen using the accurate RTK GNSS positions gives a higher RMSE in position. As discussed earlier, this might be because the PARS were moved and the positions that accompany the dataset is not accurate, or that the alignment adjusts for some other effect.

Table 7.3: Result with PARS RTK GNSS position calibration and alignment rotation. Both estimators give a higher RMSE in position when using the GNSS calibrated PARS positions. Here a fixed lag of 1s on the FGO smoother was used as above.

Type	Unit	ESKF	FGO	Type	Unit	ESKF	FGO
North	[m]	12.6	12.6	Roll	[°]	2.3	2.3
East	[m]	16.0	16.6	Pitch	[°]	2.9	2.9
Down	[m]	16.6	15.7	Yaw	[°]	18.1	18.3

The effect of outlier rejection

To test whether the outlier rejection schemes were effective, a similar run was performed on the PARS measurements except for this time outlier rejection was turned off for both estimators. Here the FGO was run with a lag of 1s. The position error is plotted in Figure 7.5 where it is clear that both estimators are very sucpectible to the outliers, see for instance, the peak in down-position at approximately 450s. The RMSE displayed in Table 7.4 is considerably higher in all position estimates. Interestingly, the FGO gives a lower RMSE in position, from Figure 7.5 this seems to be predominantly from the ESKF struggling during the takeoff phase.

Table 7.4: RMSE of position and attitude for without outlier rejection on PARS measurements.

Type	Unit	ESKF	FGO	Type	Unit	ESKF	FGO
North	[m]	14.3	12.5	Roll	[°]	2.4	2.4
East	[m]	17.9	15.3	Pitch	[°]	2.9	2.9
Down	[m]	15.0	13.5	Yaw	[°]	17.8	18.4

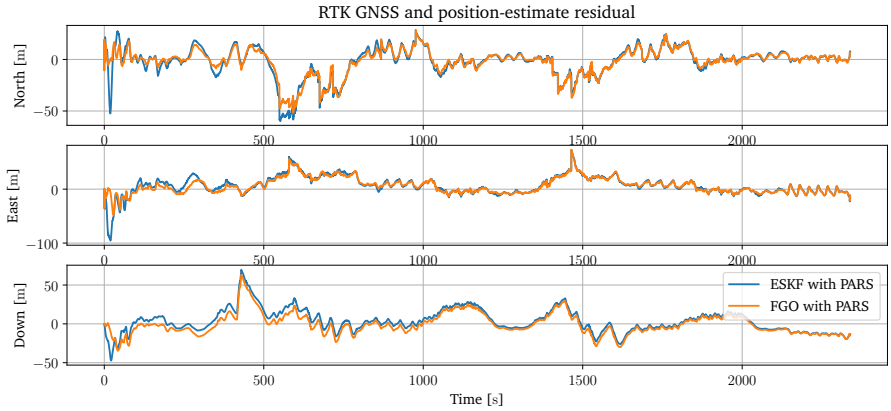


Figure 7.5: Experimental results with PARS with no outlier rejection on. Error from onboard AHRS and RTK GNSS. Turning off the outlier rejection makes the two estimators exhibit very similar behavior. Here a 1s lag is used on the FGO. The RMSE is found in Table 7.4

7.3 Runtime

With regards to runtime, an accurate comparison is difficult as the FGO estimator is run using the `C++ GTSAM` wrappers while the ESKF is implemented in pure Python. While `GTSAM` is built for high performance, the ESKF implementation has not been created with this in mind and runs significantly slower. Regardless, the runtime of each of the estimators was recorded and is displayed in Table 7.5. The part of the dataset that was used has a length of 40min 36s, and the ratio between the runtime and the dataset length is also displayed. An interesting thing to note is how nonlinearly the runtime of the incrementally fixed lag smoother scales with the fixed lag interval when using the PARS factor.

Table 7.5: Runtime on field data. The length of the dataset is 40min 36s. The relative row refers to the ratio of runtime to actual time.

	GNSS			PARS		
	ESKF	FGO 1[s]	FGO 30[s]	ESKF	FGO 1[s]	FGO 30[s]
runtime	17 min 34s	3 min 31s	3 min 40s	18 min 03s	5 min 11s	4 h 09min
relative	40.1%	8.6%	9.0%	44.5%	12.8%	613%

7.4 Online calibration

To test the factor graph online calibration algorithm outlined in Section 6.6 the same PARS experiment as above was run with some changes. Firstly the new PARS factor was of course used, with two new variables, namely the pose of PARS 1 and PARS 2. In addition, GNSS factors were placed on the variables when they were available. The changes in tuning made from the last section is found in Table 7.6. The PARS poses were initialized with the GNSS and compass measurements found in Table 7.7

The resulting PARS pose estimation up to approximately 85s is found in Figure 7.6, and the full length is displayed in Figure C.2 in the appendix. The full length is slightly before the UAV returns to start because the PARS lose coverage. For both radios the estimated poses converge rapidly. However, as seen PARS 1 takes a bit longer to converge than PARS 2. When PARS 2 gets its first measurements, yaw converges almost instantly. The fact that PARS 1 converges slower, could be because there are more outliers in the measurements from this radio which are down weighed resulting in slower convergence. The difference in the number of outliers is clearly seen in Figure 7.7, which will be introduced later.

The PARS mounting translation and orientation after the full run is displayed in Table 7.8, which also shows the result from offline calibration and the angles achieved in [16] on flight 2. Although the orientations found match quite well, there is some discrepancy, especially in the translation.

Like the offline method, the online calibration method also seems to give a position shift. In Figure 7.6, this is clear for PARS 1, and in the full run in Figure C.2, the mounting position for PARS2 also changes by a few meters. This may be the effect of outliers or that the initialization is inaccurate. The fact that the position movement differs so much between the offline and online calibration does not corroborate the theory that the radio was moved and the GNSS-measured position is inaccurate. It seems more likely that GNSS measured position is correct. In any case, one could add a stronger prior on the PARS position if one is absolutely certain that the GNSS-measured position is accurate. To simply not optimize the translation at all seems like a better approach as it is relatively easy to measure the position of the radio accurately.

Table 7.6: The tuning variables that have been changes since Section 7.2, the rest is the same as in table Table 7.1.The poses were initialized using the manual calibration parameters found in Table 6.2.

Type	Value	Unit
GNNS all axes std.	1	[m]
PARS 1 Elevation std.	0.1	[°]
PARS 1 Azimuth std.	0.1	[°]
PARS 2 Elevation std.	0.1	[°]
PARS 2 Azimuth std.	0.1	[°]
Fixed Lag	10	[s]

Type	Value	Unit
PARS North prior std.	1	[m]
PARS East prior std.	1	[m]
PARS Down prior std.	1	[m]
PARS Roll prior std.	1	[°]
PARS Pitch prior std.	1	[°]
PARS Yaw prior std.	15	[°]

Table 7.7: Online calibration pose initialization. Recall that NED origin is defined centered at the GNSS measured position of PARS 1.

	North [m]	East [m]	Down [m]	Roll [°]	Pitch [°]	Yaw [°]
Init.	0	0	0	0	0	-65.5
	-1525.4	-2082.9	-37.9	0	0	26.7

Table 7.8: Online calibration results. The last two rows are the results from flight 2 in the extended Okuhara paper [16] table 1 and table 2.

	North [m]	East [m]	Down [m]	Roll [°]	Pitch [°]	Yaw [°]
Online	7.6	4.9	2.5	2.1	1.7	-75.9
	-1537.7	-2087.1	-27.2	2.1	-1.4	16.7
Offline	5.1	0.3	-0.1	1.9	2.1	-75.0
	-1526.7	-2086.3	-13.9	2.6	0.3	16.4
[16]	—	—	—	0.0040	-0.0014	-74.7
	—	—	—	-0.00016	-0.00044	16.2

Secondly, as seen in Table 7.8, the roll and pitch angles for the offline and online calibration are in the order of 2° while these angles are about three orders of magnitude smaller in [16]. This is probably because of the relatively high prior of 1° put on these angles. This is considered reasonable as manually leveling the antenna to an accuracy of less than 1° seems improbable. Most importantly, the yaw angles for the online method converge to approximately the same as with the offline method and in [16]. This is encouraging as this is the hardest to calibrate as described in [16].

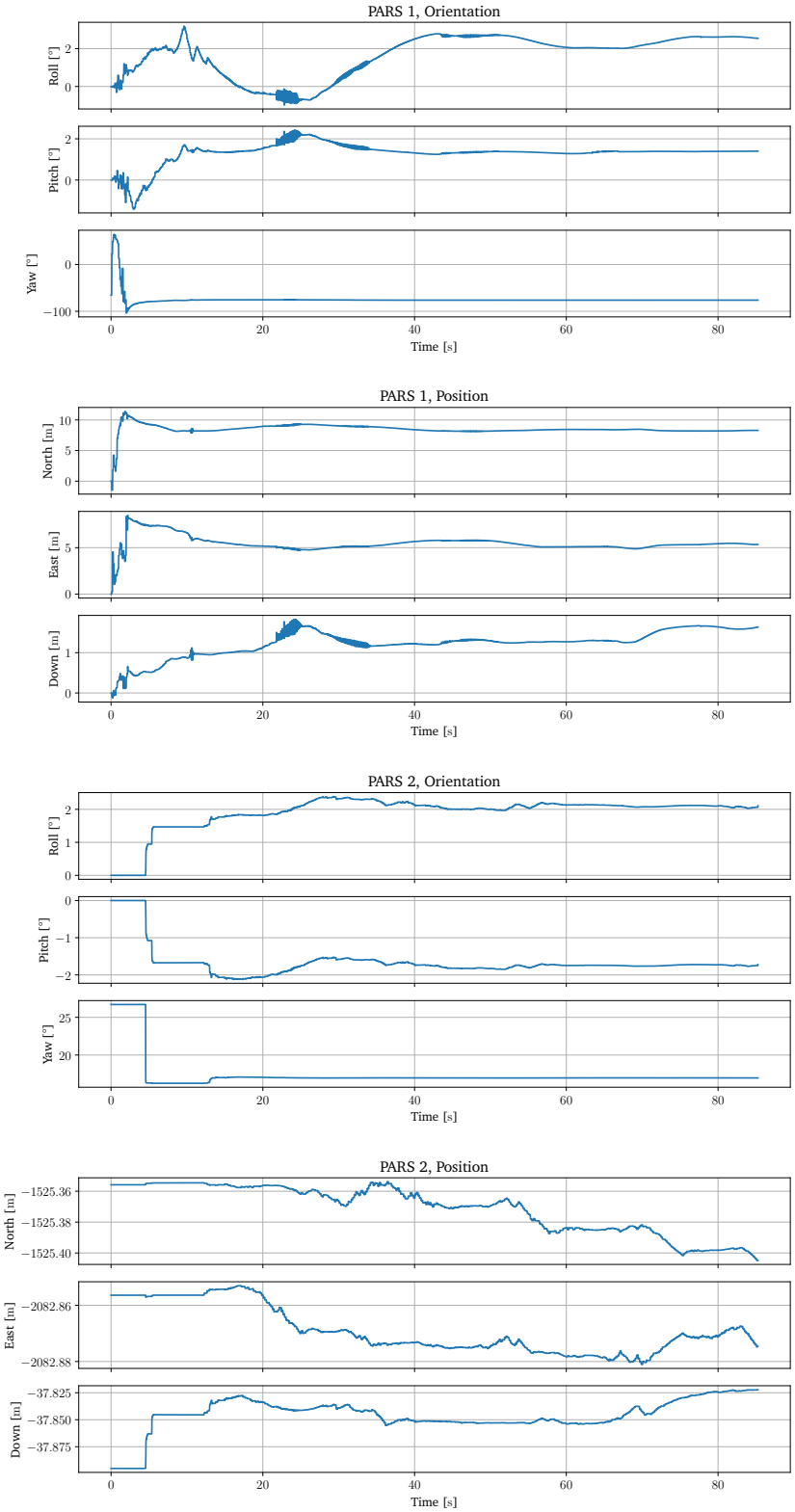


Figure 7.6: Online calibration results. PARS position and orientation. This figure shows approximately the first 85s, the full run is found in Figure C.2 in the appendix.

