Enhancing GNSS Precision for Mobile Devices with Sensor Fusion Techniques: A Case Study on eBike Tracking Using State Estimation

Richard Byström

William Sjödin

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6212
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

© 2023 by Richard Byström & William Sjödin. All rights reserved.
Printed in Sweden by Tryckeriet i E-huset
Lund 2023

Abstract

Electric bicycles have over time become a common method of transportation. With a rapidly increasing user base and expensive prices, there is also an increasing demand for safety and insurance. Bike safety can be used to notify the bicycle owner of a potential theft, or to locate the position of the bike when it is lost. Improving the autonomous vehicle's position tracking when its location is unknown to the owner, simplifies the search for a lost or stolen bike.

With the use of an accelerometer and gyroscope as input, coupled with a Global Navigation Satellite System, a prediction algorithm, or filter, was developed to predict the trajectory. Three different dynamical models for the filter were tested for robustness and optimization of filter parameters to yield desired results. An Extended Kalman Filter was used for the predictions, while the tested dynamic models were of linear, first-order and second-order types. The simulations used for the model evaluation utilized four different noise models. While the Inertial Measurement Unit contained known noise, the GNSS noise remained unknown.

When the tests with simulations indicated which models had the best performance, the filter was used on data from real-world measurements. Calibration was made on the Inertial Measurement Units' inner coordinate frame to get position estimates for comparisons with satellite points.

In some cases, the lone use of a GNSS for position tracking proved to be the best, while in other cases the filter output had a higher accuracy of predicting the position correctly. On average, the second-order model proved to have the best performance, concluding that it also was the most robust model. The model had an average error of 1 meter from the true position at best, and 1.4 meters at worst.

Acknowledgements

We would like to thank several individuals who have not only helped us with the completion of this thesis but also as a whole during our studies. First, we want to show our gratitude to our faculty supervisor Yiannis Karayiannidis from the Department of Automatic Control at Lund University, who gave us helpful advice during the thesis. Further, we would like to thank you for all of the corrections and the proofreading of the report, and also giving us helpful guidance. Secondly, we would like to direct our thanks to the examiner of this thesis, Björn Olofsson.

Moreover, we want to thank the team at Bosch, specifically our supervisors Martin Heyden and Stefan Noll. Not only have they been very helpful for our work, but also interested in how the thesis could be further developed in the future. Lastly, we want to thank Anders Svensson and all of the eBike team for allowing us to do this thesis with them.

Contents

Acronyms

AC - Alternating Current
AI - Artificial Intelligence
AKF - Adaptive Kalman Filter
DC - Direct Current
DCM - Direct Cosine Matrix
DNN - Deep Neural Network
eBike - Smart Electric Bicycle
EKF - Extended Kalman Filter
GNSS - Global Navigation Satellite System
IMU - Inertial Measurement Unit
MAPE - Mean Absolute Percentage Error
ML - Machine Learning
MSE - Mean Squared Error
PDF - Probability Density Function
PF - Particle Filter
RMSE - Root Mean Squared Error
UKF - Unscented Kalman Filter

1

Introduction

This chapter aims to introduce the subject of the thesis, and reasoning toward the relevance of position tracking. Next, the problem is formulated with details of what type of data is used. Lastly, the thesis structure is presented along with some limitations.

1.1 Opening

For over 20 years, investigations have been made on the reliability of self-driving vehicles, but also on how they can be improved. Humans have had a lower trust in the reliability of self-driving vehicles, approved by a 2020 poll, where 48% of Americans say that they “would never get in a taxi or ride-share vehicle that was being driven autonomously” [PAVE, 2020]. 60% of Americans say they would have greater trust in autonomous vehicles if they “understood better how the technology works”. Lastly, 75% of Americans who own a vehicle with an advanced driver-assistance system are intrigued to see what new safety features will be implemented in future cars. Development in autonomous vehicles include adaptive cruise control, lane keep assist, self-park as well as traffic jam assist. Further, the NHTSA anticipates that features like highway autopilot will be available by 2025 [Sleight, 2021].

Self-driving cars use sensor fusion in different areas to ensure safety of the passenger, which is a method to combine signals from multiple sources. These signals are integrated together to instead yield a single signal of information. The algorithms for sensor fusion can be classified into three different categories: fusion based on least-squares techniques, fusion based on probabilistic models as well as intelligent fusion. The first techniques includes Kalman filters, optimal theory as well as regularization. Probabilistic methods are often defined to include Bayesian learning, evidence theory and recursive operators. Lastly, intelligent learning groups methods such as neural networks, fuzzy logic and genetic algorithms [Sasiadek, 2002].
Navigation, guidance and control of vehicles require information from a large

number of sources. Regardless of if a vehicle is self driven, or just interprets the surrounding environment, a combination of cameras and radars are often used. Self-driving cars use regular short-range cameras, long-range cameras, LiDAR sensors as well as a short/medium-range radar to get a better understanding of its surroundings, visible in Figure 1.1 [Yeong et al., 2021].

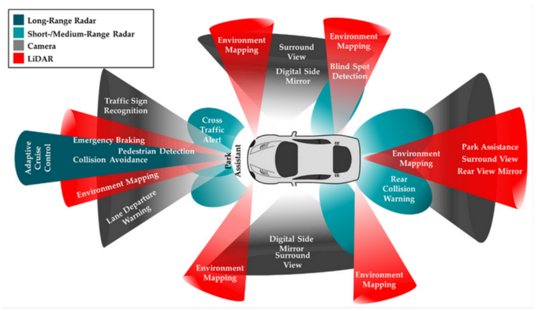

Figure 1.1 An example of how a self-driving car uses a combination of radars and sensors to get a full understanding of the surrounding environment. Red areas indicate the LiDAR coverage, grey areas show the camera coverage, blue areas display the short/medium-range radar coverage and the green area shows the long-range radar coverage [Yeong et al., 2021].

Sensor fusion can also be used to track different vehicles, but continuous information about the location of the car must be accurate. Position tracking with a GPS is one of the most common localization technologies used in current times [Mapscaping, 2023]. Over the years, the accuracy of GPS measurements has been greatly increased, while the price of the technology has been reduced. However, there are still investigations made on how the localization accuracy can be improved further. The Global Navigation Satellite System which includes GPS, the European Union's Galileo, Russia's GLONASS, as well as more countries are currently working on their own GNSS navigation solutions. The U.S. Government is currently working on launching new GPS satellites which aims to improve the accuracy and availability for all users. In conclusion, the aim is to improve the accuracy of position tracking using GNSS signals [Geotab-Team, 2020].

Another practical area of application for position tracking is for localization of lost devices that have been subjected to theft. There are several ways to prevent the theft of bicycles, some more expensive than others. Installation of high-security locks might be effective, as would the simplicity of parking the bike next to other expensive bikes. If the bike has been stolen, however, none of those will be beneficial. One could opt to have an insurance for the bike, *or* use tracking devices to find where the bike has been taken after the theft [Toll, 2022]. A simple GPS tracker could potentially solve the problem, but the position tracking is not always as precise as need be. Combining the GPS with another sensor in a fusion framework can improve the accuracy, resulting in a higher performance position tracking. This means, that using a combination of GNSS measurements together with an Inertial Measurement Unit output, the position tracking can be improved.

The main difference between position tracking of lost devices compared to self-driving cars is that computational speed is not as much of an issue for the localization of misplaced equipment. In traffic, a crash can happen in the span of milliseconds which means that an almost instant reaction from the driver, or in this case, the computer, is needed to prevent that such an incident occurs in the first place. More computational-heavy estimation algorithms can be used for remote localization of stolen devices since usually the difference between seconds is not an issue in this scenario.

1.2 Problem formulation

To improve the accuracy and minimize measurement noise for the anti-theft device, sensor fusion of two navigational sensors in an eBike will be investigated. The two sensors are an Intertial Measurement Unit and a Global Navigational Satellite System receiver. The IMU is an electronic device that measures the force, angular velocity and the orientation of a body. This is measured by a combination of accelerometers, gyroscopes, and magnetometers [Vectornav, 2023]. GNSS refers to an array of satellites providing statistics from space that transmit positioning and timing data to GNSS receivers. The data is then used to determine the location of the receivers [EUASP, 2023].

Firstly, the idea is to use the local data from the IMU together with the far-reaching GNSS data and, with the help of an estimation algorithm, to find a more accurate travelling path in comparison to the true path taken. If done correctly, the estimated data should more accurately represent the path that the eBike ventured through after it was stolen compared to only using the GNSS as a source for localization.

Further, the thesis aims to evaluate the performance of the estimation algorithm and identify properties that results in accurate position tracking. In other words, we want to find ways to determine whether the algorithm is a better alternative compared to the lone usage of a GNSS, or if there are ways to further increase the accuracy of the position tracking. It is subjective to decide what a good enough accuracy may be, but the aim is to have a better accuracy than the GNSS. Further, an average accuracy of the predicted position being located within 2 meters is strived for, as a 2 meter proximity simplifies the search of a lost eBike.

1.3 Thesis structure

The process of the thesis is to implement a prediction algorithm which increases the accuracy of the position tracking in a GNSS. Further investigations are conducted to enhance the accuracy of the position tracking. The thesis begins with a background of everything needed to understand how the project was developed, with information about the mathematics and sensors used. Next, the method is described, with a red thread connecting the calibrations, models and optimization techniques together. Next, the results are presented, starting off with a description of how the performance is evaluated, followed with detailed performance results for both simulations and real-world tests. Lastly, the results are discussed, with ideas for future work to improve the performance even further.

1.4 Limitations

As mentioned previously, this thesis revolves around improving the position tracking of an eBike, regardless of the program running offline or online. The algorithm is therefore implemented without any computational efficiency in mind, as well as memory storage.
Further, the thesis limits itself to use *one* type of prediction algorithm, and instead investigate different model dynamics instead. The investigation of different model dynamics will be limited to three models, while the number of noise models used in the generation of the simulated GNSS noise is limited to four. The different model dynamics all represent a physical way of viewing how the position changes over time, and while more models could be included, they would require measurements of which the sensor used during the thesis could not require. Four noise models were chosen to give a broad enough spectrum for evaluations, leading to a more secure conclusion of which model dynamic had the best consistency.

1.5 Individual contributions

The authors of this thesis have contributed all-in-all equally, with each focusing on respective areas. Richard investigated different theoretical approaches to the thesis, for implementations, testing and evaluations, while William worked with development of the algorithm. However, both parties discussed the theory and the development diligently during the thesis. Further, Richard ran tests and evaluations on simulated data, while William focused on the real-world data optimizations. The discussions were made with equal contributions, and the respective results lead to the same conclusion. To repeat ourselves, even though the authors had different tasks, both parties still contributed in every step of the thesis.

2

Background

This section aims to give an introduction to what is used to combine an Inertial Measurement Unit with Global Navigation Satellite Systems in a sensor fusion framework. First, the basic mathematics is described, which is used to develop models and algorithms. This includes a description of how systems are represented in a linear and non-linear way, how to linearize non-linear systems as well as statistical methods to evaluate performance. Next, different ways to compute rotation are described, and used in pre-processing of the IMU data. This includes a description of the different computations and a comparison between them. The fundamentals of the sensors used are then described. Next, the main algorithm is described thoroughly, as well as the reasoning for the choice of the algorithm. Different models of noise are also described and used to increase the performance on real-world data. A random search algorithm is briefly discussed, used to find optimal hyperparameters for the prediction algorithm. Lastly, some previous articles of relevance are brought up.

2.1 Basic mathematics

Representation of systems

In control theory, a system is commonly modeled with the help of differential equations. These equations are a way to describe how a function is related to the derivative of the states, and they are commonly used for physical quantities.
A basic differential equation has the form of:

$$\frac{dx}{dt} = f(x), \tag{2.1}$$

where if $f(x)$ is a linear function, then the system is called linear. An example of this is $f(x) = 3x + 1$. However, if this is not the case, the system is called non-linear. While linear theory describes many physical systems well, some dynamics can cause non-linear effects. This could for example happen when a physical quantity achieves a limit and is therefore saturated. In this case, the function $f(x)$ is

strictly non-linear.
Another case is when there are several variables involved in the system. These are frequently occurring in several applications, control engineering being one. In this case, the variables are u, which is the input of the system, x, which is the system state, and lastly, y, which is the process output. These equations take the form of

$$\begin{cases} \frac{dx}{dt} = f(x,u) \\ y = g(x,u) \end{cases} \tag{2.2}$$

where $f(x,u), g(x,u)$ can be linear, or non-linear. In a linear case, they are commonly rewritten as

$$\begin{aligned} \tfrac{dx}{dt} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2.3}$$

where A,B,C,D are matrices which describe the dynamics of a system.
These last equations describe a continuous-time model of a control system. However, the physical system can still be continuous in time, but the model can be described in discrete time, and for that reason, there are also equations describing a time-discrete system:

$$\begin{aligned} x_{k+1} &= \Phi x_k + \Gamma u_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{2.4}$$

where Φ, Γ are the discrete-time versions of the former A and B under the assumption of zero-order hold sampling, computed by [Årzén, 2014]:

$$\begin{cases} \Phi = e^{Ah} \\ \Gamma = \int_0^h e^{As} B ds \end{cases} \tag{2.5}$$

while C and D remains the same for a discrete-time system. h denotes the sample time of the system. The key difference between a time-discrete system is the requirement of a known sample rate, since there is not a continuous stream of values [Glad and Ljung, 1997].