To verify the achieved PARS poses, the RTK GNSS measurements of the vehicle was utilized. The GNSS measurements were transformed to the $\{r\}$-frames and projected to elevation and azimuth angles using the obtained poses. The error between this and the PARS measurements was then calculated and plotted in Figure 7.7. As seen, the error hovers around 0° indicating that the calibration is accurate. A similar figure is made using the compass and RTK GNSS measured PARS pose and is found in Figure C.1 in the appendix. Here, the azimuth error is clearly biased and hovers around 10° indicating inaccurate calibration.

In Table 7.9, the RMSE from this type of experiment is shown for the compass and GNSS, online and offline calibration procedures as well as using the values from [16]. To try to avoid the RMSE being too affected by the outliers, all errors larger than 15° were removed from the RMSE calculation. It is important to note that the values from Okuhara are from a different flight, so this is not a fair comparison, as multiple factors could have changed between flights. In any case, all calibration procedures give an error at least an order of magnitude lower than using manual calibration, except for PARS 1 elevation. This performance increase might not seem that great, but over large distances, small differences in measured angles will result in big discrepancies. The fact that little performance increase is seen in PARS 1 elevation is probably because the measurements here are heavily affected by multipath as seen in Figure 7.7, and the roll-pitch leveling was accurately done.

Further, the offline calibration seems to give the best results. This is probably because this procedure had access to the range measurements, and by using this extra piece of information higher accuracy is achieved. The online calibration algorithm seems to work, yielding PARS mounting orientation estimates that are commensurate with the offline procedure and in the Okuhara paper.

Table 7.9: Online calibration elevation and azimuth RMSE from projected RTK GNSS.

	Unit	Compass	Online	Offline	[16]
Pars 1. Elev.	[°]	2.68	2.02	2.10	2.68
Pars 1. Azi.	[°]	8.90	0.96	0.84	1.34
Pars 2. Elev.	[°]	1.46	0.95	0.88	1.46
Pars 2. Azi.	[°]	10.32	0.64	0.53	0.58

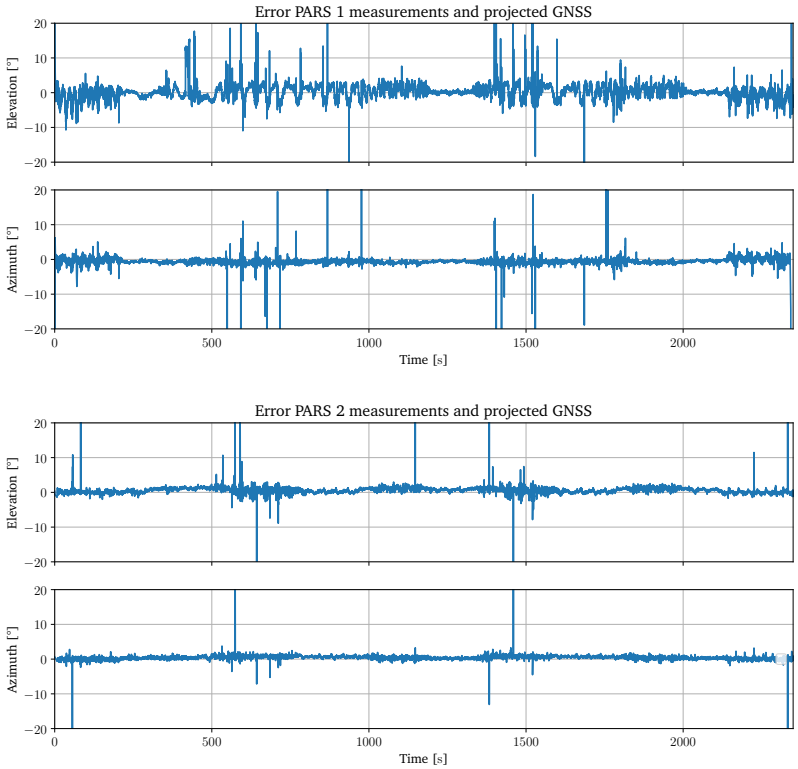


Figure 7.7: Error between PARS measured angles and GNSS measurements projected to elevation and azimuth using calibrated pose in Table 7.8.

Chapter 8

Overall discussion of experimental results

Overall, FGO seems to give better performance than the ESKF in terms of RMSE error between ground truth. Still, the performance increase is not as large as what was hoped. In the simulation scenarios, this might be expected as the measurements used here do not contain outliers. However, the data from the field experiments do contain a significant amount of outliers, and the expectation was that a fixed interval smoothing approach would mitigate these to a larger degree. As mentioned in Section 7.2, increasing the lag from 1s to 30s did not yield a smoother trajectory in the presence of outliers on its own. The reason that increasing the lag does not improve performance is unclear. It could be a tuning problem where the IMU factors have too much uncertainty associated with them, thus making them unable to constrain the motion of the vehicle when PARS outliers arrive. On the other hand, the factor graph approach seems to be more robust towards poor initial estimates, where smoothing seems to make a difference. This became especially clear when outlier rejection was turned off in Section 7.2, where FGO performed better in the takeoff phase.

The introduction of the Huber robust error model did have a considerable positive effect in terms of tolerating outliers. Additionally, this approach required very little effort or tuning to implement, in contrast to the ESKF, which requires the estimator to be highly consistent for the outlier rejection approach to be effective. This might be the main reason better outlier rejection was achieved using the FGO approach. Due to time constraints, not much time was spent tuning the estimators, and for an accurate comparison, the same tuning needed to be used for both estimators requiring a tradeoff. In light of this, the fact that the outlier rejection scheme in `GTSAM` was simple to setup without as much testing or extensive tuning is promising with regard to practical usability. Filter tuning is a time-consuming task, and one can assume that faster development and field deployment can be achieved using said method.

Perhaps the main benefit of the factor graph framework, experienced throughout this project, is how modifiable it is when it comes to adding additional states or measurements. Because of this, there was enough time to add the radio mounting poses in the optimization and experiment with online calibration.

Chapter 9

Concluding remarks

9.1 Further work

Concerning further work, several aspects could be improved. When it comes to outlier rejection using the Kalman filter, it could be beneficial to implement a sequential ESKF that does not reject both measurements in case only one is an outlier. The next step in online calibration is to calibrate the PARS pose online without GNSS measurements. The problem with this is that PARS gives two measurements while the pose has six degrees of freedom. However, if one were to use the range measurement as well, and only calibrate rotation, there would be the same constraints as unknowns. Another approach could be to let the measurements accumulate for a while before one starts the optimization of the pose. In addition, the code for both estimators should be ported to `C++` to increase performance and maintainability. Using

9.2 Conclusion

In this work, a factor graph based incremental fixed lag smoother has been implemented using the `GTSAM` framework to fuse PARS and IMU measurements. The FGO approach has been compared with an error state Kalman filter on data from simulation and field experiments. The simulation results indicate that the performance of the two approaches is similar, but FGO may yield better performance in the presence of poor initial conditions. To handle the multipath issue present in the PARS measurements from the field data, two different approaches for outlier rejection are tested for the filter and FGO estimators. The results from field experiments gave RMSE in NED position of 9.9, 14.1, 13.0m for the ESKF and 10.2, 14.0, 11.1m for FGO with a lag of 30s. The factor graph yields better results in the down position, which is most affected by multipath. The robust cost outlier rejection was observed to be the main source of performance improvement and requires much less tuning than the χ^2 threshold rejection method. Prior to these experiments, the PARS mounting pose was calibrated using an offline method, which cannot be used in real-time. Therefore a solution to the problem of calibrating the PARS antenna mounting position and orientation online using FGO was proposed and tested.

The results from the online calibration method were promising, yielding pose estimates that were commensurate with the offline method.

References

[1] J. Zidan, E. I. Adegoke, E. Kampert, S. A. Birrell, C. R. Ford, and M. D. Higgins, "GNSS Vulnerabilities and Existing Solutions: A Review of the Literature," *IEEE Access*, vol. 9, pp. 153 960–153 976, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2973759. (visited on 06/11/2023).

[2] S. Strøm, "Kraftig økning av GPS-jamming over Finnmark," *NRK*, Feb. 24, 2023. [Online]. Available: https://www.nrk.no/tromsogfinnmark/kraftig-okning-av-gps-jamming-over-finnmark-1.16309499 (visited on 06/11/2023).

[3] T. F. Kristensen, "Luftambulansen utsatt for massiv GPS-jamming: - Tyder på at det stammer fra Russland," *NRK*, Dec. 23, 2022. [Online]. Available: https://www.nrk.no/tromsogfinnmark/luftambulansen-i-finnmark-trolig-utsatt-for-russisk-gps-jamming-1.16231204 (visited on 06/11/2023).

[4] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned Aircraft Capture and Control Via GPS Spoofing: Unmanned Aircraft Capture and Control," *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, Jul. 2014, ISSN: 15564959. DOI: 10.1002/rob.21513. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/rob.21513 (visited on 06/11/2023).

[5] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Robotics: Science and Systems X*, Robotics: Science and Systems Foundation, Jul. 12, 2014, ISBN: 978-0-9923747-0-9. DOI: 10.15607/RSS.2014.X.007. [Online]. Available: http://www.roboticsproceedings.org/rss10/p07.pdf (visited on 06/11/2023).

[6] 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, Sep. 2017, ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364917728574. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364917728574 (visited on 06/11/2023).

[7] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021, ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2021.3075644. arXiv: 2007.11898 [cs]. [Online]. Available: http://arxiv.org/abs/2007.11898 (visited on 06/11/2023).

[8] N. Khedekar, M. Kulkarni, and K. Alexis, "MIMOSA: A Multi-Modal SLAM Framework for Resilient Autonomy against Sensor Degradation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan: IEEE, Oct. 23, 2022, pp. 7153–7159, ISBN: 978-1-66547-927-1. DOI: 10.1109/IROS47612.2022.9981108. (visited on 06/11/2023).

[9] Y. Wei, "Review of the Evolution of Phased-Array Radar," *SHS Web of Conferences*, vol. 144, A. Luqman, Q. Zhang, and W. Liu, Eds., p. 02 008, 2022, ISSN: 2261-2424. DOI: 10.1051/shsconf/202214402008. [Online]. Available: https://www.shs-conferences.org/10.1051/shsconf/202214402008 (visited on 06/12/2023).

[10] N. Ojaroudiparchin, M. Shen, and G. F. Pedersen, "A 28 GHz FR-4 compatible phased array antenna for 5G mobile phone applications,"

[11] R. Wallis and Sheng Cheng, "Phased-array antenna system for the MESSENGER deep space mission," in *2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542)*, vol. 1, Big Sky, MT, USA: IEEE, 2001, pp. 1/41–1/49, ISBN: 978-0-7803-6599-5. DOI: 10.1109/AERO.2001.931694. (visited on 06/12/2023).

[12] M. L. Sollie, K. Gryte, T. H. Bryne, and T. A. Johansen, "Outdoor Navigation Using Bluetooth Angle-of-Arrival Measurements," *IEEE Access*, vol. 10, pp. 88 012–88 033, 2022, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3199772. (visited on 05/10/2023).

[13] C. Fulton, M. Yeary, D. Thompson, J. Lake, and A. Mitchell, "Digital Phased Arrays: Challenges and Opportunities," *Proceedings of the IEEE*, vol. 104, no. 3, pp. 487–503, Mar. 2016, ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2015.2501804. (visited on 05/18/2023).

[14] K. Gryte, "Precision control of fixed-wing UAV and robust navigation in GNSS-denied environments," NTNU, Trondheim, Jun. 2020.

[15] Chetvorno, *File:Phased array antenna system.svg*, Dec. 2, 2016. [Online]. Available: https://commons.wikimedia.org/wiki/File:Phased_array_antenna_system.svg.

[16] M. Okuhara, T. H. Bryne, K. Gryte, and T. A. Johansen, "Phased Array Radio Navigation System on UAVs: GNSS-based Calibration in the Field," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, Athens, Greece: IEEE, Jun. 15, 2021, pp. 210–218, ISBN: 978-1-66541-535-4. DOI: 10.1109/ICUAS51884.2021.9476807. (visited on 05/18/2023).

[17] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, 2nd. Artech House, 2013, ISBN: 978-1-60807-005-3.

[18] J. R. Carpenter and C. N. D'Souza, "Navigation filter best practices," NASA, Technical Publication (TP) NASA/TP–2018–219822, 2018. [Online]. Available: https://ntrs.nasa.gov/api/citations/20180003657/downloads/20180003657.pdf.

[19] V. Indelman, S. Williams, M. Kaess, and F. Dellaert, "Factor graph based incremental smoothing in inertial navigation systems," in *15th International Conference on Information Fusion*, Singapore, pp. 2154–2161. [Online]. Available: https://ieeexplore.ieee.org/document/6290565.

[20] S. M. Albrektsen, T. H. Bryne, and T. A. Johansen, "Phased array radio system aided inertial navigation for unmanned aerial vehicles," in *2018 IEEE Aerospace Conference*, Big Sky, MT: IEEE, Mar. 2018, pp. 1–11, ISBN: 978-1-5386-2014-4. DOI: 10.1109/AERO.2018.8396433. (visited on 06/11/2023).

[21] S. M. Albrektsen, T. H. Bryne, and T. A. Johansen, "Robust and Secure UAV Navigation Using GNSS, Phased-Array Radio System and Inertial Sensor Fusion," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*, Copenhagen: IEEE, Aug. 2018, pp. 1338–1345, ISBN: 978-1-5386-7698-1. DOI: 10.1109/CCTA.2018.8511354. (visited on 06/17/2023).

[22] Ø. K. Helgesen and E. F. Brekke, "Bearings-only tracking using factor graphs," [Online]. Available: https://folk.ntnu.no/edmundfo/fusion2022preprints/HelgesenBrekkeBearingFactor.pdf.

[23] J. Dong and Z. Lv, "MiniSAM: A flexible factor graph non-linear least squares optimization framework," *CoRR*, vol. abs/1909.00903, 2019. [Online]. Available: http://arxiv.org/abs/1909.00903.

[24] J. Solà, J. Deray, and D. Atchuthan. "A micro Lie theory for state estimation in robotics." arXiv: 1812.01537 [cs]. (Dec. 8, 2021), [Online]. Available: http://arxiv.org/abs/1812.01537 (visited on 04/07/2023), preprint.

[25] T. V. Haavardsholm, "A handbook in Visual SLAM," en, p. 95, 2021. [Online]. Available: https://github.com/tussedrotten/vslam-handbook.

[26] E. Brekke, *Fundamentals of Sensor Fusion*. Unpublished, 2022.

[27] J. Solà. "Quaternion kinematics for the error-state Kalman filter." arXiv: 1711.02508 [cs]. (Nov. 3, 2017), [Online]. Available: http://arxiv.org/abs/1711.02508 (visited on 02/09/2023), preprint.

[28] J. A. Farrell, F. O. Silva, F. Rahman, and J. Wendel, "Inertial Measurement Unit Error Modeling Tutorial: Inertial Navigation System State Estimation with Real-Time Sensor Calibration," *IEEE Control Systems*, vol. 42, no. 6, pp. 40–66, Dec. 2022, ISSN: 1066-033X, 1941-000X. DOI: 10.1109/MCS.2022.3209059. (visited on 06/27/2023).

[29] K. Gade, "Integrering av treghetsnavigasjon i en autonom undervannsfarkost," Norwegian Defence Research Establishment (FFI), Tech. Rep. FFI/RAPPORT-97/03179, 1997. [Online]. Available: https://www.navlab.net/Publications/.

[30] M. L. Sollie, T. H. Bryne, K. Gryte, and T. A. Johansen, "Reducing Ground Reflection Multipath Errors for Bluetooth Angle-of-Arrival Estimation by Combining Independent Antenna Arrays," *IEEE Antennas and Wireless Propagation Letters*, pp. 1–5, 2023, ISSN: 1536-1225, 1548-5757. DOI: 10.1109/LAWP.2023.3243166. (visited on 05/10/2023).

[31] F. Dellaert and M. Kaess, "Factor Graphs for Robot Perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017, ISSN: 1935-8253, 1935-8261. DOI: 10.1561/2300000043. [Online]. Available: http://www.nowpublishers.com/article/Details/ROB-043 (visited on 01/23/2023).

[32] F. Dellaert and GTSAM Contributors, *Borglab/gtsam*, version 4.2a8, Georgia Tech Borg Lab, May 2022. DOI: 10.5281/zenodo.5794541. [Online]. Available: https://github.com/borglab/gtsam).

[33] H. Martiros, A. Miller, N. Bucki, B. Solliday, R. Kennedy, J. Zhu, T. Dang, D. Pattison, H. Zheng, T. Tomic, P. Henry, G. Cross, J. VanderMey, A. Sun, S. Wang, and K. Holtz, "SymForce: Symbolic Computation and Code Generation for Robotics," in *Robotics: Science and Systems XVIII*, Jun. 27, 2022. DOI: 10.15607/RSS.2022.XVIII.041. arXiv: 2204.07889 [cs]. [Online]. Available: http://arxiv.org/abs/2204.07889 (visited on 06/27/2023).

[34] J. Sola, J. Vallve, J. Casals, J. Deray, M. Fourmy, D. Atchuthan, A. Corominas-Murtra, and J. Andrade-Cetto. "WOLF: A modular estimation framework for robotics based on factor graphs." arXiv: 2110.12919 [cs]. (Feb. 16, 2022), [Online]. Available: http://arxiv.org/abs/2110.12919 (visited on 02/06/2023), preprint.

[35] T. D. Barfoot, *State Estimation for Robotics*, 1st. Cambridge University Press, Jul. 31, 2017, ISBN: 978-1-316-67152-8. DOI: 10.1017/9781316671528. (visited on 04/11/2023).

[36] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental Smoothing and Mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008, ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2008.2006706. (visited on 01/20/2023).

[37] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, Feb. 2012, ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364911430419. (visited on 01/20/2023).

[38] T. Lupton and S. Sukkarieh, "Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 61–76, Feb. 2012, ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2011.2170332. (visited on 06/25/2023).

[39] 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, Feb. 2017, ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2016.2597321. (visited on 02/06/2023).

[40] L.-C. N. Tokle and E. Martens, *Dataset from TTK4250*, unpublished.

[41] E. Hjermstad, K. Steinsland, and E. S. Lie, "TTK4250 - Sensor Fusion. Graded Assignment 1," unpublished, Nov. 2, 2022.

[42] E. Martens, L.-C. N. Tokle, and E. Brekke, *Graded Assignment 1, Assignment 5, TTK4250 Sensor Fusion*, unpublished, Nov. 2022.

[43] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China: IEEE, May 2011, pp. 3607–3613, ISBN: 978-1-61284-386-5. DOI: 10.1109/ICRA.2011.5979949. (visited on 06/14/2023).

[44] F. Dellaert. "Factor Graphs and GTSAM." (2019), [Online]. Available: https://gtsam.org/tutorials/intro.html (visited on 06/14/2023).

[45] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[46] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020. DOI: 10.1038/s41586-020-2649-2.

[47] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, . Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[48] M. Hirsch, F. G. Nievinski, and M. Kleder, *Pymap3d*. [Online]. Available: https://github.com/geospace-code/pymap3d (visited on 07/01/2022).

[49] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[50] T. I. Fossen and T. Perez, *Marine systems simulator (MSS)*, 2004. [Online]. Available: https://github.com/cybergalactic/MSS.

[51] T. V. Haavardsholm, *Simple-factorgraph-example*. [Online]. Available: https://github.com/tussedrotten/simple-factorgraph-example/tree/main (visited on 07/01/2022).

[52] T. V. Haavardsholm, *Visgeom*. [Online]. Available: https://github.com/tussedrotten/visgeom (visited on 07/01/2022).

[53] T. V. Haavardsholm, *TTK21 lecture 3 exercises*, unpublished, 2021.

[54] K. Liu. "Finite Difference." (Aug. 5, 2020), [Online]. Available: https://rh8liuqy.github.io/Finite_Difference.html (visited on 07/03/2023).

[55] K. Gryte, M. Okuhara, O. K. Hasler, and P. Kvaløy, *Raudstein X8 003 Radionor Two Antenna Position Data*, unpublished, Oct. 8, 2020.

[56] C. Stachniss, "Iterative Closest Point: Point Cloud Alignment," 2020. [Online]. Available: https://www.ipb.uni-bonn.de/html/teaching/msr2-2020/sse2-03-icp.pdf.

[57] F. Gustafsson, *Statistical Sensor Fusion*, 2nd. Studentlitteratur, 2012.

[58] V. Agrawal. "Look Ma, No RANSAC." (Sep. 20, 2019), [Online]. Available: https://gtsam.org/2019/09/20/robust-noise-model.html (visited on 06/22/2022).