Linearization

More often than not, an interest lies within maintaining a constant level of some quantities. At this equilibrium point, the system is approximated to behave in a linear way, which means that linearization around this point is valid. Some algorithms used in control theory only work for linear systems, such as the computation of transfer functions and regular Kalman filters. To be able to use these methods on non-linear systems, linearization is needed. Further, a linear system is easily understandable in its dynamics [Glad and Ljung, 1997].

Consider the system in equation (2.2). The process of linearization follows these steps [Glad and Ljung, 1997, Chapter 11]:

- Find the equilibrium point chosen to linearize around, $f(x_0, u_0) = 0$,
- The linearization of the system at this point is

$$\begin{aligned}\Delta\dot{x} &= A\Delta x + B\Delta u \\ \Delta y &= C\Delta x + D\Delta u\end{aligned} \tag{2.6}$$

where $\Delta x = x - x_0$, $\Delta u = u - u_0$ and $\Delta y = y - y_0$. The matrices A, B, C and D are given by the derivatives of the respective function at the equilibrium point, i.e.,

$$A = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}, B = \begin{pmatrix} \frac{\partial f_1}{\partial u} \\ \vdots \\ \frac{\partial f_n}{\partial u} \end{pmatrix} \tag{2.7}$$

$$C = \begin{pmatrix} \frac{\partial g}{\partial x_1} & \cdots & \frac{\partial g}{\partial x_n} \end{pmatrix} \text{ and } D = \frac{\partial g}{\partial u} \tag{2.8}$$

Statistics

Probability is a way to study uncertainty. It can be thought of as a degree of belief about an event, i.e., how likely it is that an event will happen. Probability distributions are used to build other concepts, like probabilistic modeling, graphical models, and model selection [Deisenroth et al., 2021]. Further, process disturbances as well as measurement noise are mainly stochastic, meaning that they have to be modeled with that in regard [Lindgren et al., 2013].
Let X be a random or stochastic variable and $f(x)$ the density function of its distribution. Then, the following:

$$P(a < X \leq b) = \int_a^b f(x)dx \tag{2.9}$$

defines the probability that X takes any value within $a < X \leq b$ [Hofmann-Wellenhof et al., 2008, Chapter 7.3]. The mean and variance follow as:

$$\mu = \int_{-\infty}^{+\infty} xf(x)dx \tag{2.10}$$

$$\sigma^2 = \int_{-\infty}^{+\infty} (x-\mu)^2 f(x)dx \tag{2.11}$$

or with time-discrete values:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.12}$$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^2 \tag{2.13}$$

with N being the number of samples. Note that the square root of the variance, i.e., σ is the standard deviation. These variables can be used to compute a confidence interval, which is a range of estimates for an unknown variable, meaning that it is a probability of the real unknown variable existing within that interval.
The covariance is defined as:

$$\operatorname{cov}(X,Y) = \sigma_{XY} = \sum_{(x,y)\in S} f(x,y)(x-\mu_X)(y-\mu_Y), \tag{2.14}$$

with S defining the whole set of values that (x,y) can take, and if X and Y are uncorrelated, $\operatorname{cov}(X,Y) = 0$. For the discrete-time case, it is written as

$$\operatorname{cov}(X,Y) = \sigma_{XY} = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu_X)(y_i-\mu_Y). \tag{2.15}$$

The covariance of a variable with itself is

$$\operatorname{cov}(X,X) = V[X] = \sigma_X^2, \tag{2.16}$$

i.e., simply the variance [Park, 2018].

The basics of machine learning utilize that there is a prediction of the output given a certain input. From a statistical perspective, these predictions are based on the conditional class probabilities

$$P(y=m|x) \tag{2.17}$$

where y is the output, and x is the input. This relation is described as the probability for class m given that the input x is known. Looking at just $P(y|x)$, there is an implication that the class label y is thought of as a random variable [Lindholm et al., 2022].

Statistical methods are also a way to describe systems of equations, similar to those in Section 2.1. Consider the non-linear system described in Equation (2.2), with the exception that there is no input, i.e.,

$$\begin{aligned} x_{k+1} &= f(x_k, v_k)\ ,\ v_k \sim p_{v_k}\ ,\ x_1 \sim p_{x_1} \\ y_k &= g(x_k) + e_k\ ,\ e_k \sim p_{e_k} \end{aligned} \tag{2.18}$$

where v_k is a stochastic noise process defined by the known probability density function (PDF) p_{v_k}. Similarly, e_k defines the measurement noise with known PDF p_{e_k}. The model can be defined in terms of conditional PDFs as

$$x_{k+1} \sim p(x_{k+1}|x_k) \tag{2.19}$$

$$y_k \sim p(y_k|x_k) \tag{2.20}$$

which is a more general model than what is described in Equations (2.18). This is because there is an allowance for implicit measurement relations $h(y_k, x_k, e_k) = 0$ as well as differential-algebraic equations that add state constraints to x_{k+1} [Gustafsson, 2010].

2.2 Computations of rotation

A GNSS as well as an IMU use what is known as coordinate frames, which is an inner representation of their own coordinate system. The GNSS uses satellites for measurements, while the IMU is highly dependent on how it is mounted on the eBike. These frames are rarely aligned with each other, leading to problems in the estimation. This is solved by rotating the coordinate frame of the IMU, setting its xy-plane to be parallel to the xy-plane of the GNSS. Further, the accelerometer in the IMU measures gravity as an acceleration, which must be compensated for. An IMU frame with a tilt causes gravity to affect more than one direction, i.e., the acceleration in x, y and z can all be affected. Rotating the coordinate frame to have the z-direction pointing upwards simplifies this compensation.
Rotations can be made in three different ways, which will all be discussed in this section.

Rotation matrices

The usage of rotation matrices enables the application of a Direct Cosine Matrix, DCM, on the IMU for gravity compensation [Hyyti and Visala, 2015]. The rotation matrix has to be updated continuously, since any movement will change the reference frame of the IMU. Multiplying one rotation matrix with another one represents how the coordinate frame is further rotated. This allows for a simple representation of how the matrix changes over time, as it is equal to the past matrix multiplied by the new rotation. A differential equation is used to express these updates, according to:

$$R(t+dt) = R(t)\begin{pmatrix} 1 & -d\theta_z & d\theta_y \\ d\theta_z & 1 & -d\theta_x \\ -d\theta_y & d\theta_x & 1 \end{pmatrix} \tag{2.21}$$

where

$$d\theta_x = \omega_x dt \tag{2.22}$$
$$d\theta_y = \omega_y dt \tag{2.23}$$
$$d\theta_z = \omega_z dt \tag{2.24}$$

are the angular velocities measured from the gyroscope [Premerlani and Bizard, 2009]. This equation updates the DCM from gyro signals, which are not affected by gravity. For simplicity, the rotation matrix R is initiated as $R = I$ where I denotes the identity matrix.

Euler angles

Another way of describing rotations is with the use of Euler angles [Weisstein, 2009]. Similar to rotation matrices, it describes rotations with angles and trigonometry. A rotation of ψ radians about the x-axis is defined as

$$R_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix}. \tag{2.25}$$

Similarly, a rotation of θ radians about the y-axis and a rotation of φ about the z-axis are defined as

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}, R_z(\varphi) = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.26}$$

The angles ψ, θ, φ are then the Euler angles, which give how much the coordinate frame has been rotated about each axis. A combination of rotations about several axes equals a multiplication of the matrices. However, they do not commute, meaning that the order of the axes which the frame rotates about will affect the resulting combined matrix. A rotation about x, then y, and lastly z would look like

$$R = R_z(\varphi)R_y(\theta)R_x(\psi) \tag{2.27}$$

[Slabaugh, 1999].

Quaternions

Another way of describing rotations is with quaternions, which can be described with four-dimensional vectors. They are commonly used in computer graphics as well as navigation, and they originate as an extension of complex numbers [Goldman, 2011].
Complex numbers were invented to extend calculations beyond real numbers, and have proven themselves useful in several applications. They take the form of

$$z = a + bi \tag{2.28}$$

where $i = \sqrt{-1}$ denotes the imaginary number, a is the real part and b is the imaginary part. The magnitude r of z is computed by the absolute value, and the phase angle φ is computed using trigonometry:

$$\begin{aligned} r &= |z| = \sqrt{a^2 + b^2} \\ \varphi &= \tan^{-1}\left(\frac{b}{a}\right) \text{ rad.} \end{aligned} \tag{2.29}$$

Since there is a possibility to achieve the magnitude as well as the angle from the rectangular form $z = a + bi$, the opposite is also achievable:

$$z = r(\cos\varphi + i\sin\varphi) = re^{i\varphi} \implies \tag{2.30}$$

$$a = r\cos\varphi \text{ , } b = r\sin\varphi \tag{2.31}$$

[Månsson and Nordbeck, 2011]. Lastly, the multiplication of a complex number with i is geometrically viewed as a *rotation*, 90° in a counterclockwise direction, which is a main property that is used for quaternions.

Quaternions can be represented by entities in a 4-dimensional space, just as complex numbers can be represented by entities in a 2-dimensional space. Quaternions are simply an extension of complex numbers into 4 dimensions. Similarly to complex numbers, quaternions can be expressed in a rectangular form:

$$q = a\mathbf{1} + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} \tag{2.32}$$

where a, b, c, d are real numbers, and $\mathbf{1}, \mathbf{i}, \mathbf{j}, \mathbf{k}$ are basis vectors. All of these basis vector extensions, i.e., $\mathbf{i}, \mathbf{j}, \mathbf{k}$, have the property of $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$. However, quaternions are *not* commutative and have their own multiplication table according to Table 2.1.

Table 2.1 Quaternions multiplication table [Goldman, 2011].

	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

Recall that multiplication with i can be viewed geometrically as a rotation. j and k have similar properties but in other dimensions.
Similar to how multiplications of complex numbers represent rotations in the complex plane, multiplications with quaternions can be used to represent rotations in a 3-dimensional Euclidean space.
In some cases, the usage of quaternions appears to be a more natural way to perceive rotation rather than Euler angles, which has several disadvantages. The main disadvantage is that gimbal lock can occur using Euler angles [Dam et al., 1998]. A gimbal is a pivoted support that enables rotations about an axis. A set of three gimbals, one mounted on the other with orthogonal mounting axes allow an object mounted on the innermost gimbal to remain independent of outer rotations, see Figure 2.1. However, if the body subject to rotations is rotated about an axis normal to the common plane of the gimbals, the platform becomes locked. Gimbal lock can be worked around, by predicting how successive rotations about the basis axes affect each other. It is possible to create a series of rotations, however, one degree of freedom will be lost [Hemingway and O'Reilly, 2018].

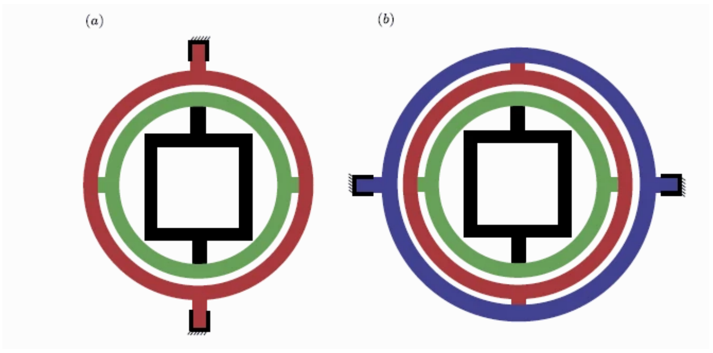

Figure 2.1 Example of a platform mounted in (a) two and (b) three gimbals for a three-axis and a four-axis suspension, respectively [Hemingway and O'Reilly, 2018].

The usage of quaternions removes the problem of gimbal lock. Further, quaternions also have an obvious geometrical interpretation, coordinate system independency as well as a compact representation [Dam et al., 1998].

2.3 Inertial Measurement Unit

An Interial Measurement Unit, or IMU for short, is an electronic device that includes gyroscopes and accelerometers. These sensors enable tracking of rotational and translational movements [Madgwick et al., 2011]. The IMU is used to measure whether the eBike has been rotated. In contrast to a GNSS, the IMU can sense small changes in orientation, as it measures the acceleration and angular velocity.
Recent developments allow for the production of IMU-enabled GPS services, where an IMU allows a GPS receiver to work when the GPS signals are unavailable. Some examples of events when this is apparent are in tunnels, buildings or when there is interference from electronics [Fang et al., 2018].

Figure 2.2 The bcm3 unit containing a GNSS as well as an IMU.

The different sensors used in an IMU measure their respective quantities, in three different directions. All of the sensors have their own coordinate frames. However, these frames are more or less aligned with each other and do not require calibration on their own. Calibrations are instead made to align the IMU with a GNSS. A unit combining a IMU with a GNSS can be viewed in Figure 2.2.

Accelerometer

Measurement of acceleration is often based on connecting a proof mass to some form of spring, in relation to the sensor's coat [Grahm et al., 1996]. When external vibrations are applied to the coat, the proof mass stays still, and the acceleration is then measured from the distance between the mass and the coat. There are several types of accelerometers, namely piezoelectric accelerometers, strain-gage accelerometers, and the mass balance accelerometer [Grahm et al., 1996]. The former is the most used due to it being very light, while it also has a very high resonance frequency.

The piezoelectric accelerometer is a seismic transducer, which uses a piezoelectric crystal to measure the position of the proof mass [Grahm et al., 1996]. When the piezoelectric element is subject to compression or elongation, a charge is generated over the element. This charge is proportional to the acceleration, $Q = K \cdot a$, where K is a constant. From a physical point of view, the piezoelectric accelerometer is a charge generator that charges the internal capacitance. The voltage over the capacitance is then

$$U = \frac{Q}{C_g} = \frac{Ka}{C_g}. \tag{2.33}$$

Since the piezoelectric accelerometer only measures AC voltages, it will in theory not give a correct output when the eBike is still. In practice, however, noise will still be apparent in the measurement. As soon as the accelerometer is set in motion, it measures the acceleration [Grahm et al., 1996, Chapter 9].