[59] Z. Zhang, "Parameter estimation techniques: A tutorial with application to conic fitting," *Image and Vision Computing*, vol. 15, no. 1, pp. 59–76, Jan. 1997, ISSN: 02628856. DOI: 10.1016/S0262-8856(96)01112-2. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0262885696011122 (visited on 06/22/2023).

Appendix A

Simulation Results

A.1 GNSS-aided INS

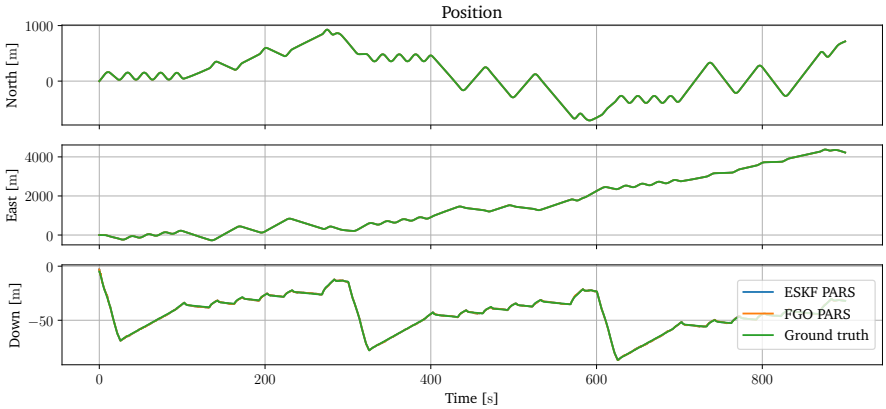


Figure A.1: Simulation result with GNSS. Position.

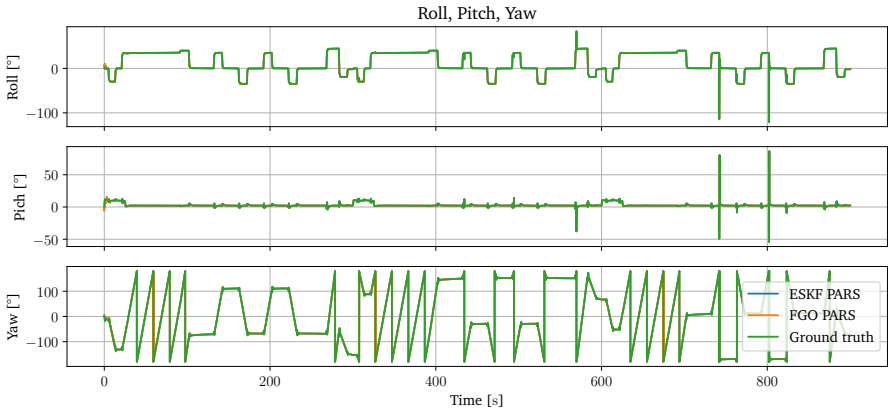


Figure A.2: Simulation result with GNSS. Attitude.

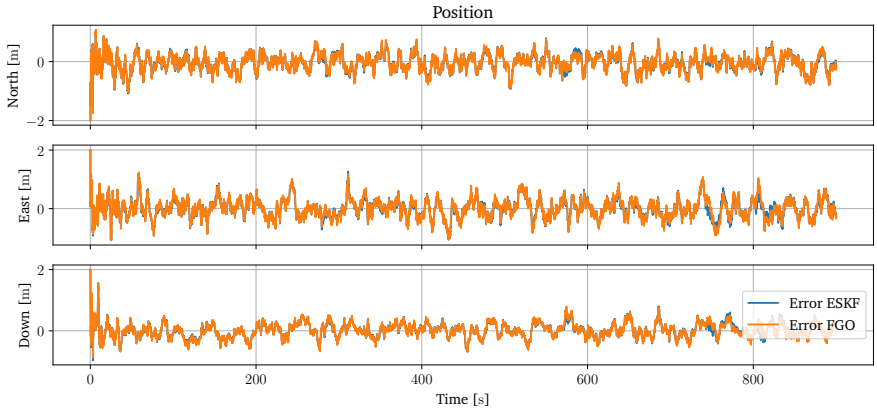


Figure A.3: Simulation result with GNSS. Error position.

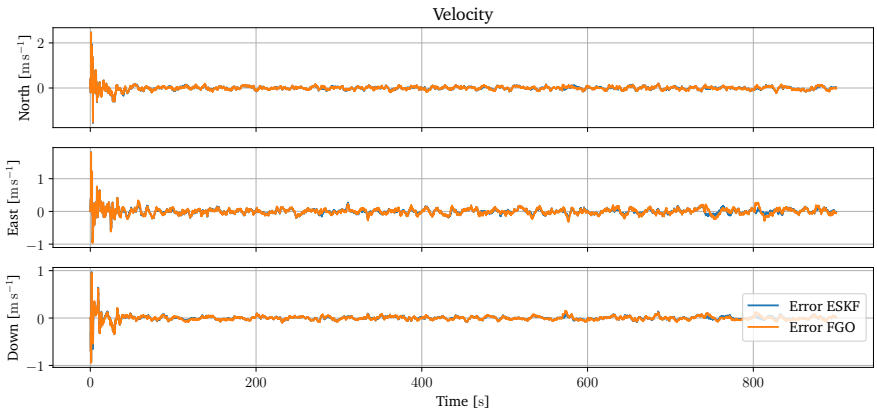


Figure A.4: Simulation result with GNSS. Error velocity.

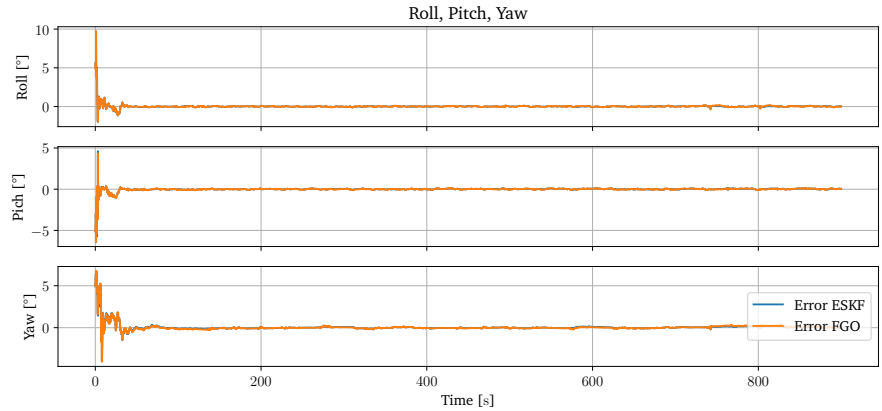


Figure A.5: Simulation result with GNSS. Error attitude.

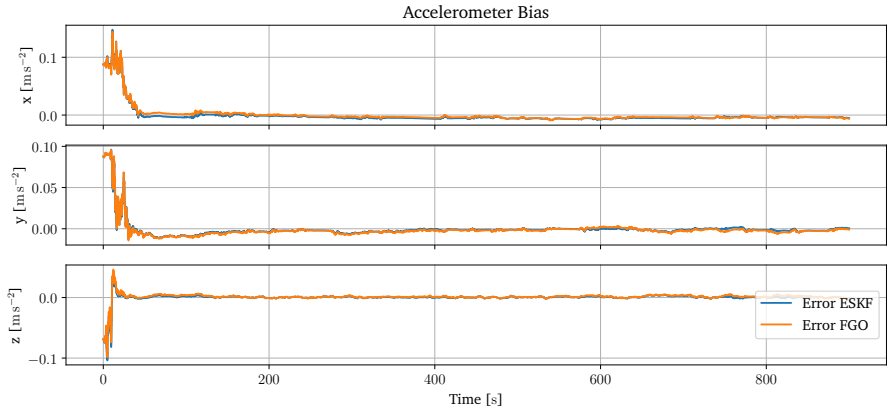


Figure A.6: Simulation result with GNSS. Error accelerometer bias.

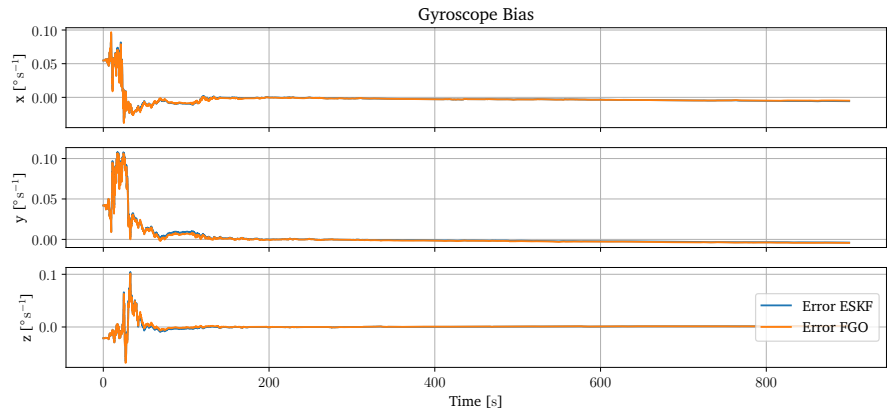


Figure A.7: Simulation result with GNSS. Error gyroscope bias.

A.2 PARS-aided INS using one ground radio at origin

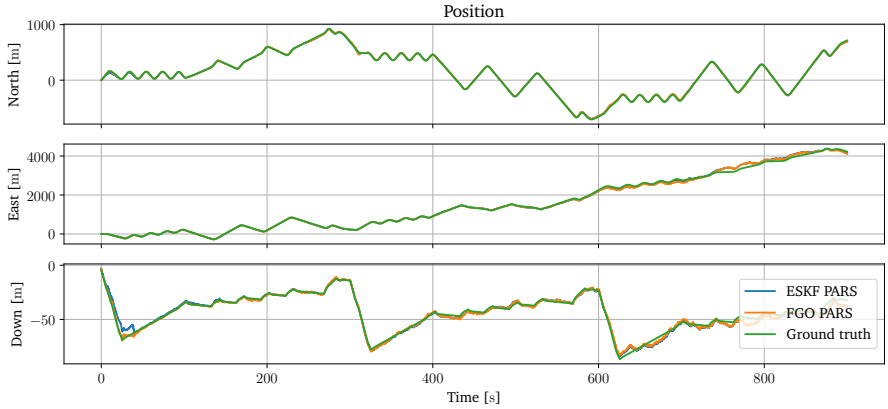


Figure A.8: Simulation result with one PARS. Position.

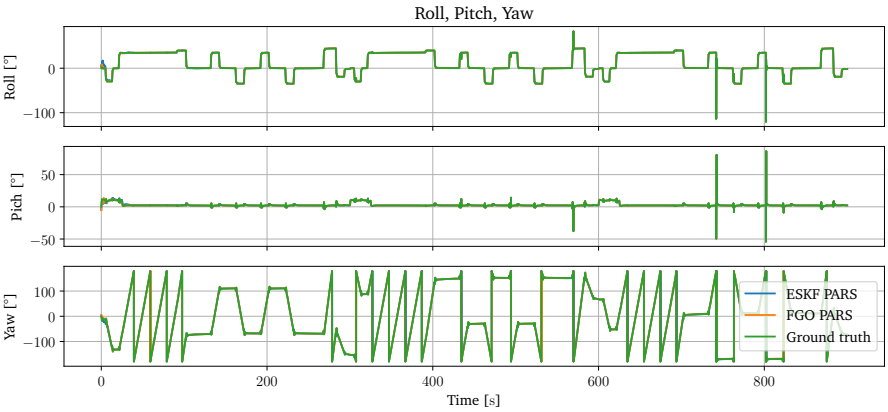


Figure A.9: Simulation result with one PARS. Attitude.