Gyroscope

Gyroscopes are devices that are mounted on a body. This enables the sensor to measure an angular velocity if the body is rotating, which can be used alone or in more complex systems as a combination with other sensors. Some examples of these more complex systems are gyrocompasses, *IMUs* and inertial navigation systems. There are several classes of gyroscopes that depend on the involved technology, as well as the operating physical principles. Some of the more common gyroscopes are mechanical gyroscopes, optical gyroscopes as well as micro-electromechanical system gyroscopes (MEMS) [Passaro et al., 2017].

The micro-electromechanical system gyroscopes generally use a vibrating element as a sensing element to detect angular velocity [Watson, 2020]. They are based on the transfer of energy between two vibration modes caused by the Coriolis acceleration, which is an apparent acceleration that is observed in a rotating frame of reference.

2.4 Global Navigation Satellite System

The Global Navigation Satellite System, GNSS for short, covers each individual global satellite-based positioning system as well as the combination of the systems. The generic GNSS receiver consists of three functional blocks: a radio frequency front-end (RF), a digital signal processor (DSP), and a navigation processor. The RF front-end receives the incoming signals from satellites, which are then converted to an intermediate frequency, and an A/D converter samples the signals. The DSP correlates generated signals with satellite signals and proves code ranges, carrier phases, Doppler frequencies, as well as the navigation data streams. The navigation processor decodes the navigation message to gain information, which it uses to compute position, velocity, and time information [Hofmann-Wellenhof et al., 2008, Chapter 4.3].

Considering the distance from satellites to Earth, the GNSS has a good accuracy to be a reliable source of position tracking. However, on a close scale between sample points, there is a risk that the GNSS does not recognize yaw movements, meaning that a turn could be represented as a straight line through buildings. The IMU can be used to compensate for this, thanks to a much faster sample time, enabling position tracking on a smaller scale.
The GNSS provides the location for the measurement. From the location, a velocity magnitude and direction can be computed. However, the GNSS must be in motion to give directional information. If the bicycle is stopped and rotated, the GNSS will not comprehend this rotation and will still assume that the direction stays the same [Premerlani and Bizard, 2009].

2.5 Kalman Filters

The Kalman filter is an algorithm that uses a series of measurements over time and a dynamic model, and then estimates the state of a process, in a way that minimizes the mean of a squared error between this estimated state and the real state [Mu and Xiong, 2018]. The goal of the filter is to compute optimal estimates of the states of a modeled process, using noisy measurements of that process, recursively. There are difficulties in modeling a real process, and the model is rarely a match with the real process. However, Kalman filters are still useful in the sense that they are relatively simple and robust [Welch, 2020].
There are several types of Kalman filters, and each has its own area of usefulness. They operate on the same working principles, but vary in complexity depending on

their model. Due to non-uniform validation methods and tuning parameters for the filters, the performance can be difficult to assess [Campestrini et al., 2016].

If there is no possibility to measure all states, the Kalman filter can be used to predict those un-measurable states. Normally, the state vector is not directly observed, but only the outputs y. The Kalman filter is used to reconstruct the estimation of the states, $\hat{x}$, with only y and u. The simplest simulation of the system is of the form

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) \tag{2.34}$$

and the estimation error is then formed as

$$\tilde{x}(t) = x(t) - \hat{x}(t). \tag{2.35}$$

If the estimation is perfect, $\tilde{x}(t) = 0$ as there will be no error. The goal in the general observer problem is to minimize this estimation error [Glad and Ljung, 1997, Chapter 5].

Observability

Consider an n^{th}-order system

$$\begin{aligned} x_{k+1} &= \Phi x_k \\ y_k &= C x_k. \end{aligned} \tag{2.36}$$

By measuring the outputs $y_0, y_1, ..., y_{n-1}$, it is possible to uniquely determine the initial state x_0 if, and only if, the observability matrix [Glad and Ljung, 1997, Chapter 3]:

$$O = \begin{pmatrix} C \\ C\Phi \\ C\Phi^2 \\ . \\ . \\ . \\ C\Phi^{n-1} \end{pmatrix} \tag{2.37}$$

has rank n. Further, a system is detectable if its unobservable subspace is stable. A Kalman filter requires an observable system, as unobservable states yield no information, which means that the filter estimate for the states will not converge to a meaningful solution [Southallzy et al., 1998].

The discrete time-varying Kalman Filter

Now, consider equation (2.36) but with explicit modeling of the process disturbance w_k and the measurement noise v_k as white noise processes with known variances, using u_k as a system input. The process is assumed to be observable, and we have:

$$\begin{aligned} x_{k+1} &= \Phi x_k + \Gamma u_k + G w_k \\ y_k &= C x_k + D u_k + v_k \end{aligned} \tag{2.38}$$

where

$$\begin{pmatrix} E[w_k w_k^T] & E[w_k v_k^T] \\ E[v_k w_k^T] & E[v_k v_k^T] \end{pmatrix} = \begin{pmatrix} Q & R_{wv} \\ R_{wv}^T & R \end{pmatrix} \tag{2.39}$$

are the covariance matrices of the state disturbance and measurement noise. Further, an assumption can be made that $R_{wv} = R_{wv}^T = 0$ as w_k and v_k are uncorrelated due to them being white noise processes [Glad and Ljung, 1997, Chapter 5].
The so-called time-varying Kalman filter can then be described as:

$$\begin{cases} \hat{x}_{k+1|k} = \Phi \hat{x}_{k|k} + \Gamma u_k \\ P_{k+1|k} = \Phi P_{k|k} \Phi^T + GQG^T \end{cases} \\ \begin{cases} \hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k(y_k - C\hat{x}_{k|k-1}) \\ P_{k|k} = P_{k|k-1} - L_k C P_{k|k-1} \end{cases} \tag{2.40}$$

where the first bracket is known as the *prediction step* and the latter is the *measurement step* [Glad and Ljung, 1997, Chapter 5]. Further, the optimal filter gain at time step k is given as:

$$L_k = P_{k|k-1} C^T (C P_{k|k-1} C^T + R)^{-1}. \tag{2.41}$$

Recall the covariance matrices in equation (2.39). With the following assumptions:

1. $Q = Q^T > 0$ is a positive definite matrix,
2. $R = R^T > 0$ is a positive definite matrix,
3. The pair (Φ, G) is controllable, i.e.,

$$\operatorname{rank}[G|\Phi G|...|\Phi^{n-1}G] = n, \tag{2.42}$$

4. The pair (Φ, C) is observable, i.e.,

$$\operatorname{rank}[C^T|\Phi^T C^T|...|\Phi^{T^{n-1}} C^T] = n \tag{2.43}$$

yields the following result:

1. The prediction matrix $P_{k|k-1}$ converges to a constant-valued matrix

$$\lim_{k\to\infty} P_{k|k-1} = P \tag{2.44}$$

   where P is a symmetric positive definite matrix, i.e., $P = P^T > 0$,

2. P is the unique positive definite solution of the discrete algebraic Riccati equation:

$$P = \Phi P \Phi^T - \Phi P C^T [CPC^T + R]^{-1} C P \Phi^T, \tag{2.45}$$

3. P is independent of $\sum_0$ provided that $\sum_0 \geq 0$

where $\sum_0 = E[(x_0 - \mu_{x_0})(x_0 - \mu_{x_0})^T]$ denotes the covariance matrix of the initial state [Ribiero, 2004].

The Extended Kalman Filter

The standard Kalman filter is formulated for linear systems. When the considered dynamics are non-linear, the system model must be linearized to apply the Kalman filter. Consider the dynamics describing a non-linear system:

$$\begin{aligned} x_{k+1} &= f_k(x_k) + g_k(u_k) + w_k \\ y_k &= h(x_k) + v_k \end{aligned} \tag{2.46}$$

where

$$\begin{aligned} &x_k \in \mathbb{R}^n \text{ , } f_k(x_k) : \mathbb{R}^n \longrightarrow \mathbb{R}^n \\ &u_k \in \mathbb{R}^\ell \text{ , } g(u_k) : \mathbb{R}^\ell \longrightarrow \mathbb{R}^n \\ &y_k \in \mathbb{R}^r \text{ , } h_k(x_k) : \mathbb{R}^n \longrightarrow \mathbb{R}^r \\ &v_k \in \mathbb{R}^r \\ &w_k \in \mathbb{R}^n \end{aligned} \tag{2.47}$$

and $\{v_k\}, \{w_k\}$ are still white Gaussian, independent processes with zero mean and unit variance. These processes have the covariance matrices of:

$$E[v_k v_k^T] = R_k \text{ , } E[w_k w_k^T] = Q_k \tag{2.48}$$

and the initial condition is considered a Gaussian random vector. There is then a set of system measurements $Y_1^k = \{y_1, y_2, ..., y_k\}$. The goal of the filter is to obtain an estimate of the state that the system has, given these measurements.
The EKF is composed of the following steps [Becker, 2023]:

1. Consider the last predicted state $\hat{x}_{k|k}$,
2. Linearize the system dynamics, $x_{k+1} = f_k(x_k) + g_k(u_k) + w_k$ around this state,
3. Apply the prediction step of the Kalman filter to the linearized system dynamics, yielding $\hat{x}_{k+1|k}$ and $P_{k+1|k}$,
4. Linearize the observation dynamics $y_k = h_k(x_k)$ around the predicted state $\hat{x}_{k+1|k}$,
5. Apply the filtering or update cycle, yielding $\hat{x}_{k+1|k+1}$ and $P_{k+1|k+1}$.

Now, let F_k and G_k be the Jacobian, partial derivative, matrices of f and g, as well as H being the observation matrix describing the measurable states. Recalling the similarity in equation (2.40), the Extended Kalman filter instead consist of [Ribiero, 2004]:

$$\begin{aligned} &\begin{cases} \hat{x}_{k+1|k} = f(\hat{x}_{k|k}) + g(u_k) + w_k \\ P_{k+1|k} = F_{k+1} P_{k|k} F_{k+1}^T + Q_{k+1} \end{cases} \\ &\begin{cases} \hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k [y_k - H \hat{x}_{k|k-1}] \\ L_k = P_{k|k-1} H^T (H P_{k|k-1} H^T + R_k)^{-1} \\ P_{k|k} = (I - L_k H) P_{k|k-1} (I - L_k H)^T + L_k R_k L_k^T \end{cases} \end{aligned} \tag{2.49}$$

where I denotes the identity matrix, i.e., a square matrix with ones along the diagonal, and zeros elsewhere. These steps are denoted as the *prediction cycle* (first bracket), and *measurement cycle* (second bracket) [Ribiero, 2004]. Here, $\hat{x}_{k+1|k}$ denotes the predicted next step, given the last estimated state $\hat{x}_{k|k}$. $P_{k+1|k}$ denotes the predicted error covariance estimate, given $P_{k|k}$. As for the measurement step, $\hat{x}_{k|k}$ denotes the measured state, $P_{k|k}$ is the error covariance given the measured output, and L_k is, as mentioned, the optimal filter gain. The goal is to find a L_k that minimizes the diagonal sum of $P_{k|k}$, yielding optimal filtering [Becker, 2023].

2.6 Noise models

Recalling the section of Kalman filters, the process disturbance, as well as the measurement noise, are assumed to be white Gaussian noise processes. In reality, noise can take any form and can be completely random. Assumptions of white Gaussian noise can therefore lead to worse performances. A solution to this problem is to investigate different noise models that could fit the noisy data in a better way.

Gaussian noise

The Gaussian distribution is the most studied probability distribution for continuous stochastic variables, because of its simplicity and the computationally convenient properties that it has. It is not only used to define methods of machine learning like linear regression but also in signal processing as in Kalman filters, control theory as in Linear Quadratic Regulators, as well as in basic statistics with hypothesis testing [Deisenroth et al., 2021].
A Gaussian distribution is a common model for noise, given its simplicity. In a Gaussian process, all process values have normal distributions, and all linear operations like summation, differentiation, and integration produce normally distributed variables [Lindgren et al., 2013]. The probability density function is given as

$$p(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.50}$$

where σ is the standard deviation and μ is the mean [Boyat and Joshi, 2015], [Deisenroth et al., 2021].

White noise

Noise is characterized by the magnitude, and the spectrum of power is *constant* for white noise. The noise power is essentially equivalent to the power spectral density function. A random vector is said to be a white noise vector if each component has a probability distribution with mean and variance as $\mu = 0, \sigma^2 < \infty$, and is statistically independent, meaning that their joint probability distribution is the product of each individual components distribution [Fessler, 1998].

Poisson noise

Poisson Noise is a basic form of uncertainty, used in the measurement of light, and commonly found in electronics [Trussell and Zhang, 2012]. It is modeled using the Poisson distribution which has its probability density function given as

$$p(\lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{2.51}$$

where λ denotes the mean and variance, and k is the number of occurrences. While a general Gaussian distribution is modeled as $\mathcal{N}(\mu, \sigma^2)$, the Poisson distribution instead models as $Po(\lambda)$ and under specific conditions, it can be approximated with the help of a normal distribution as$\mathcal{N}(\lambda, \lambda)$, i.e., the Poisson process is a special case of a Normal distribution. This approximation becomes more accurate for larger values of λ [Hasinoff, 2014].

Moving Average Noise

The Moving Average (MA) process is a basic form of linear filter, which is defined by the equation

$$y_t = e_t + c_1 e_{t-1} + \cdots + c_q e_{t-q} \tag{2.52}$$

where $c_q \neq 0$ and e_t is a zero-mean white noise process with variance σ^2. The constants $c_1, c_2, \ldots, c_q$ are moving average coefficients. In other words, a process of order q, shortened as $MA(q)$, is a linear combination of several white noise realizations [Jakobsson, 2020].

Autoregressive noise

An Autoregressive (AR) process is a representation of a random process that can be used to represent time-varying processes in nature, economics, etc [Møller, 2008]. It is, along with the MA process, one of the two most common basic linear processes. It is defined as

$$y_t + a_1 y_{t-1} + \cdots + a_p y_{t-p} = e_t \tag{2.53}$$

where $a_p \neq 0$ and e_t is a zero-mean white noise process with variance σ^2, it is also dependent on a white noise process. The constants $a_1, a_2, \ldots, a_p$ are autoregressive coefficients, and the process of order p, described as $AR(p)$, is a linear combination of previous values [Jakobsson, 2020].

ARMA noise

The Autoregressive Moving Average (ARMA) process is a combination of the previously mentioned $AR(p)$ and $MA(q)$ processes [Brockwell, 2001]. It is defined as

$$y_t + a_1 y_{t-1} + \cdots + a_p y_{t-p} = e_t + c_1 e_{t-1} + \cdots + c_q e_{t-q} \tag{2.54}$$

where the conditions on the variables are the same as before. Modeling the polynomials

$$A(z) = 1 + a_1 z^{-1} + \cdots + a_p z^{-p},$$
$$C(z) = 1 + c_1 z^{-1} + \cdots + c_q z^{-q},$$

the process is stationary if the roots of $A(z) = 0$ lie within the unit circle, and it is invertible if the roots of $C(z) = 0$ do.
The process is appropriate when the system functions are series of unobserved shocks. It is a tool for understanding and predicting future values, given a time series of data [Jakobsson, 2020].

2.7 Random search

To choose ideal parameters for the Kalman filter, algorithms such as a random search can be used [Bergstra and Bengio, 2012]. They are frequently used in stochastic optimization problems such as simulations, where the goal is to find optimal hyperparameters given certain constraints. A random search picks random parameters and evaluates the performance, using evaluation metrics such as Root Mean Squared Errors. If certain criteria are met, the parameters are saved. These criteria can be to minimize an error, or a cost function, or to maximize a certain value. If criteria are not met, the algorithm picks new random values. Given the interval of parameters able to be chosen, as well as the number of iterations, the optimal parameters converge [Zabinsky et al., 2009].

2.8 Related work

Kalman filters have been historically useful in the context of navigation for vehicles, motion planning, and even neuron dynamics to modulate the brain cortex [Schiff, 2009].

An article was written on how to improve the accuracy of a GPS using a Weighted Kalman filter based on the variance estimation method. The article used a regular Kalman filter and focused on the noise variance estimation, which improved the accuracy with up to 30% [Shokri et al., 2020]. They chose to improve this aspect instead of using sensor fusion, based on the argument of sensor fusion being costly and having a high time complexity. This means that there is no certainty that the combination of sensor fusion along with noise variance estimation has been investigated, which could yield even better results.

In a book written by Wan and Van Der Merwe [Wan and Van Der Merwe, 2000], there was an investigation on how the Extended Kalman filter compared in performance against an Unscented Kalman filter. While the EKF worked well for nonlinear systems with noise that was Gaussian, or close to Gaussian, the performance deteriorated when the noise models differed. The UKF was able to handle this problem using a sampling approach, approximating the state distribution by Gaussian variables. This means, that for tasks where the process disturbances, as well as the measurement noise, are unknown, the UKF has an overall better performance [Wan and Van Der Merwe, 2000].

Campestrini et al. wrote an article on reviewing different Kalman filters by applying an enhanced validation method [Campestrini et al., 2016]. The filters tested were the regular Kalman filter, the Extended Kalman filter, as well as the Unscented Kalman filter, along with some modified versions of these filters. Used for state-of-charge estimation of lithium-ion cells, the conclusion was that the Dual Extended Kalman filter gave the best performance. Next, the writers were to investigate optimal tuning parameters of the filter [Campestrini et al., 2016].

3
Method

In this chapter, the key elements of this thesis are described. Firstly, the data collection and data generation used for testing and simulations are described. Secondly, the calibration in the pre-processing is described, which includes rotation of coordinate frames, compensation of bias and gravity, as well as conversion between different coordinate systems. Next, the details of different model dynamics with the respective Kalman filter parameters are explained. Lastly, ways of optimizing the performance are discussed.

3.1 Data collection

As the main purpose is to improve the accuracy of the GNSS, the data is retrieved from realistic scenarios where an electric bicycle would be used. These scenarios are riding the eBike on clear paths such as the streets, but also on walkways, as well as in parks. The point is not only to implement a position tracking algorithm that works on the streets, but also if the eBike would be taken off the road network where GNSS signals may be distorted.
The data sets consist of measured values from the IMU, as well as the output from the GNSS. The IMU gives an acceleration in the x, y and z-directions from the accelerometer, as well as an angular velocity around these axes from the gyroscope. The GNSS gives coordinates in North and East, as well as the velocity of the eBike between two points. It also gives the horizontal dilution of precision, which is a way to describe how errors in the measurement affect the final state estimation [Dudek and Jenkin, 2010]. These coordinates need to be converted to values in a coordinate system corresponding to the eBike, that is in x, y, z. Coordinates from the GNSS is converted to the Earth-Centered Earth-Fixed coordinate system (ECEF) [Drake, 2002]. From there, the data is converted to a tangential plane that corresponds to the movements of the eBike. This is to simplify the comparison to the IMU data, since the IMU measurements are given in a tangential coordinate frame after internal rotations.

3.2 Data generation

While the goal is to improve the accuracy of the position tracking for real data, the reliability of a ground truth for comparison and evaluation is limited. Because of this, data is generated to enable the possibility of performance tests of prediction algorithm in simulations.
The data is generated to correspond to the data of an IMU and a GNSS, meaning that there is a need of acceleration as well as angular velocity. The velocity is generated using functions based on random seeds, which correspond to an approximation of realistic trajectories of movement across an x,y-plane. The acceleration and position were then retrieved from numerical differentiation and integration of the velocity respectively, and the angle of direction is computed using the velocity. Finally, the angular velocity correspond to numerical differentiation of the directional angle.
The position integrated from the generated velocity represents the ground truth of the trajectory. From this ground truth, the data for the IMU as well as the GNSS can be developed by the addition of noise. This noise can take many different forms, as discussed in Section 2.6. A general approach was to implement the noise as Gaussian distributions with different parameters. When simulating the trajectories, the variance of the noise was also implemented as varying over time, to ensure that the Kalman filter would work in a more general sense, and not only for the simplest of examples. The IMU had given noise for both accelerometer and gyroscope measurements, while the GNSS noise was unknown. The noise models used were therefore implemented on the GNSS, which are Gaussian noise, Poisson noise, AR noise and lastly ARMA noise, while the IMU noise model stayed the same for all simulations.

3.3 Calibration of the IMU

Calibration is necessary for the raw data obtained from the IMU. Although the accelerometer and gyroscope share the same coordinate frame, it differs from the earth's coordinate frame. The sensor cannot be mounted parallel to the ground, requiring rotations of the coordinate frame to eliminate the effect of gravity on the measurements. This adjustment is important since the earth's gravity results in a constant acceleration towards the ground. Ideally, the calibration enables measurements on several bikes, instead of limiting a good performance to one specific.

Rotation of the IMU coordinate frame

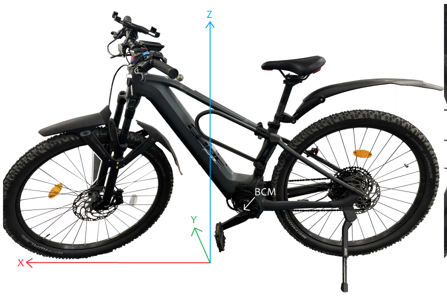

Figure 3.1 The eBike's coordinate frame visualized.

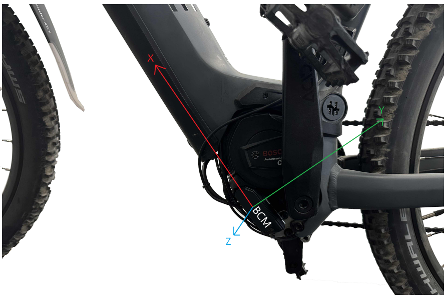

Figure 3.2 The mounted BCM's coordinates frame visualized.

To achieve a compatible input for the Kalman filter, the IMU has to use the same coordinate frame as the GNSS. Since the IMU can be mounted on the eBike in several ways, this orientation can be changed. While the main test runs were made on the same eBike, the IMU still has a shifted coordinate frame in comparison to the one of the bike, see Figures 3.1, 3.2. The goal is to find a coordinate frame that has the same positive z direction as gravity. This is done by measuring the gravity vector $\mathbf{g} = \begin{pmatrix} g_1 & g_2 & g_3 \end{pmatrix}$, which is used to create the vertical axis $\hat{e}_3 = \mathbf{g}/|g|$. The other two axes are orthogonal to $\hat{e}_3$, implemented as:

$$\hat{e}_2 = \begin{pmatrix} g_2 & -g_1 & 0 \end{pmatrix} \text{ and } \hat{e}_1 = \hat{e}_2 \times \hat{e}_3. \tag{3.1}$$

Gravity compensation

If the gravity component is not properly subtracted, an offset in acceleration is created. This offset leads to large drifts in position when integrated. There are several ways of implementing gravity compensation. From a physical point of view, the goal is to represent the orientation of the IMU with respect to the earth as a rotation. This can be done for example by the usage of either quaternions or rotation matrices. While quaternions allow for simple computations as a result of the requirement of only four parameters (compare to rotation matrices using 9), rotation matrices have an advantage of being a natural fit for a navigation problem.

Two different methods of calibration were used, one including rotation matrices, and the other containing quaternions. Using rotation matrices, the coordinate frame of the IMU was continuously rotated with the matrix, to achieve a frame similar to the earth's tangential coordinate frame. The gravity was then removed in the z-direction according to:

$$a(t+1) = R(t+1)a(t) - g \tag{3.2}$$

where g denotes the gravity of earth, and a the acceleration.

Quaternions were used in a similar fashion. The quaternions corresponding to the rotation, q, as well as the conjugate, q^*, were calculated, and then used as:

$$a' = qaq^* \tag{3.3}$$
$$a'' = a' - g \tag{3.4}$$
$$a''' = q^{-1}a''(q^{-1})^* \tag{3.5}$$

where a', a'', a''' simply denotes changes in the initial acceleration vector a. Simply put, the coordinates of the acceleration are transformed to the one of the earth. The gravity is removed, and the coordinate frame is rotated back to the original as a final step. Over time, the coordinate frame will rotate because of the inherent drift in the gyro, caused by the bias in the sensors. Therefore, this process is done for each iteration of the Kalman filter to ensure that only the gravity component of the vectors is removed.
These are two different methods of rotating the IMU coordinate frame, namely Direct Cosine Matrices, Equation (3.2), and quaternions, Equation (3.5), and the method mainly used is described in equation (3.2).

Bias calibration

The IMU is noisy, and have a bias. The noise can be evaluated using models, which is one of the methods used for tuning the Kalman filter. The bias for the gyroscope is estimated by running a simulation of the IMU laying completely still, and removing the mean value. For the accelerometer on the other hand, a test run was made of rotations to receive gravity in all directions. In other words, the IMU was rotated around a fixed point to be affected by gravity in all possible directions. Using spherical regression, a sphere was created which mimics the data points achieved, similar to how linear regression fits a line to a limited number of samples in two dimensions. The key difference is that in linear regression, a line is fit to the points, while the sphere is created using planes. The points are grouped by their proximity, and a plane is then fit to those. Several planes are then combined to create the sphere. This sphere closely resembles how the accelerometer is affected by gravity, but also gives the bias of the sensor, since all of the measured points are gravity accelerations in different directions. The bias is represented by the middle point of the sphere, which is then subtracted from the raw data, see Figure 3.3. Linear regression yields the same sphere, and may be more precise, but requires more computations in comparison to using planes. As the centre of the sphere is sought, the precision of the sphere's surface is irrelevant, leading to a superiority in using spherical regression.

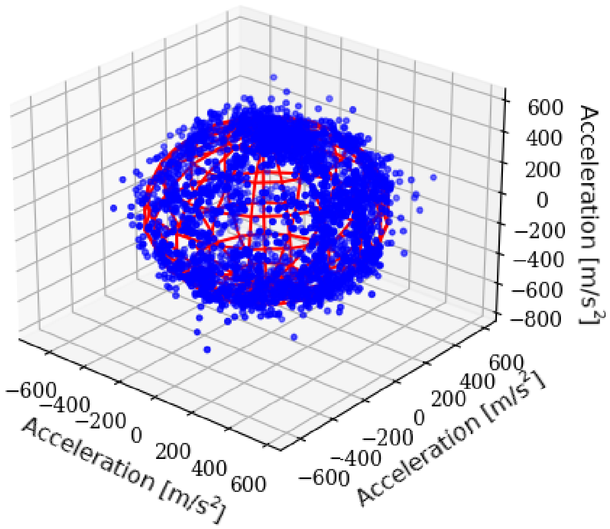

Figure 3.3 Spherical regression of a test run where the IMU is rotated, but not moved in different directions. Data samples in blue, created sphere in red.