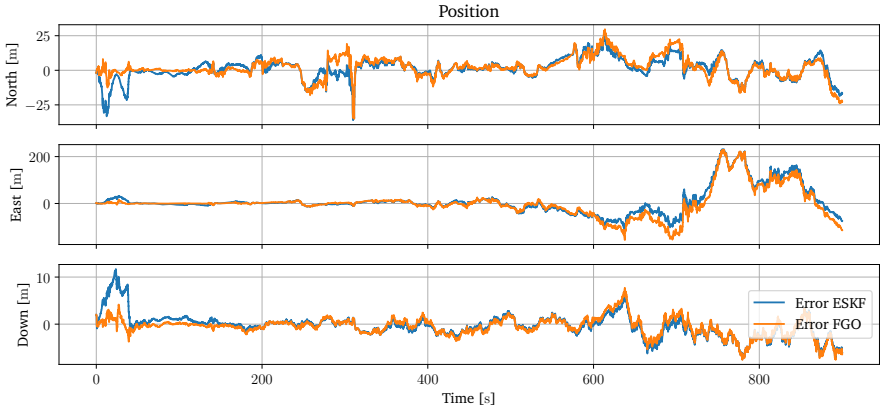


Figure A.10: Simulation result with one PARS. Error position.

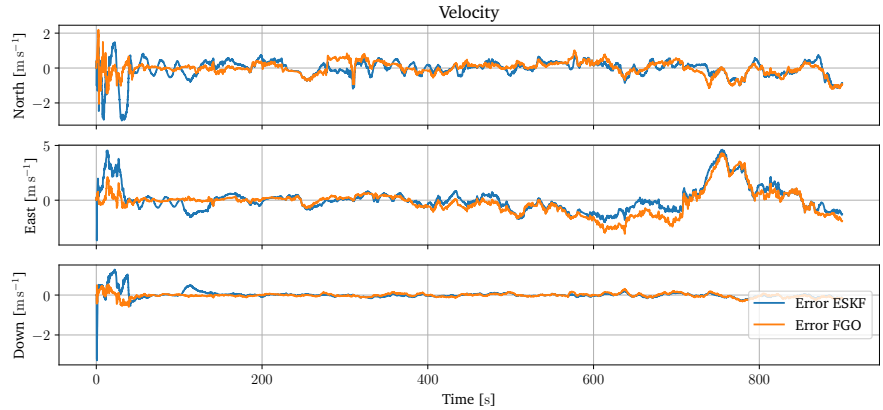


Figure A.11: Simulation result with one PARS. Error velocity.

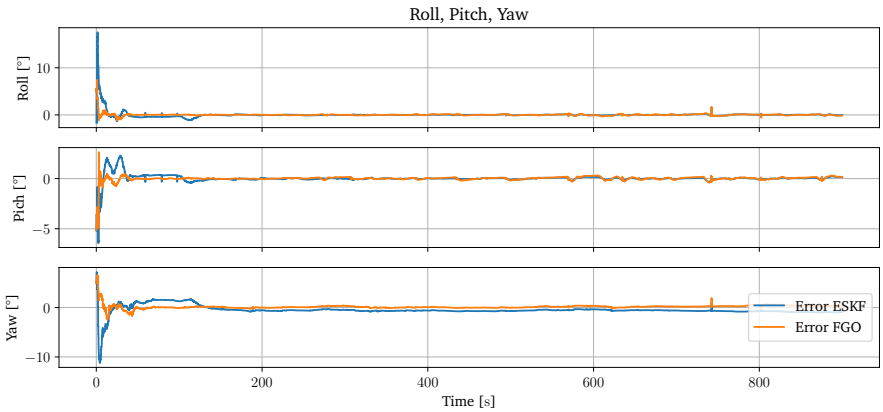


Figure A.12: Simulation result with one PARS. Error attitude.

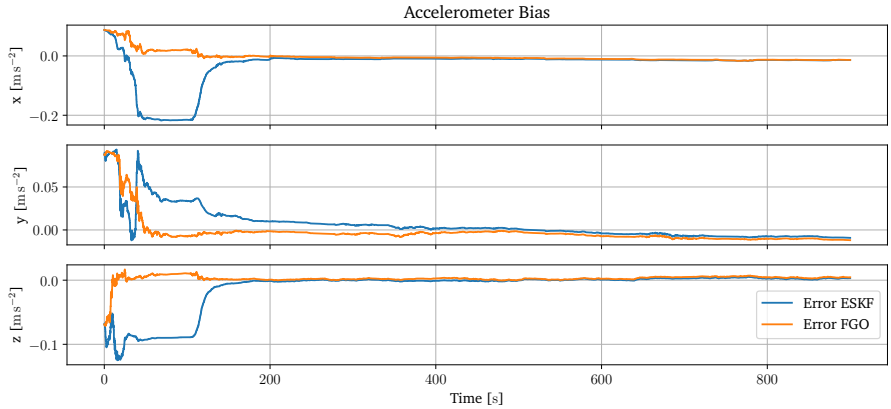


Figure A.13: Simulation result with one PARS. Error accelerometer bias.

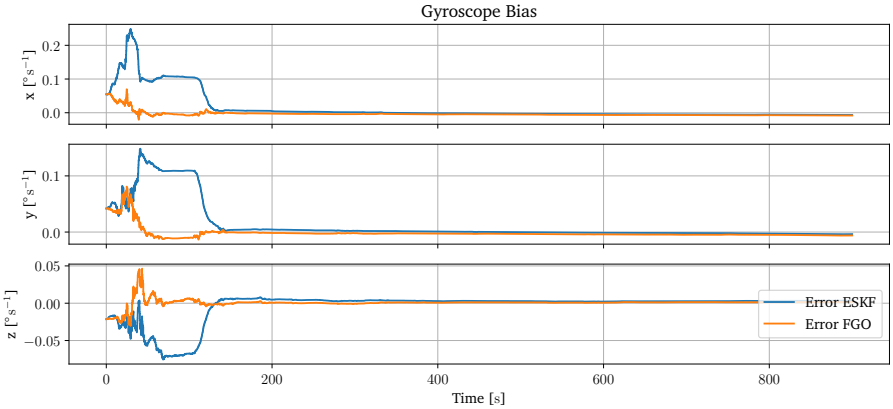


Figure A.14: Simulation result with one PARS. Error gyroscope bias.

A.3 PARS-aided INS using three ground radios

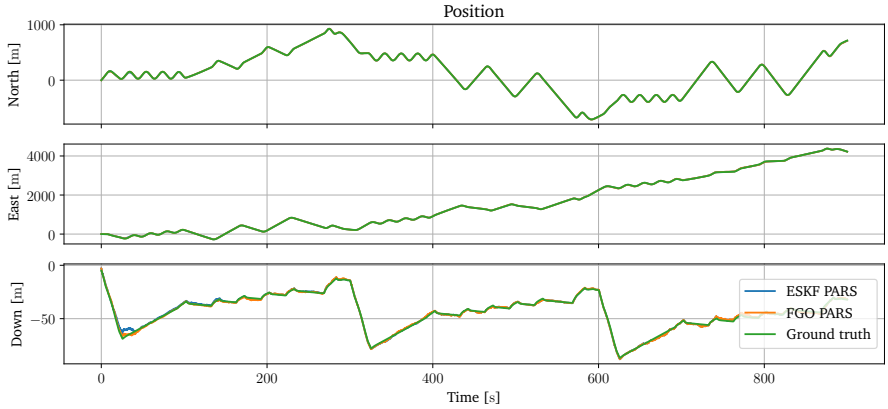


Figure A.15: Simulation result with three PARS. Position.

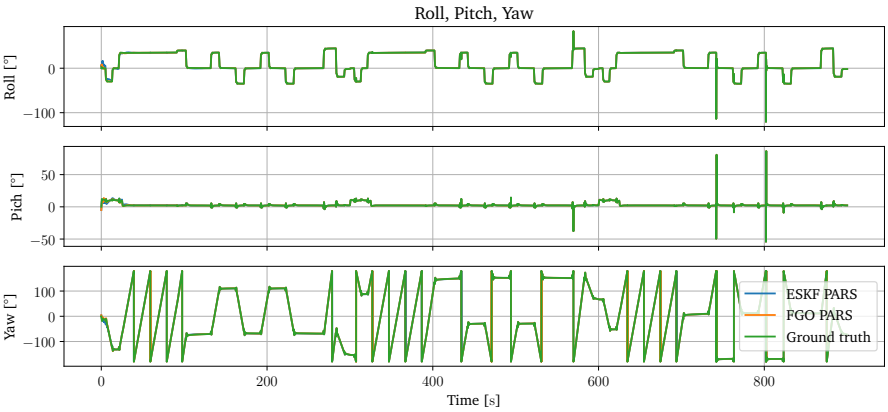


Figure A.16: Simulation result with three PARS. Attitude.

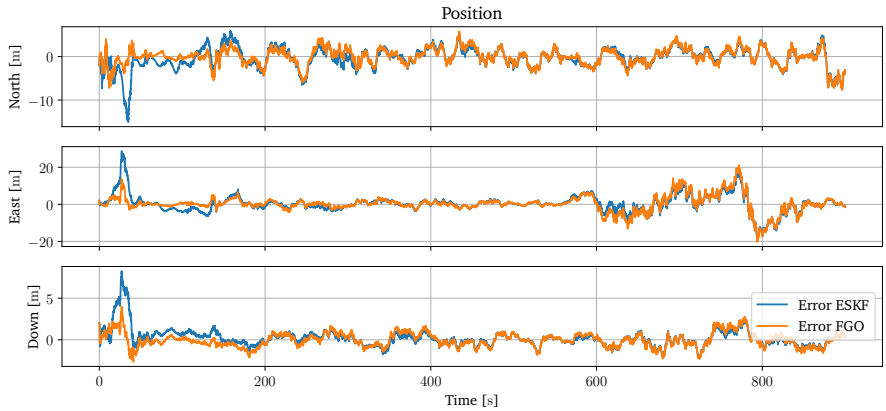


Figure A.17: Simulation result with three PARS. Error position.