3.4 Calibration of the GNSS

The GNSS gave geodetic coordinates, that is latitude, longitude and altitude. These are converted to tangential coordinates, which is a similar frame as the one of the IMU. These coordinates are to be represented in the same frame as those of the IMU for the Kalman filter to have a realistic chance of predicting the next step.
The process of converting geodetic coordinates into tangential coordinates is simplified by considering the earth as a sphere mathematically. The idea is that, since all the measurements are located within just a few kilometers of each other, the error will be negligible enough for the sensor to be used for navigation. The alternative of using elliptical geometry to represent the earth can become computationally heavy, and is unnecessary for the purpose of this thesis.
To start the conversion, the average circumference of the earth is needed. This can either be calculated or found to be approximately 40075 km. To convert longitude (λ) and latitude (φ) into meters, the idea that one degree in geodetic coordinates is the same as one degree around the spherical earth is used. In meters, one degree in latitude corresponds to: $40075/360 \approx 111.32$ km. The value for longitude is dependent on which degree of latitude one is located. Trigonometry is used to consider this, and one degree in longitude is calculated to: $40075 \cdot \cos(\varphi)/360$. Similarly, one degree in latitude is computed as: $40075 \cdot \sin(\varphi)/360$.

The degrees of longitude and latitude can now be transformed into meters in an efficient way. The initial value of the bikes position is used as origin since using the equator as origin can be quite cumbersome. Using the initial position value as an origin also works well for the IMU, since it has its origin at the starting position of the bike.

3.5 Models

As Kalman filters are the main approach in this thesis, a mathematical model is needed to make predictions. Using the data from both the accelerometer and gyroscope, the model resembles how the measurements change over time. While the accelerometer gives a plain acceleration as output, a position is needed to be compared with the position given by the GNSS. This is done by using a double integrator in the modeling.
The system is assumed to have no noteworthy disturbances in the beginning, meaning that the state propagates with respect to the last state, as well as the input only. The nonlinear difference equation

$$x_{k+1} = f(x_k) + g(u_k) \tag{3.6}$$

describes how the state vector changes over time. This can then be modeled in several ways, each one having their own pros and cons.

The Kalman filter is divided in a prediction and a measurement step. The state is predicted through the dynamics in equation (3.6), while the measured output, z_k, is modeled as

$$z_k = h(x_k^{GNSS}) \tag{3.7}$$

with the same dynamics for all of the different models. The error covariance P is then estimated as

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q \tag{3.8}$$

where F_k is the Jacobian matrix from the model dynamics and Q is the process noise covariance matrix. The P matrix denotes the estimated covariance over time, and is chosen to have initial values of Q. In a way, this tells the Kalman filter that certain state measurements may be more accurate than others. An example of this is when the IMU coordinate frame is rotated from the GNSS. In this case, the angle given from the IMU readings is still accurate, while the positions and velocities could give wrongful values. For the purpose of this study, cross-covariance between the states is neglected, as the measurement noise between different states is minuscule in comparison to the covariances of the states.

Linear model

The linear model used disregards the angle of direction for the IMU. It only considers the acceleration in different directions and uses that as an input to the filter. While simple, as well as efficient in an ideal setting, i.e., when the orientation of the GNSS and IMU are the same, the performance is poor when the coordinate frames are not aligned. The states and inputs used are:

$$\hat{X}_k = \begin{pmatrix} \hat{x}_k \\ \hat{y}_k \\ \hat{v}_{x,k} \\ \hat{v}_{y,k} \end{pmatrix} \text{ and } u_k = \begin{pmatrix} a_{x,k} \\ a_{y,k} \end{pmatrix} \tag{3.9}$$

where x_k, y_k denotes the position of the eBike at time step k, $v_{x,k}$, $v_{y,k}$ are the directional velocities and $a_{x,k}, a_{y,k}$ are the directional accelerations.
The linear model dynamics are formed as:

$$\begin{aligned} x_{k+1} &= x_k + v_{x,k}T_s \\ y_{k+1} &= y_k + v_{y,k}T_s \\ v_{x,k+1} &= v_{x,k} + a_{x,k}T_s \\ v_{y,k+1} &= v_{y,k} + a_{y,k}T_s. \end{aligned} \tag{3.10}$$

where T_s denotes the sample time of the IMU. Using a linear model, the Jacobian matrix is simply a constant matrix , without any dependency of which time step the measurement is made. It, as well as the observation matrix, are computed as:

$$F = \begin{pmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{3.12}$$

where H is determined from the knowledge that there is no speed measured from the GNSS, although it can be computed. The only readings used are the positions.

First-order model

The first-order model is similar to the linear model, with the addition of the angle of direction as a state. The two different velocity states are replaced by a speed, received by integrating the norm of the acceleration in x and y. The positions are computed by using this speed, v_k, together with this angle, to give a more accurate representation of how the position changes over time. The state vectors, as well as the input vector, takes the form:

$$\hat{X}_k = \begin{pmatrix} \hat{x}_k \\ \hat{y}_k \\ \hat{\theta}_k \\ \hat{v}_k \end{pmatrix} \text{ and } u_k = \begin{pmatrix} a_{x,k} \\ a_{y,k} \\ \omega_k \end{pmatrix} \tag{3.13}$$

where θ_k denotes the angle of direction and ω_k denotes the angular velocity. The model dynamics are:

$$\begin{aligned} x_{k+1} &= x_k + v_k T_s \cos\theta_k \\ y_{k+1} &= y_k + v_k T_s \sin\theta_k \\ \theta_{k+1} &= \theta_k + \omega_k T_s \\ v_{k+1} &= v_k + a_k T_s \\ a_{k+1} &= \sqrt{a_{x,k}^2 + a_{y,k}^2} \end{aligned} \tag{3.14}$$

Since the model is nonlinear, the Jacobian matrix changes over every time step k. It, along with H, are formed as:

$$F_{k+1} = \begin{pmatrix} 1 & 0 & -v_k T_s \sin\theta_k & T_s \cos\theta_k \\ 0 & 1 & v_k T_s \cos\theta_k & T_s \sin\theta_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.15}$$

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{3.16}$$

The observation matrix only uses the x, y positions as GNSS readings.
To ensure that the Kalman filter works as expected, some consideration has to be put into the initial guesses of the states. In the case of this study, the positional values, x and y, are set to zero and the velocity scalar, v, is also set to zero. θ needs to be more carefully considered as this value affects the rotation of the Kalman filter positions. Fortunately, the GNSS can be used to get an acceptable angle for θ. To do this, two points are calculated based on the GNSS data. The first point is the average of the first five values of the data, the second point is the average of the next five values. By averaging, the effect of possible outliers can be reduced, resulting in a more precise value for θ. A first-order differentiation is then used to get the vector between the two points. Lastly, trigonometry is used to find the angle between y and x. This is used as the initial value for θ.

Second-order model

The second-order model utilizes the knowledge that both the velocity, as well as the acceleration affects the position. This models views the rotation when computing the position, as well as the velocity. Further, the position is not only correlated to the previous velocity, but also to the current acceleration. The states used are the same as in the previous models, i.e.,

$$\hat{X}_k = \begin{pmatrix} \hat{x}_k \\ \hat{y}_k \\ \hat{\theta}_k \\ \hat{v}_k \end{pmatrix} \text{ and } u_k = \begin{pmatrix} a_{x,k} \\ a_{y,k} \\ \omega_k \end{pmatrix}. \tag{3.17}$$

The models are now formed as:

$$\begin{aligned} x_{k+1} &= x_k + v_k T_s \cos\theta_k + a_k \frac{T_s^2}{2} \cos\theta_k \\ y_{k+1} &= y_k + v_k T_s \sin\theta_k + a_k \frac{T_s^2}{2} \sin\theta_k \\ \theta_{k+1} &= \theta_k + \omega_k T_s \\ v_{k+1} &= v_k + a_k T_s \\ a_k &= \sqrt{a_{x,k}^2 + a_{y,k}^2} \end{aligned} \tag{3.18}$$

which yields the same state vector as for Model 1. Further, the Jacobian matrix F_k along with the observation matrix H are computed as:

$$F_{k+1} = \begin{pmatrix} 1 & 0 & -T_s \sin\theta_k (v_k + a_{k+1}\frac{T_s}{2}) & T_s \cos\theta_k \\ 0 & 1 & T_s \cos\theta_k (v_k + a_{k+1}\frac{T_s}{2}) & T_s \sin\theta_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.19}$$

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{3.20}$$

The same initial values for the states are used for this model as the first-order model.

3.6 Parameter optimization

Recall the matrix Q in equation (2.49), which is a 4x4 diagonal matrix defining the process noise covariance. Put more simply, Q is the amount of trust in each of the different process states. An approximation of Q is found by running several simulations, utilizing a logarithmic random search optimization method and testing different parameters for the parameters in Q to find the best performance. The logarithmic random search optimization is employed since the logarithmic scale of the search allows for more efficient exploration of the search space. The magnitudes for Q with the lowest RMSE compared to ground truth is then used for the real data. The simulations are done using given IMU noise, offset in sensors, rotated a certain angle and estimated GNSS noise found from calculating the respective statistical parameters in Equation (2.11). The GNSS noise varies however, which means that the optimization can be improved further by using methods of noise modeling.

4

Evaluation and Results

This chapter revolves around presenting the results. First, we describe the methods used for evaluation. Then, the IMU calibration regarding removing the gravitation from the acceleration data is presented. Next, results on simulated data with respective errors are shown, mainly investigating the performance of different models under certain scenarios. Lastly, the results on the real-world data are presented, using the best models from the simulated results.

4.1 Evaluation

Since the goal is to improve the position tracking of the GNSS on the eBike, a visual interpretation can be used to evaluate whether the Kalman filter has proven to be successful. A ground truth can also be used as a comparison with the values from the Kalman filter output for evaluation. Tracking applications in a cell phone can be used to represent the position during a run with the bicycle. However, there is no certainty that these applications are accurate enough to provide a true value. Instead, simulations can be made to test the performance. While easier, due to the knowledge of the noise in the simulated data, it gives a rough estimate of how well the Kalman filter performs on real-world data.
Evaluations for the simulations are made by computing a Root Mean Squared Error, which shows how much of an improvement the Kalman filter output is in comparison to the GNSS data, with respect to the ground truth. The RMSE gives an output in meters, indicating the average distance from the predicted position to the true position, and likewise for the GNSS measurement. While this is an overall performance, an interest also lies in investigating different cases. For example, a filter with a specific dynamic model can have excellent accuracy when the trajectory is a straight line, but poor performance when turning. Similarly, the performance can be of high quality when the IMU coordinate frame is the same as the ground truth. However, when the frame is subject to a rotation, the performance can instead be poor. A model with good performance should be robust when the data is subject to different disturbances. These disturbances include different types of noise, shorter

and longer trajectories that can take any shape or form, but also when the IMU data is rotated or subject to gyroscope drifts and accelerometer offsets. If all of these still give an improvement in comparison to the GNSS, then the conclusion can be drawn that the Kalman filter improves the accuracy.

After evaluating the models on simulated data, the testing continues on real-world data. Coordinates were picked by hand using satellite images, and a chosen path was then closely followed with the eBike. The points picked were sparse, meaning that there were a lot more readings from the sensors than true values. As compensation, interpolation was used between the points to generate a path that ideally represents the true path. This path was then used to calculate the RMSE by taking the shortest distance from all of the interpolated points to the measured coordinate point. Further, a Mean Absolute Percentage Error was also calculated for these runs, yielding another metric for comparisons in performance. This, since MAPE has a better robustness when subject to outliers, which were common for the real-world scenarios [Allwright, 2022].

A problem with this evaluation method that appears is the loss of the time aspect. Instead of having time-synced coordinate points between the Kalman filter/GNSS and the satellite points, only the distance is considered. Because of this, some information is lost and therefore this does not correspond to a ground truth in the same sense as the simulated data does, but it is sufficient for the testing purposes of this thesis.

4.2 Calibration

Calibrations are mainly made to remove the gravity component of accelerometer measurements, but also to enable the coordinates of the IMU to be comparable to those of the GNSS. Viewing Figure 4.1, it is visible how the different axes of the accelerometer have been rotated correctly, as the acceleration in the z-direction is centered at 0 at all times. The data remains noisy however, meaning that methods of noise estimation and recalibration in each iteration of the Kalman filter could prove effective in improving the result further.

4.3 Simulations

Considering the generated data mentioned earlier, the filter was simulated to give a response of how accurate the performance was. The Kalman filter was evaluated on the data consisting of acceleration, velocity, position, the angle of direction as well as the angular velocity. The performance is then calculated by computations of a root mean squared error between the GNSS measurements and the ground truth, as well as the Kalman filter estimate with the ground truth. This gives an

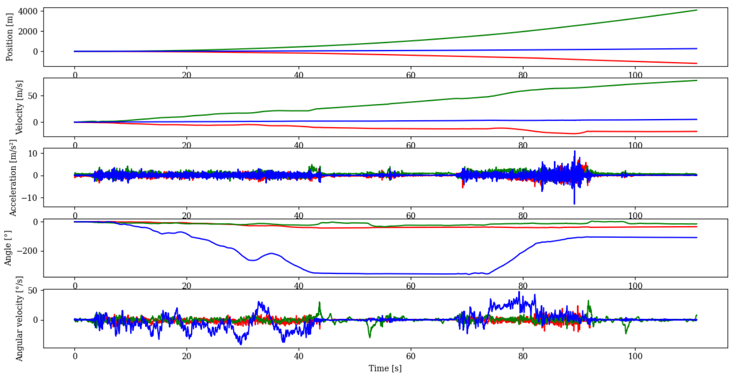

Figure 4.1 From the top: Position, velocity, and acceleration in x, y, and z from the accelerometer, as well as the angle and angular velocity around x, y and z from the gyroscope. Red-x, green-y, blue-z.

average error in both x and y, which in turn can be used to yield an average error. A smaller root mean squared error indicates a better-performing Kalman filter for the case, consisting of a specified trajectory and chosen noise. Testing this for several trajectories, noise models and IMU rotations then gives a general performance for each of the model dynamics. The best-performing dynamical model will then be used for the real-world data.

The data consisted of seven different trajectories. These were chosen to represent realistic scenarios, to get the highest chance of successful performance on the eBike. The data consisted of:

- A simple trajectory. This is the basic case, consisting of a path with a small curvature over time.
- A curved trajectory. This represents how an eBike can be ridden in real life, with sharper turns indicating how the bike can turn at crossroads or on curved paths.
- Rotated trajectories. The IMU is mounted on different eBikes in different ways, which means that the coordinate frame is rotated in most cases. The filter must be able to handle these rotations to be functional on real-world data.
- A trajectory with drift. The gyroscope has a tendency to drift, and the filter must be able to function regardless of drift in any capacity. After a rotation has been made on the IMU coordinate frame, drift is added to the gyroscope.

- A trajectory containing offset. While calibrations have been made to remove the offset from the accelerometer, the filter must still be able to handle small offsets that can be remaining.

A dynamic model able to handle all of the cases is considered to be functional and can then be used on real-world data.

Simplest trajectory

The simplest trajectory consisted of a path containing a slight curvature, but in other regards mostly straight. The IMU was left to be similar to the ground truth, apart from added noise, i.e. there was no initial rotation on the IMU data. The path was subject to small IMU drifts over time, but short enough for these drifts not to be destructive of the performance.

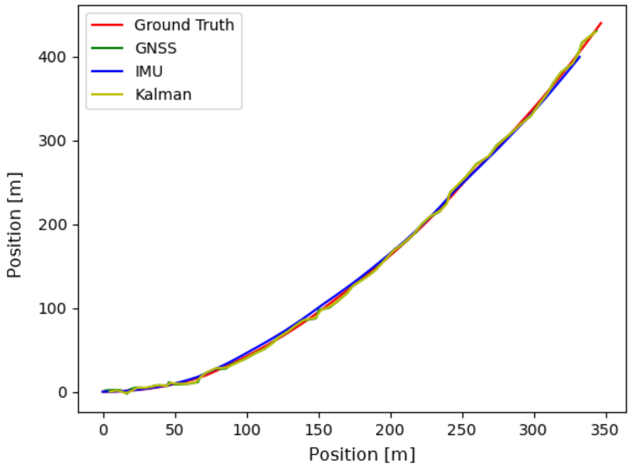

Figure 4.2 The simplest trajectory tested, with respective IMU, GNSS, and Kalman filter outputs. Axes showing the position in x and y.

The RMSE for different models subject to mentioned noise models were varying between model dynamics, but were still similar enough for these results to not be conclusive. The values can be seen in Table 4.1.

Trajectory with increased frequency of turns

The second trajectory tested was similar to the simple one mentioned previously, but the difference was that it turned more. The data was generated using functions dependent on a random angle. With an increase in the span of the angle, the frequency of turns increased. Some of the turns were also sharper, imitating realistic

Table 4.1 RMSE for the simplest trajectory, in meters.

Noise \ Models	Linear	First-Order	Second-Order	GNSS
Gaussian	0.21	0.74	0.79	0.96
Poisson	0.34	1.27	1.31	1.33
AR	0.49	2.13	2.10	1.90
ARMA	0.69	2.46	2.44	2.75
Average	0.43	1.65	1.66	1.74

turns at crossroads.

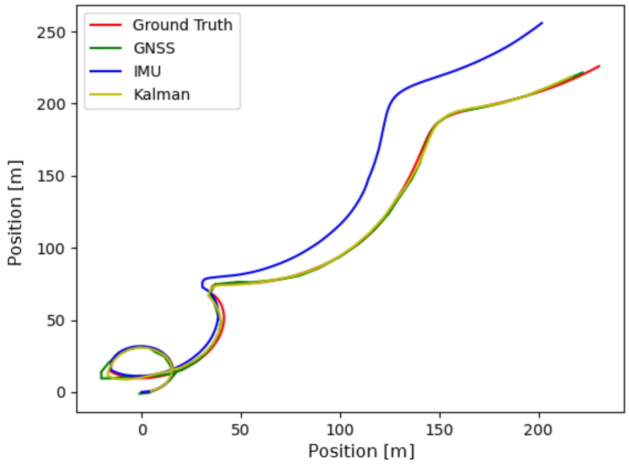

Figure 4.3 The second trajectory tested, consisting of more frequent and sharper turns, with respective IMU, GNSS, and Kalman filter outputs. Axes showing the position in x and y.

For ideal results on real-world data, the Kalman filter handles the turns with grace. The resulting values can be viewed in Table 4.2.

Rotated trajectory

The goal of the rotated trajectory was to see how the filter handled the case of a rotated IMU coordinate frame. In other words, the path could be the exact same as the previous ones, but with a rotated IMU. This simulates a more realistic scenario since the frames of the real-world data in most cases are not aligned perfectly.
The tests were separated into three different cases. One with a slight rotation, 30°, another with a quarter rotation, 90°, and lastly the worst case scenario, i.e., when

Table 4.2 RMSE for the trajectory containing an increased number of turns, in meters.

Noise \ Models	Linear	First-order	Second-order	GNSS
Gaussian	0.23	1.08	1.31	1.02
Poisson	0.43	1.90	1.37	1.46
AR	0.33	1.49	1.48	1.58
ARMA	0.45	2.06	2.31	2.69
Average	0.38	1.63	1.61	1.69

the IMU was rotated 180° from the GNSS frame.

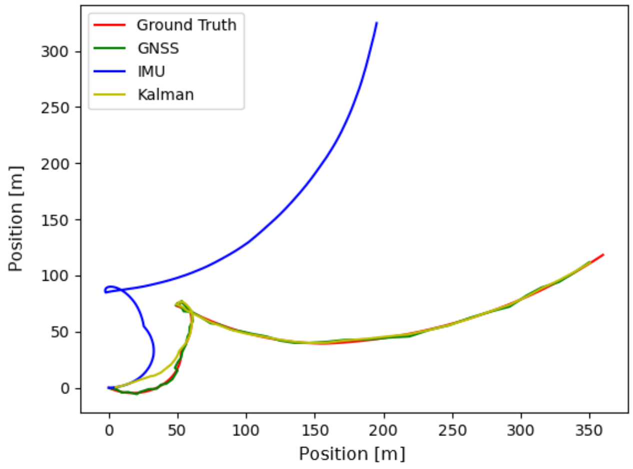

Figure 4.4 The first of the third batch of simulations, focusing on the rotation of the IMU coordinate frame, here 30°. Visible in the figure is that the Kalman filter converges after a certain amount of time. This is recurrent in all tests containing IMU rotations. Axes showing the position in x and y.

The models were tested with the same noise models as the previous simulations to achieve a good approximation of which model handles coordinate frame rotations the best.
While more similar than before, some of the models still proved to be a better alternative than the lone usage of the GNSS. The linear model is significantly worse compared to the previous results without a rotation, due to the negligence of the initial heading direction, see Table 4.3.

Table 4.3 RMSE for a 30° IMU rotation, in meters.

Noise \ Models	Linear	First-order	Second-order	GNSS
Gaussian	74.97	0.83	0.83	1.00
Poisson	75.09	1.42	1.40	1.45
AR	74.82	1.50	1.49	1.90
ARMA	74.89	2.18	2.19	2.33
Average	74.94	1.48	1.48	1.67

The significance of adding an angular state in the filter was enhanced further when the IMU frame was rotated additionally, see Table 4.4.

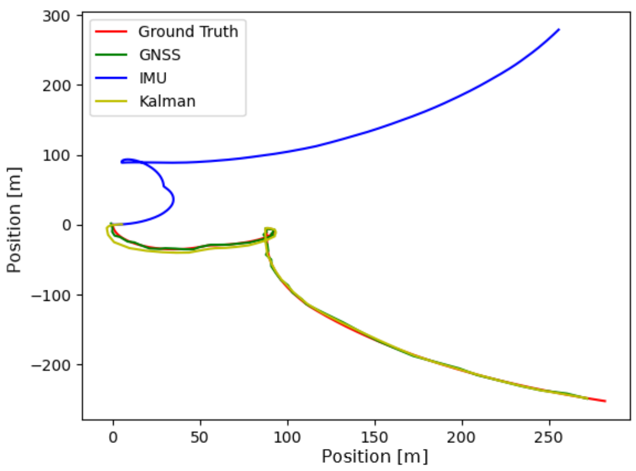

Figure 4.5 The second of the third batch of simulations, focusing on the rotation of the IMU coordinate frame. Here, a 90° rotation is displayed. Axes showing the position in x and y.

Further, the first and second-order models proved to be performing similarly, regardless of how much the IMU frame was rotated, see Table 4.5.

Simulated gyro drift

Next, a drift in the gyro data was added to examine how the different models were able to handle it. A filter is not able to handle real-world data without being robust to drift. The drift was added gradually over time in a linear fashion, trying to imitate what the real-world data looks like.
The models were able to handle the drift well. All in all, the first and second-order

Table 4.4 RMSE for a 90° IMU rotation, in meters.

Noise \ Models	Linear	First-order	Second-order	GNSS
Gaussian	170.96	0.76	0.72	0.93
Poisson	171.30	1.43	1.43	1.52
AR	170.98	1.45	1.46	1.46
ARMA	170.50	2.43	2.44	2.40
Average	170.94	1.52	1.51	1.58

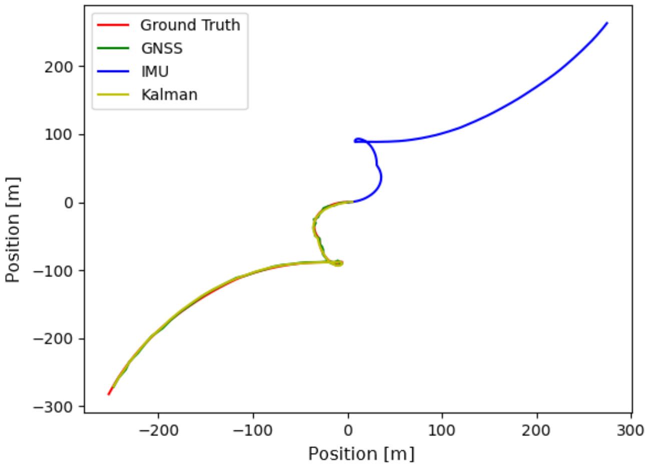

Figure 4.6 The last of the third batch of simulations, focusing on the rotation of the IMU coordinate frame. Here, a 180° rotation is displayed. Axes showing the position in x and y.

Table 4.5 RMSE for a 180° IMU rotation, in meters.

Noise \ Models	Linear	First-Order	Second-order	GNSS
Gaussian	301.58	0.95	0.89	1.02
Poisson	302.25	1.49	1.25	1.40
AR	301.70	1.57	1.67	1.68
ARMA	301.04	2.91	3.01	2.92
Average	301.65	1.73	1.70	1.75

models are able to handle all of the different cases, while the linear model mainly struggles with rotations of the IMU frame, see Table 4.6. Due to this, the linear model is not chosen to be used on real-world data, as it only works when there is a perfect alignment between the coordinate frames.

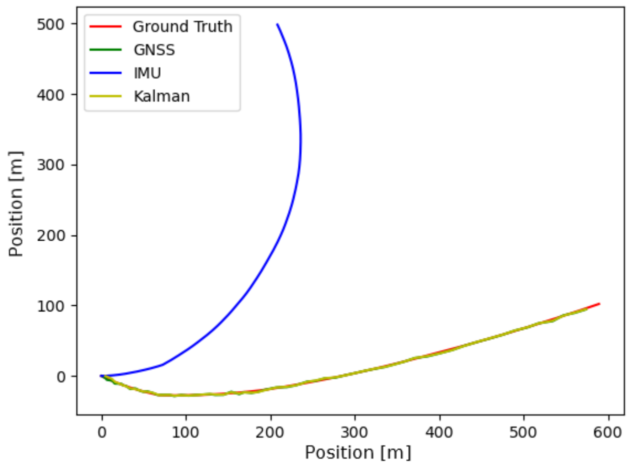

Figure 4.7 Trajectory with gyro drift added, with respective IMU, GNSS, and Kalman filter outputs. A rotation of 30° of the IMU was also added. Axes showing the position in *x* and *y*.

Table 4.6 RMSE for a simulated case containing gyro drift, in meters.

Models / Noise	Linear	First-order	Second-order	GNSS
Gaussian	75.01	0.79	0.79	0.94
Poisson	74.81	1.60	1.56	1.62
AR	75.02	1.54	1.56	1.56
ARMA	75.06	2.71	2.72	2.65
Average	74.97	1.66	1.66	1.69

Simulated offset

While a calibration was made to remove the offset from the sensors, there is no certainty that they are completely unbiased. Investigations are therefore made to see how the filter handles offset. In this case, the offsets are set to large values ($\geq 1\ \mathrm{m/s^2}$) which are unrealistic after calibrations. This offset causes the IMU position to increase rapidly in value. If the filter remains close to the ground truth, it means that it is robust to offset changes.