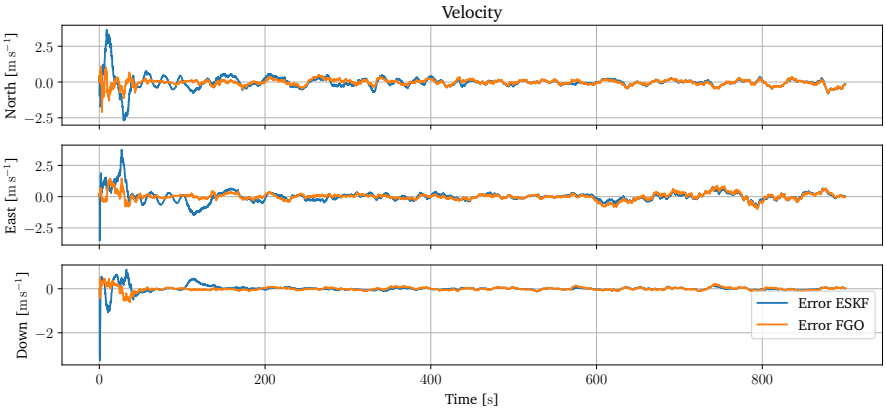


Figure A.18: Simulation result with three PARS. Error velocity.

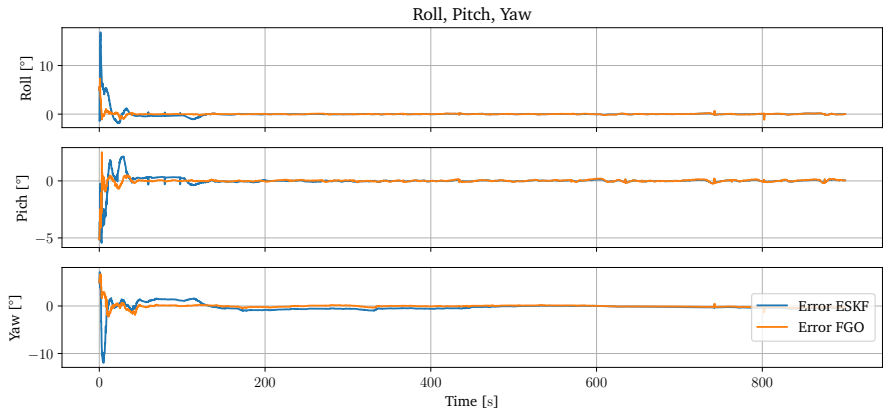


Figure A.19: Simulation result with three PARS. Error attitude.

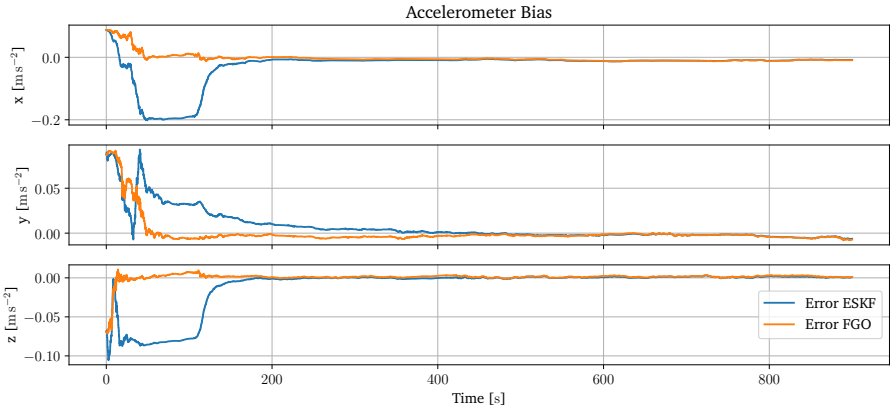


Figure A.20: Simulation result with three PARS. Error accelerometer bias.

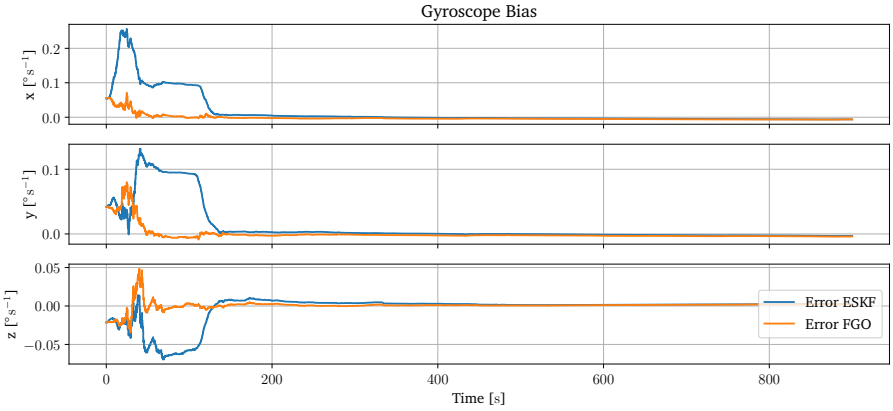


Figure A.21: Simulation result with three PARS. Error gyroscope bias.

Appendix B

Experimental Results

B.1 GNSS-aided INS

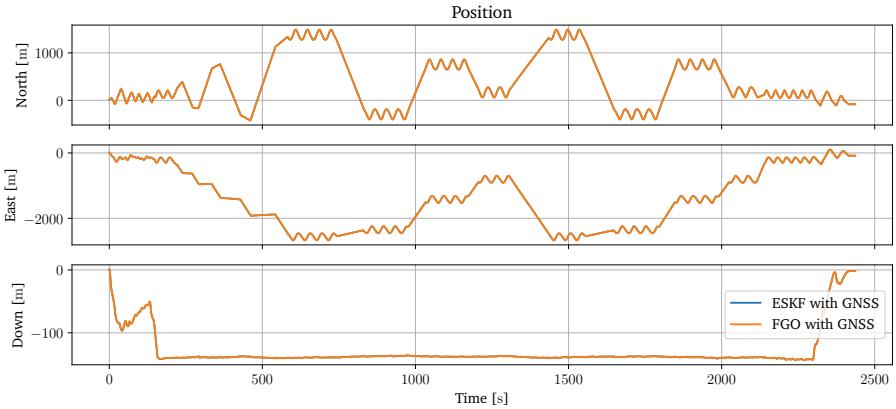


Figure B.1: Experimental result with GNNS. Position.

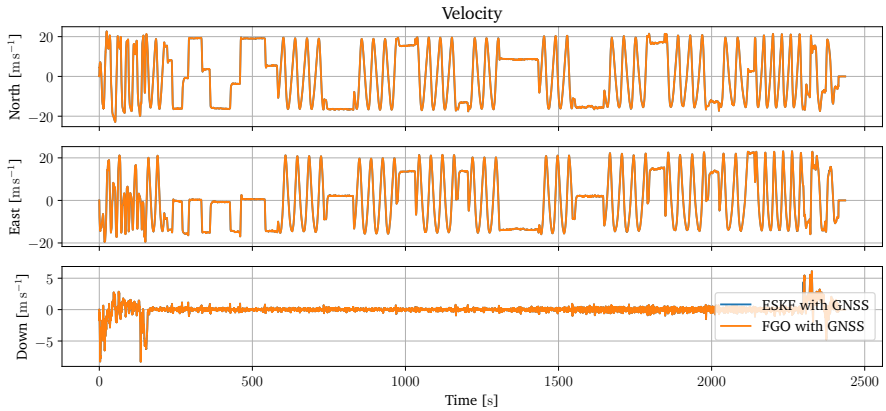


Figure B.2: Experimental result with GNNS. Velocity.

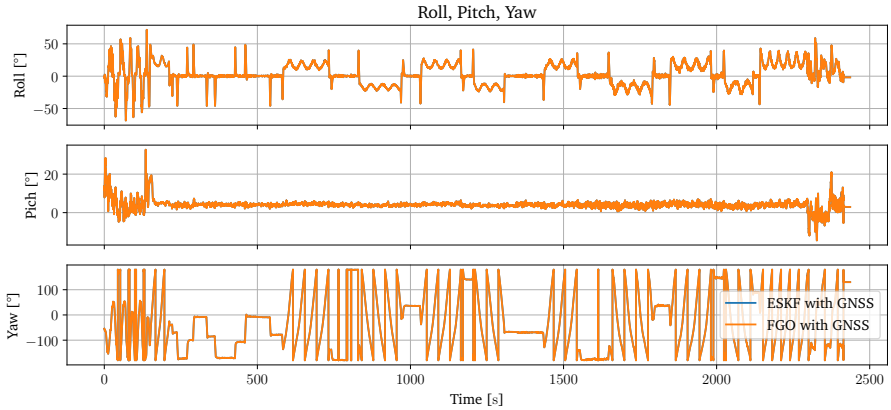


Figure B.3: Experimental result with GNNS. Attitude.

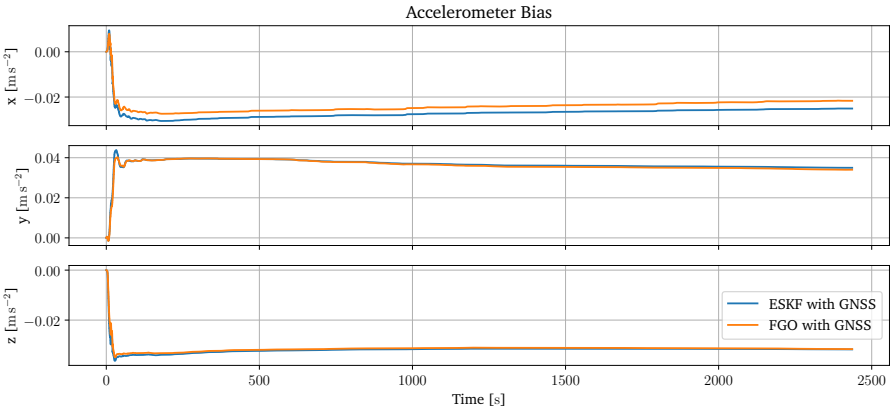


Figure B.4: Experimental result with GNNS. Accelerometer bias.

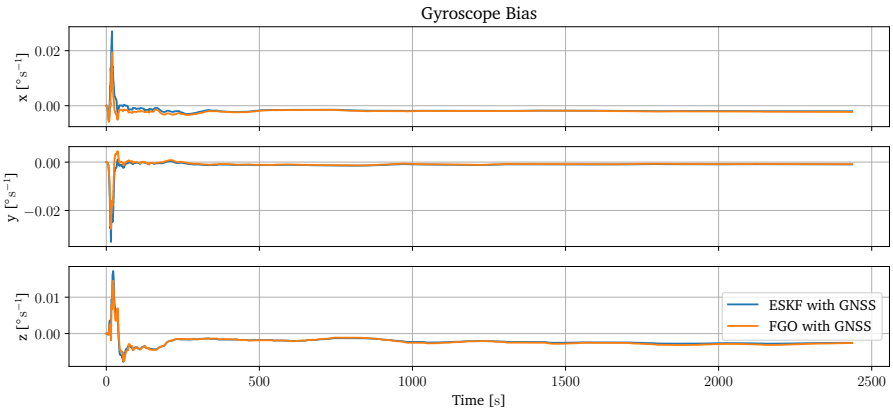


Figure B.5: Experimental result with GNNS. Gyroscope bias.