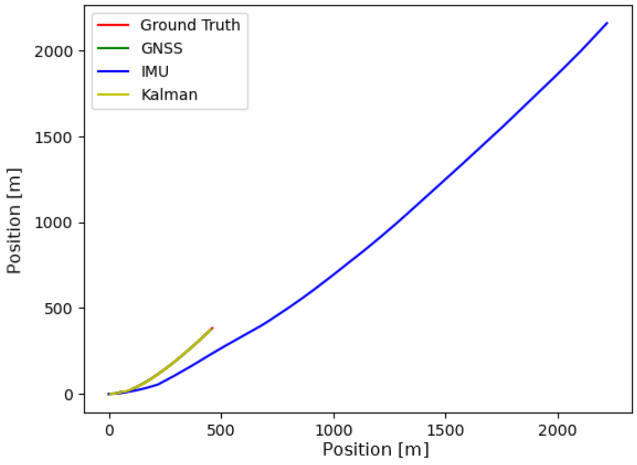

Figure 4.8 A trajectory tested with an accelerometer offset of $\geq 1\ \mathrm{m/s^2}$, with respective IMU, GNSS and Kalman filter outputs. Axes showing the position in x and y.

Given the amount of offset, it is sufficient that the filter RMSE stays relative to the one of the GNSS.

Table 4.7 RMSE for the case containing a large added offset, in meters.

Models / Noise	Linear	First-order	Second-order	GNSS
Gaussian	736.43	0.87	0.79	0.92
Poisson	736.80	1.76	1.76	1.53
AR	736.15	2.19	2.07	2.05
ARMA	736.23	2.53	2.57	2.61
Average	736.40	1.84	1.80	1.78

Visible from Table 4.7, the second-order model handles the offset better than the first-order model. However, the first-order model remains similar to the GNSS, and appears reliable. The linear model performs worse than the case subject to IMU rotations.

Parameter optimization

Three random searches were used to find the optimal magnitude of the Q parameters. The first random search was done using an IMU rotated angle of 30°, the second 90°, and the third 180° to ensure that direction does not affect the parameters. In all three cases, 100000 different combinations of power of tens were tested and the test with the lowest RMSE compared to the ground truth was kept and used for the real data. In all of the tests, the values that worked best were the magnitude of

$$\theta << x = y << v. \tag{4.1}$$

The ranges of the parameters were $\theta \in [10^{-5}, 10^{-3}]$, $x = y \in [10^{-2}, 1]$ and $v \in [10, 10^3]$. In other words, a larger trust was put in the angle retreived from the IMU compared to the positions, which in turn was more trustworthy than the velocity measurements.

4.4 Real-world data

The models were later tested on real-world data collected from different paths. The paths were chosen to represent scenarios where bicycles are commonly ridden. In total, there are four different real-world tests:

- The first test is a longer, generally straight path containing a few turns. It took place on a smaller road on the pavement.
- The second test was shorter and more curvy, inside of a park. Due to the surrounding environment such as trees, bushes, etc., the GNSS data is not as reliable in comparison to a scenario out in the open.
- The third path was in a suburban area containing more turns. Apart from the turns, the path was mostly straight.
- The fourth path was around a park on a gravel road, representing a circular path. This can represent a realistic scenario of riding around a roundabout.

Ideally, the filter output improves the performance for all of the runs. However, due to the satellite images used to interpolate an ideal path being out of date, construction, roadblocks, etc., may affect the results.

First-order model

The first-order model observed a slight improvement in accuracy in comparison to the GNSS with respect to the interpolated satellite points, on all test runs. The most noticeable changes are on tests 3 and 4, while tests 1 and 2 only gave a small improvement.

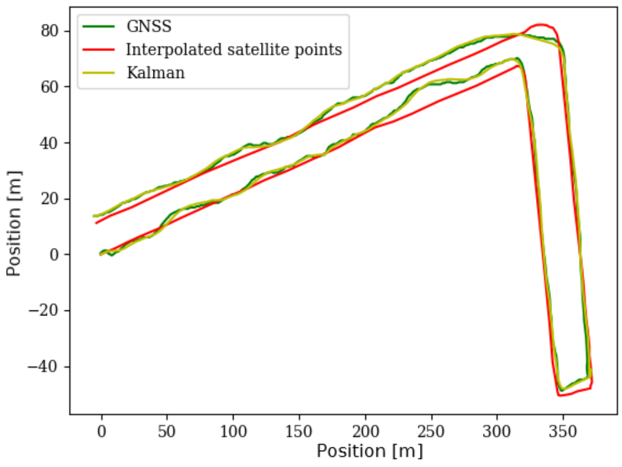

Figure 4.9 Test 1 for the first-order model tested on real GNSS and IMU data. Axes showing the position in *x* and *y*.

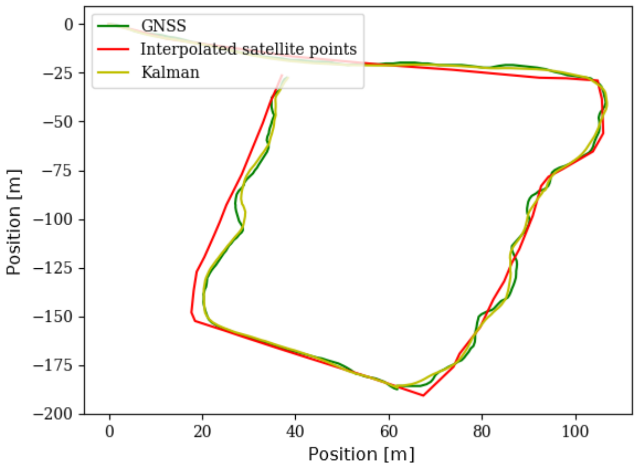

Figure 4.10 Test 2 for the first-order model tested on real GNSS and IMU data. Axes showing the position in *x* and *y*.

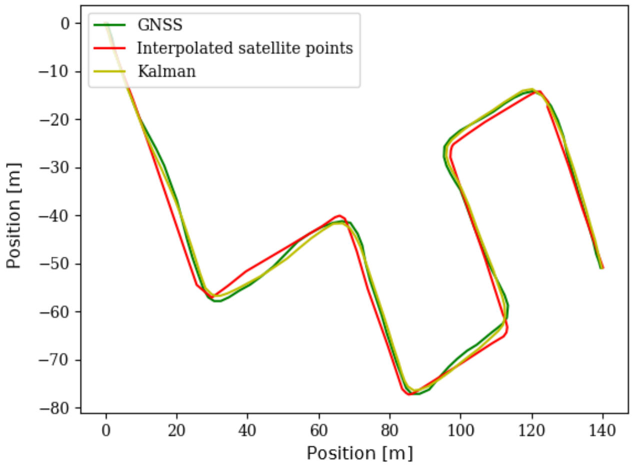

Figure 4.11 Test 3 for the first-order model tested on real GNSS and IMU data. Axes showing the position in *x* and *y*.

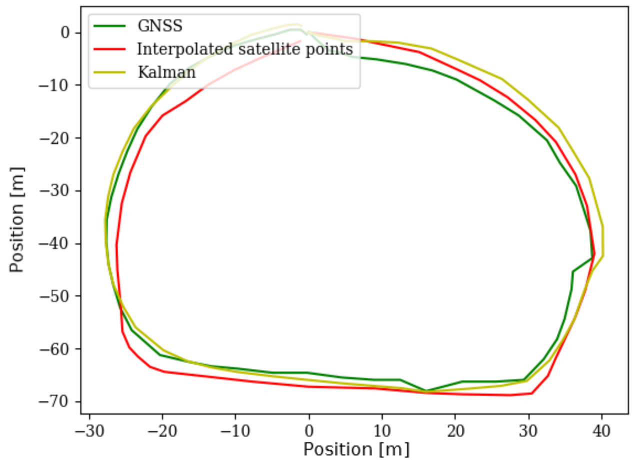

Figure 4.12 Test 4 for the first-order model tested on real GNSS and IMU data. Axes showing the position in *x* and *y*.

The tests visible in Figures 4.9-4.12 gave the results shown in Table 4.8.

Table 4.8 RMSE and MAPE for the first-order model on different real-world data sets, in meters.

Test \ Models	EKF RMSE	EKF MAPE	GNSS RMSE	GNSS MAPE
1	1.44	$6.34 \cdot 10^{-3}$	1.44	$6.17 \cdot 10^{-3}$
2	1.24	$7.02 \cdot 10^{-3}$	1.24	$6.54 \cdot 10^{-3}$
3	0.96	$6.59 \cdot 10^{-3}$	1.00	$7.56 \cdot 10^{-3}$
4	1.22	$2.18 \cdot 10^{-2}$	1.30	$4.01 \cdot 10^{-2}$

The result from the first-order model observes a slight improvement in the accuracy in all cases in comparison to only using the GNSS. While it handles some curves better, other curves are handled worse, which leads to similar performance. Tests 2 and 4 show this more specifically, the IMU tries to move away from the GNSS points but returns as soon as new data points are gathered from the GNSS. In test 3, the filter is able to handle more straight paths better than the GNSS. The most significant improvement in accuracy is for test 4 where the Kalman filter seems to have an easier time following the curvature of the path better than the GNSS.

Second-order model

While the first-order model proved to increase the performance of all scenarios, the second-order model improved the accuracy even further for tests 2 and 4.

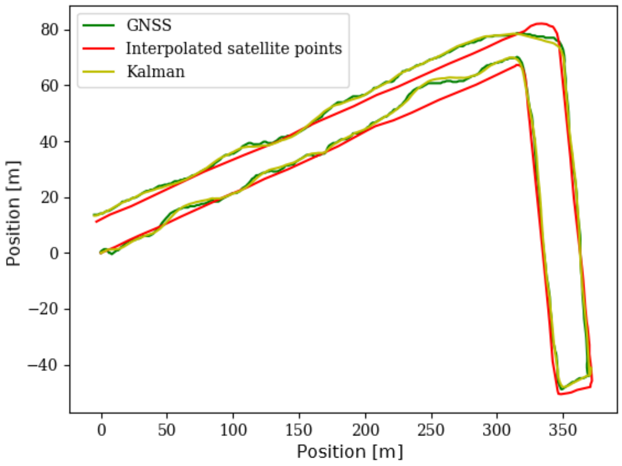

Figure 4.13 Test 1 for the second-order model tested on real GNSS and IMU data. Axes showing the position in x and y.

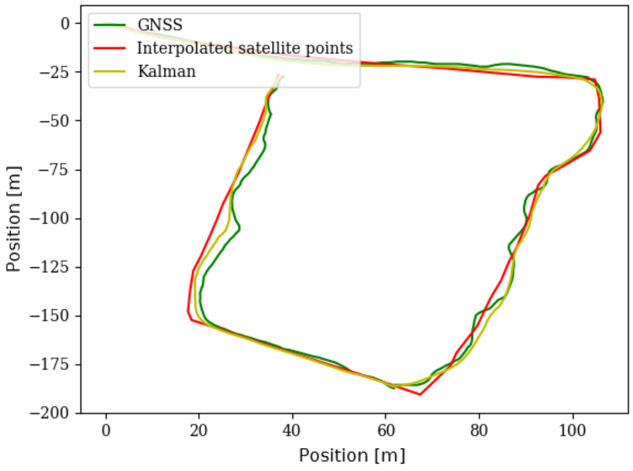

Figure 4.14 Test 2 for the second-order model tested on real GNSS and IMU data. Axes showing the position in *x* and *y*.

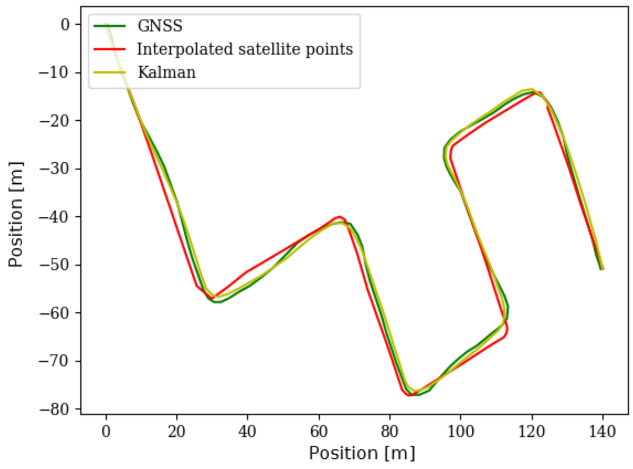

Figure 4.15 Test 3 for the second-order model tested on real GNSS and IMU data. Axes showing the position in *x* and *y*.

The tests visible in Figures 4.13-4.16 gave the results shown in Table 4.9.

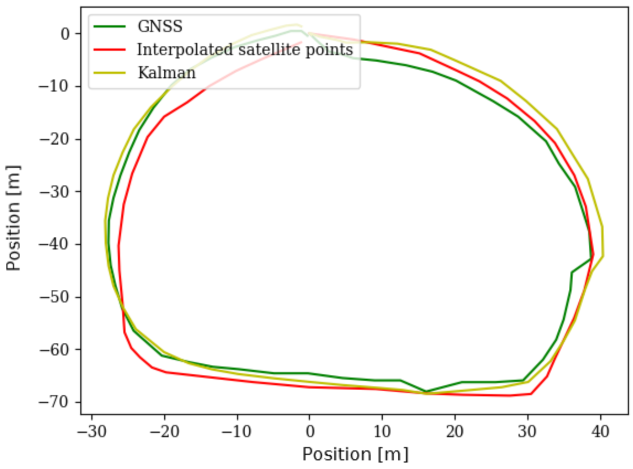

Figure 4.16 Test 4 for the second-order model tested on real GNSS and IMU data. Axes showing the position in x and y.