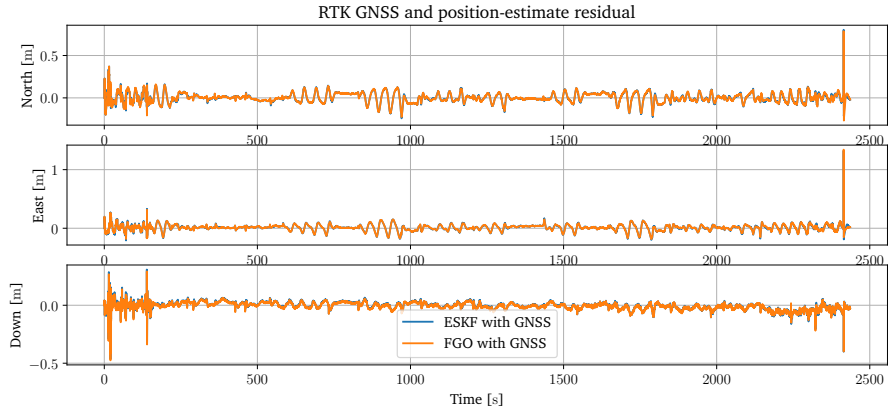


Figure B.6: Experimental result with GNNS. Position error.

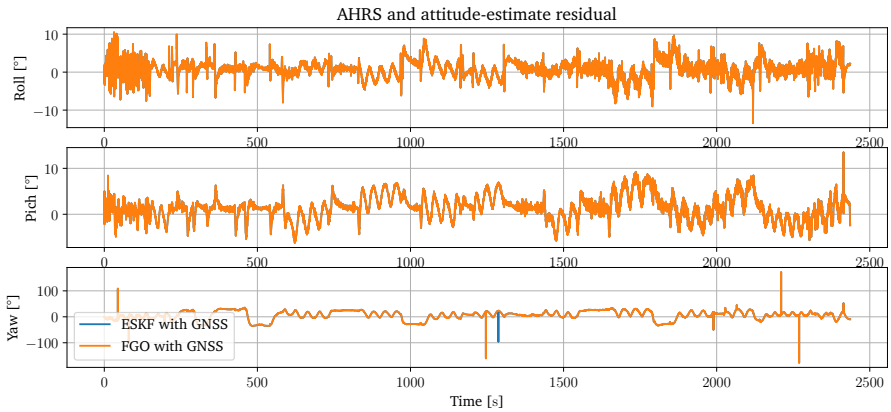


Figure B.7: Experimental result with GNNS. Attitude error.

B.2 PARS-aided INS using two ground radios

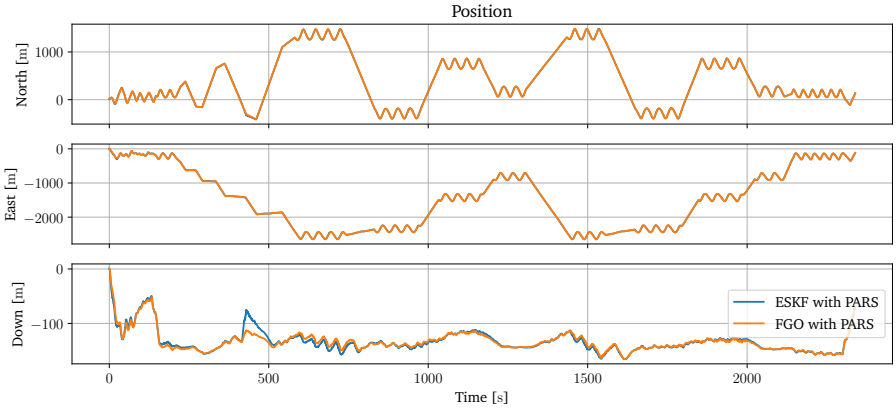


Figure B.8: Experimental result with PARS. Position.

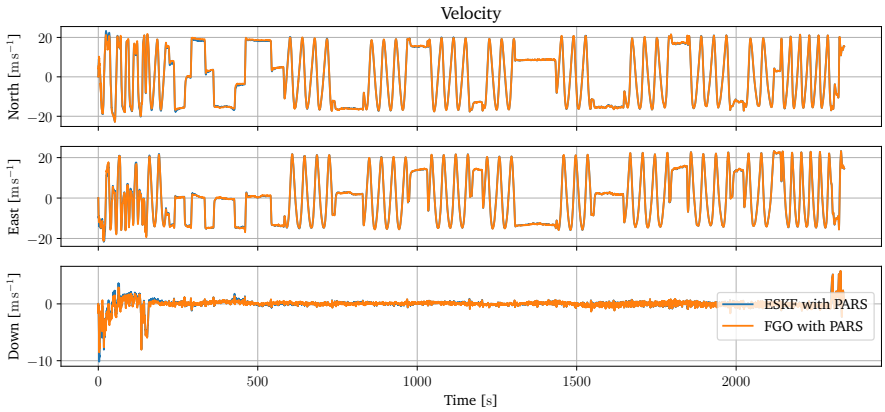


Figure B.9: Experimental result with PARS. Velocity.

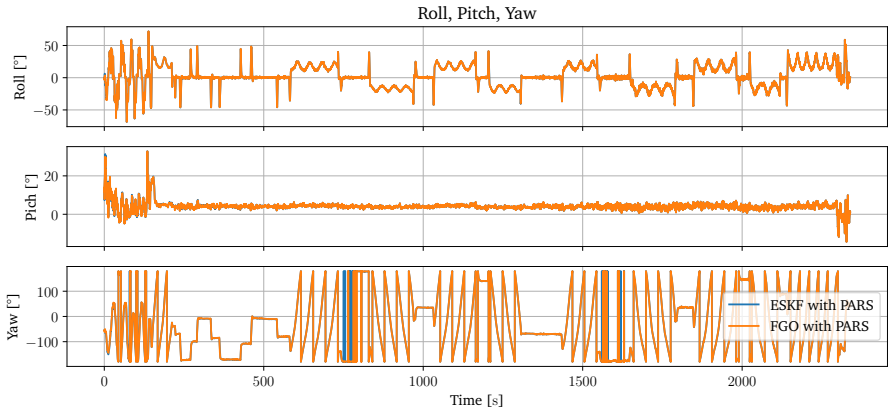


Figure B.10: Experimental result with PARS. Attitude.

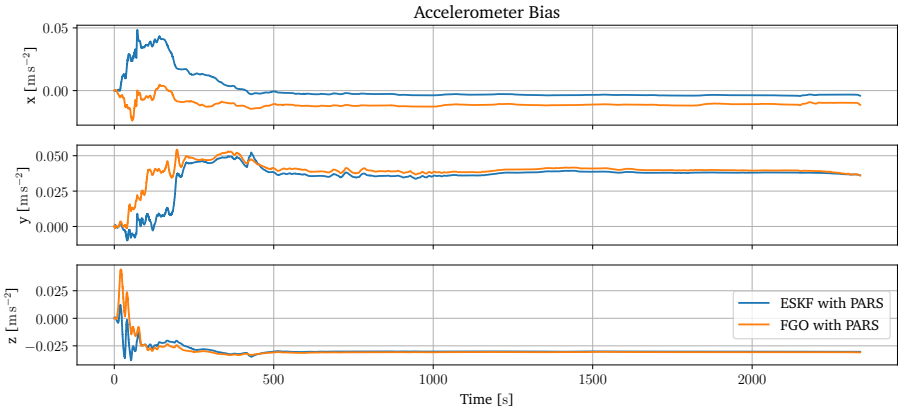


Figure B.11: Experimental result with PARS. Accelerometer bias.

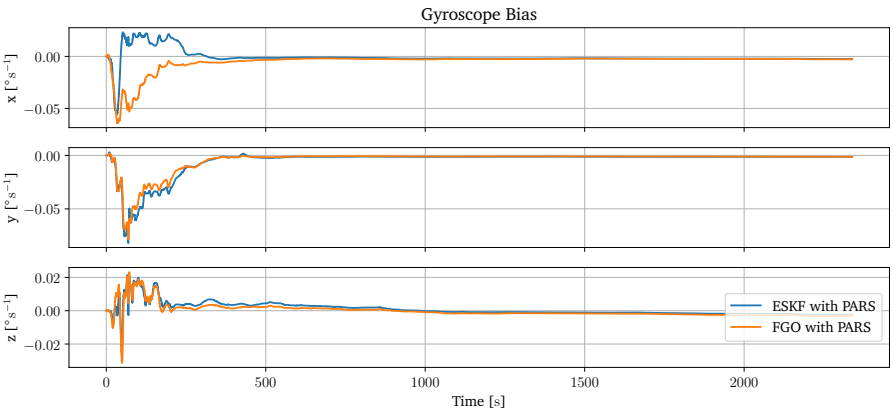


Figure B.12: Experimental result with PARS. Gyroscope bias.

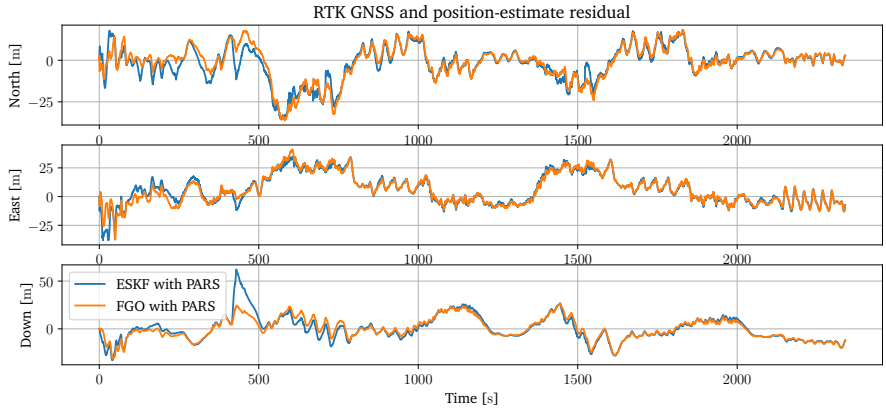


Figure B.13: Experimental result with PARS. Position error.

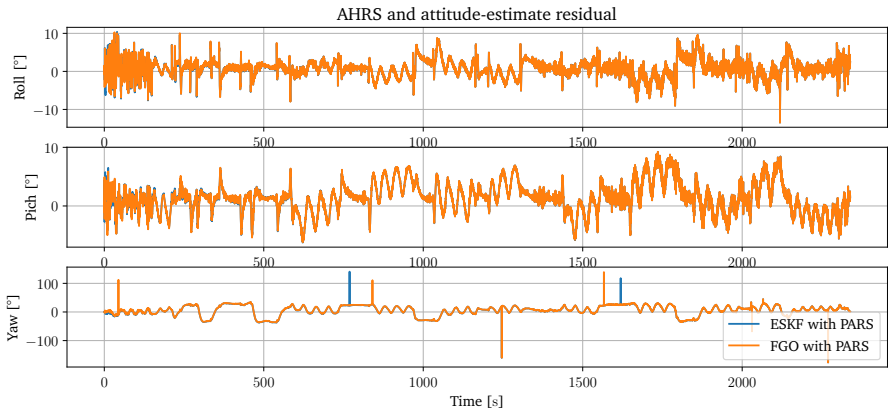


Figure B.14: Experimental result with PARS. Attitude error.

Appendix C

Online calibration

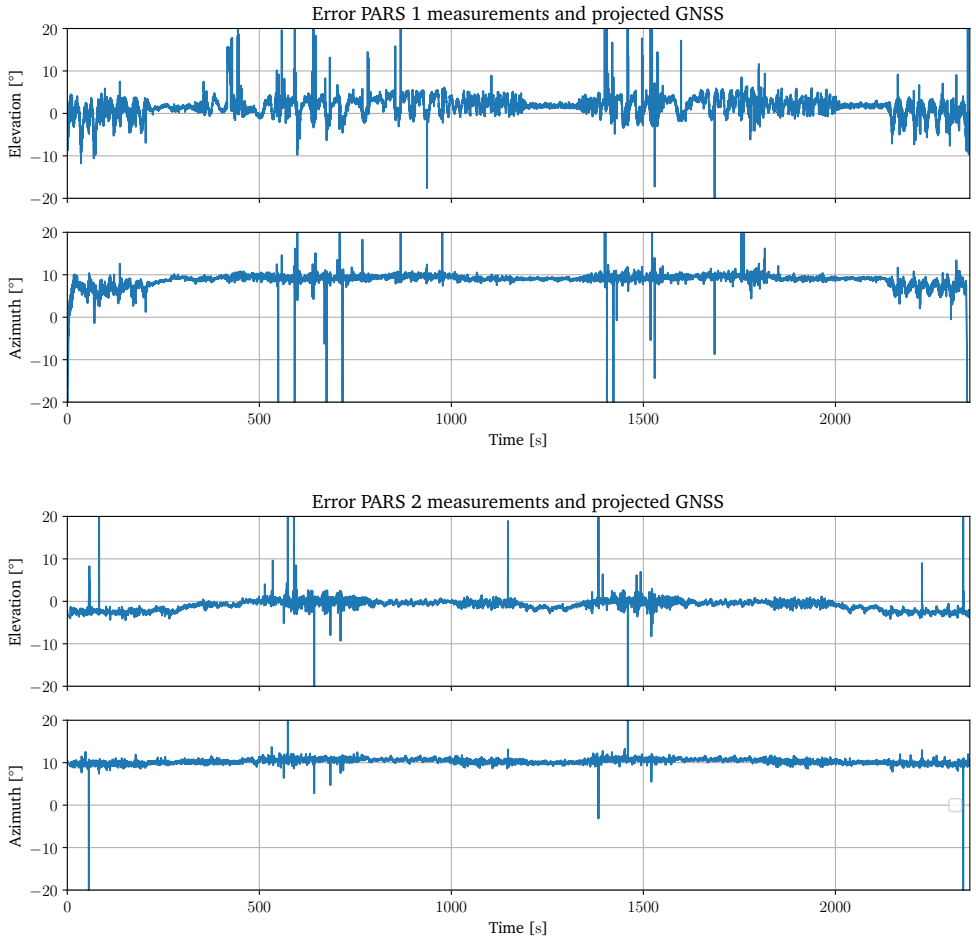


Figure C.1: Error between PARS measured angels and GNSS measurements projected to elevation and azimuth using compass and GNSS measured pose in. Clearly there is a significant calibration error.

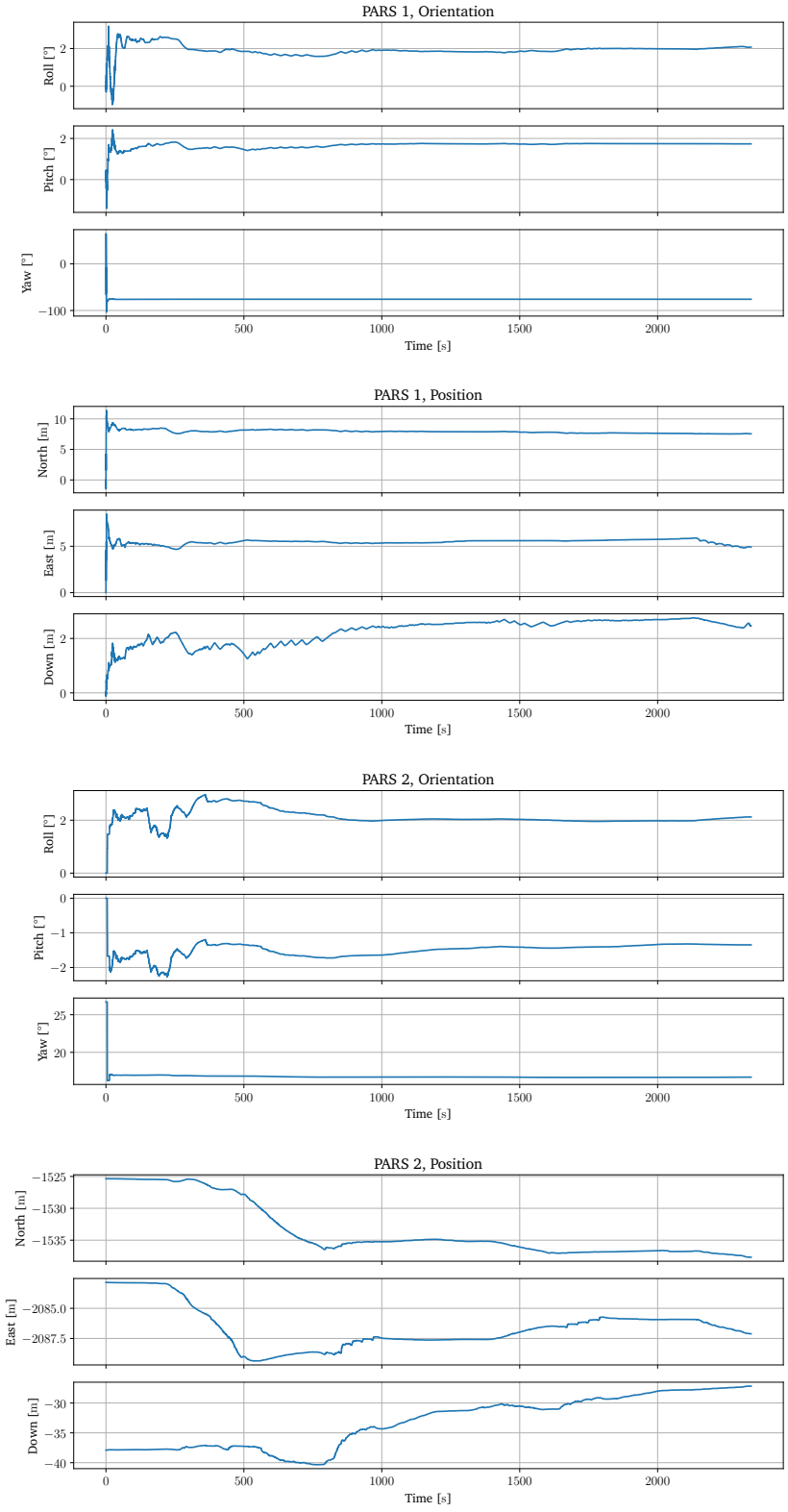
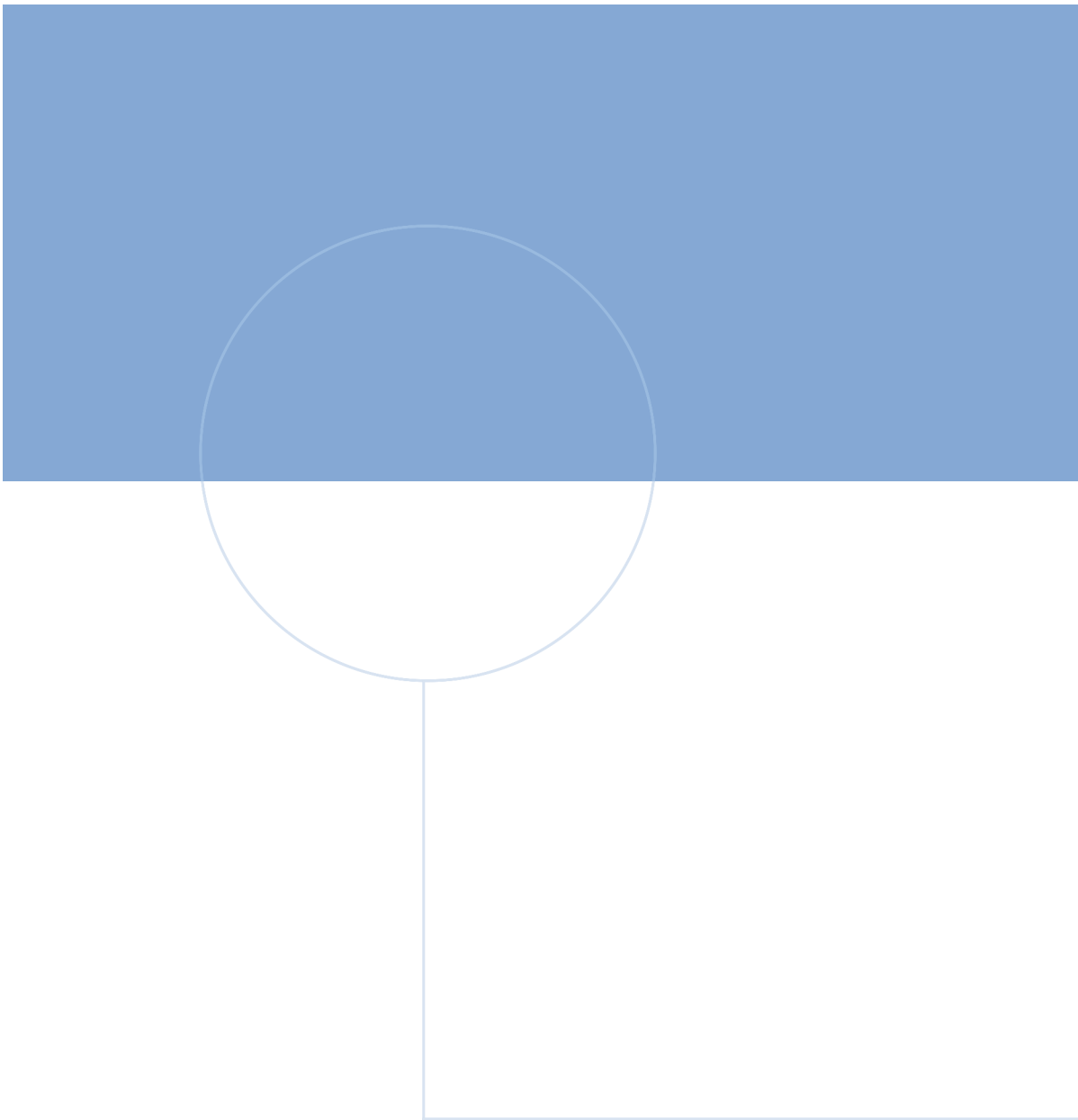


Figure C.2: Online calibration full run results. PARS position and orientation. Here th



NTNU

Norwegian University of
Science and Technology