Table 4.9 RMSE and MAPE for the second-order model on different real-world data sets, in meters.

Models / Test	EKF RMSE	EKF MAPE	GNSS RMSE	GNSS MAPE
1	1.44	$5.87 \cdot 10^{-3}$	1.44	$6.17 \cdot 10^{-3}$
2	1.03	$4.39 \cdot 10^{-3}$	1.24	$6.54 \cdot 10^{-3}$
3	1.00	$6.79 \cdot 10^{-3}$	1.00	$7.56 \cdot 10^{-3}$
4	1.21	$2.18 \cdot 10^{-2}$	1.30	$4.01 \cdot 10^{-2}$

For test 1, the accuracy is not improved noticeably, similar to the first-order model. Some slight variations and smoothing of the curves can be seen, but in general, the results are the same as only using the GNSS. In test 2, an increase in accuracy can be seen both in the metrics and also in the figure. At the end of its path (from the coordinate points $(20, -150)$ m to $(40, -25)$ m) the Kalman filter follows the curvature of the interpolated satellite path more reliably. Test 3 also improved the accuracy by a very tiny bit, some curvatures seem to follow the satellite points better, but not by much. Similar to the first-order model, straight paths are easier to make out using the Kalman filter data. Test 4 performs very similarly to test 4 for the first-order model, i.e., the Kalman filter sees a slight increase in accuracy compared to the GNSS.

The tests on real-world data uses all of the interpolated points for evaluation, hence leading to a heavy computational time. In a real-time setting, the computations are made on one point individually, leading to a faster computational time. This means that it can not be decided for certain if the algorithm works in an online setting. However, for this to be possible, the computational time has to be ≤ 1 second, as the GNSS sample is set to 1 Hz. Otherwise, the computations would not be finished once a new measurement arrives, meaning that the filter would not work for a stream of values received continuously.

5

Discussion and Conclusion

In this chapter, the previously presented results are discussed. First, some discussion about the developed calibration is presented. Second, the dynamic models as well as the performance are evaluated. Next, some different subjects that could expand this thesis are introduced, including additions as well as alternative methods. Lastly, final conclusions are presented.

5.1 Calibration

A large part of the accuracy of the Kalman filter output is in how good of a calibration was made in the pre-processing. If the calibration is perfect, the Kalman filter can choose to follow the IMU all of the time, yielding an excellent performance. However, this is speculative, as the calibration in this thesis can be improved and there is no proof that the performance will be perfect. If the calibration is worse, the Kalman filter will instead follow the GNSS, yielding a very similar accuracy to what was already existing beforehand.

5.2 Evaluations

The thesis revolves around evaluating different models and their respective performances on different simulations and test runs. Naturally, the main discussion revolves around which model has the best performance.

Model evaluation

The thesis has evaluated three types of model dynamics. The linear dynamics had the best performance of all the models on simple simulations containing only noise. When rotations were added, the model was not able to follow where the true trajectory went, due to a lack of the angle as a state. With this questionable performance, the linear model was disregarded in terms of running on actual real-world scenarios. If there was a possibility to mount the sensor on the eBike with a perfect alignment with the heading direction, the linear model could arguably work on actual data. The model may also even be preferred to the first and second-order models, given the performance in simulations for that scenario. However, considerations must also be made on gyro drift, as well as offsets, which clearly deteriorated the performance.

By comparing the first and second-order models solely on simulated results, one could argue that they have the same performance. While the first-order model is able to perform slightly better on the simple paths, the second-order model handles the specific cases of rotation, drift and offset slightly better. Both of the models improved the accuracy in every case except for large offsets. However, the latter was disregarded given that calibrations of both accelerometer and gyro data reduce this offset to almost zero. Perfect calibration is difficult to implement in practice, so the filter should still be able to handle the offsets and drifts, which both of the models do to a certain degree. Furthermore, a combination of a difficult path to interpret, large offsets and drifts, as well as a bad alignment between IMU and GNSS frames also decreases the performance, where the offset is the deciding factor between using only the GNSS, or relying on the Kalman filter.

Both of the dynamic models with good simulation performance were then tested on real-world data, and the accuracy had similar tendencies as for the simulations. Depending on the scenario, the first and second-order models took turns in which had the best performance. Both of the models outperformed the GNSS, which is the main goal. However, this improvement is barely noticeable for test 1 and 3, while it is more apparent for tests 2 and 4. The most probable cause of this is that the GNSS had a worse accuracy when tests were done in parks, due to trees distorting the satellite signals.

Visible in Figures 4.9-4.12, the first-order model has trouble working independently of the GNSS and instead chooses to follow it closer in comparison to the second-order model. This could indicate that the first-order model is more sensitive to how well the pre-processing has been made. On the other hand, the second-order model has a higher accuracy despite subpar calibrations. Looking at the results in Tables 4.8, 4.9, there is a variation of which model works the best. It appears that the first-order model works better for open suburban areas where the GNSS has a good connection, while the second-order model works better in parks where trees can block the satellites. This is most likely due to the first-order model following the GNSS more, while the second-order model is operating in a more independent fashion.

Viewing Figures 4.13-4.16, one could see that the second-order model is able to work better without relying on the GNSS signals as much as the first-order model. While this can yield a worse performance when the GNSS has accurate position measurements, visible for the third test in Tables 4.8, 4.9, it can also mean that the model handles the loss of signals better. Inaccurate positions from the GNSS also seem to be handled better than for the first-order model, visible in Figure 4.14. This could mean that the second-order model potentially has a better performance than both a lone GNSS usage, but also the first-order model, in dead-reckoning scenarios. Depending on how long these scenarios are, however, the second-order model would also start to drift off and give large inaccuracies.

Performance evaluation

The performance of filters was evaluated using a Root Mean Squared Error average for simulations, indicating an average error of how close to the actual true position the filter output was. For simulations containing accelerometer offsets, which gave the worst performance, the best-performing model was still able to stay within 1.8 meters of the ground truth on average, slightly better than the GNSS position every time. Also visible in the results is how the performance of both GNSS and filter outputs decreased when the noise models became more complex. Gaussian noise was chosen as the basic case, given that it is the most common process. The other models were then chosen to be an evolution of the Gaussian process. Poisson noise can be approximated by a Gaussian process with the same mean and variance for certain cases, while the AR and ARMA processes are linear filter combinations of Gaussian realisations. The last two worsened the performance further, but the filter output still improved the performance in comparison to using the GNSS, as the GNSS performance also decreased for these noise models.

Further, adding rotations of the IMU saw the filter output being off in the beginning. After several iterations, it converged to the ground truth. In other words, the GNSS was significantly better at tracking the position in the beginning. After a certain amount of time, the filter became the clear choice for finding the position closest to the ground truth. The time for convergence differed depending on the quantity of the initial rotation. A small rotation had a faster time for convergence, while rotations of the IMU up to 180° took longer.

When looking at the performance for real scenarios, the difference between filter outputs and GNSS measurements is not as different as for the simulations. However, the filter outputs are better on average for every test, which compared to the simulations is an improvement. The RMSE for the Kalman filter in simulations gave a slightly worse performance than for the GNSS at times, most likely because of bad parameter tuning. With optimized parameters in the ranges mentioned at the end of Section 4.2, the filter outperformed the GNSS for all scenarios.

The first-order model barely outperformed the GNSS according to the RMSE, mainly because of it following the GNSS for most of the run. However, the MAPE showed that the first-order model was worse than only using the GNSS for tests 1 and 2. The second-order model was able to distinguish errors better and reduced the constant difference between GNSS and the interpolated satellite points, visible at the end of Test 2 (Figure 4.14).

Since both of the dynamic models improve the results, both of them can be used to reliably track the position, while the second-order model is preferred thanks to its consistency. However, the test runs made were simple, and mainly out in the open. There is no knowledge of how the implementation works when subject to dead-reckoning scenarios. If the GNSS loses its signal, the models will start to only follow the IMU, leading to potential drifts. The drifts cannot be accounted for, as there is no GNSS signal to reset to. The solution to this circles back to the IMU calibration, as the better calibration will negate the potential drifts and give a more reliable tracking of the position when the tracking object is in tunnels or buildings.

Despite being very similar in theory, RMSE and MAPE seem to yield different results for tests 1 and 2 for the first model. A slight improvement can be seen for the RMSE in test 1 compared to the GNSS, meanwhile, the MAPE seems to be a little bit better for the GNSS. The same goes for test 2, where the same disagreement occurs. This contradiction where the RMSE and MAPE give different results could be explained by a variety of factors. Since the values are very close to each other this could suggest that the first-order model does not improve the performance of these tests. Another explanation could be the fault of outliers affecting the RMSE calculation of the GNSS more than the MAPE. Generally speaking, RMSE is more sensitive to outliers compared to MAPE which might have caused the inconsistency [Allwright, 2022]. Whatever the cause, the second-order model seems to handle this problem more desirably, i.e., all tests where the RMSE is improved, the MAPE is also improved.

5.3 Future work

Real-time implementation

This thesis focuses on tracking the position of an electric bicycle in an offline setting, disregarding computational efficiency and energy savings. To enable position tracking of this kind in a real-time setting, the algorithm must be rewritten with a focus on efficiency, with a perspective of both processing power and battery usage. The algorithm, written in Python code, should in this case be rewritten in a language that in general is more efficient for computations, such as C/C++ or Rust. Python was chosen because of its simplicity and flexibility, but it is, unfortunately, one of the worst performing programming languages in the perspective of time savings [Ludvigsen, 2022]. With an implementation in a more efficient language, the algorithm can be run in a real-time setting, enabling online position tracking of eBikes remotely. While the computations made are time inefficient, doing predictions after each new measurement instead increases the chance of a real-time implementation being possible. However, there is still no certainty that the current algorithm is fast enough for the GNSS sample rate, and it may all in all therefore have to be rewritten.

Improved IMU calibration

In this thesis, the focus lies on improving position-tracking using a sensor fusion framework consisting of an IMU and a GNSS. While efforts were made in calibrating both sensors, the IMU still experienced bias which causes integration errors. If the sensors are better calibrated, the performance of the position-tracking algorithm could improve by a significant margin. This is because greater faith can be laid in the IMU measurements without worrying about the drifts caused by biases decreasing the overall accuracy. Having poor IMU measurements with respect to the ground truth, the Kalman filter will mainly trust the GNSS. This in turn will barely give an improvement of the accuracy.

Since the calibration involves rotating the IMU coordinate frame into a frame that has the z-direction directly upwards, gravity can also affect the biases of the sensors and especially when integration drift in the gyro happens as well. After some time, the gyro can drift enough so that components in the gravity vector affect all of the orthogonal directions instead of just the z-direction. These effects can be seen in Figure 4.1, where both the gravity vector and sensor biases cause large integration errors over time. Re-calibrating the data once every time a new GNSS sample is retrieved can help circumvent this problem. This is done in the Kalman filter and prevents the gyro from ever drifting away as the angle is always readjusted back so that the components of the gravity vector do not interfere with the other directions. The drawback is that if the GNSS has a lower sampling rate, the re-calibration process can become so infrequent that drift in the integration step still happens.

A more elegant solution would be to use a 9-degree-of-freedom IMU (i.e. adding a magnetometer to the already existing accelerometer and gyroscope) for the gravity compensation. The magnetometer is used in conjunction with the gyro to make sure that the coordinate frame of the IMU always stays as close to the one where z faces directly upwards. Here, calibration of the biases is even more important since a bias in the magnetometer and the gyro can cause even more drift in the orientation compared to before. The consensus is that more effort has to be put into the calibration for the process to work, but will yield a more accurate result once the sensors are sufficiently calibrated. Another problem with using a magnetometer is that it comes with new challenges in handling noise. Magnetic noise or vibrations that electric machines radiate can cause the data from the magnetometer to become untrue which in turn leads to worse performance than only using the gyro.

Handling dead-reckoning scenarios

To handle dead-reckoning scenarios means being able to maneuver environments where the GNSS sensor does not get any readings at all for a while. That means that all of the positioning is put in the hands of the IMU. An example where such a scenario can happen is if the sensors are driven inside of a basement or parking garage. Here, the GNSS will not be able to give any information about the whereabouts of the eBike. With enough focus put into the calibration (making sure that drifts and errors do not escalate quickly) the IMU can be used for a limited time to accurately describe the location of the bike. In the case of this thesis, the calibration just is not sufficient enough to give any valuable information even after a few seconds inside a dead-reckoning environment.

Alternative algorithms

The main goal of this thesis was to improve the accuracy of position tracking, regardless of potential constraints. Hence, the Extended Kalman filter was used thanks to its simplicity. Alternative methods can be considered in the future, such as Unscented Kalman filters, Particle filters, and also the usage of Neural Networks. These algorithms are not guaranteed to have a better performance than the EKF, but some have proven to be more efficient and robust [Wan and Van Der Merwe, 2000]. The UKF is able to handle different types of process disturbances and measurement noise in comparison to the EKF. Further, the UKF can be viewed as an efficient and selective PF. The PF uses a Monte Carlo scheme for predictions, which can be incredibly powerful, but also computationally heavy. If one were to investigate alternative filter approaches, the UKF tends to be more accurate, and computationally efficient, in comparison to the PF [Wan and Van Der Merwe, 2000], [Schiff, 2009]. Learning-based control algorithms are also an alternative. However, these require data in large amounts for the learning process. Further, this data is required to have a known ground truth, which has proven unreliable in this thesis.

Higher-order models

The order of models in this thesis was limited to two, as a result of the IMU only being able to measure acceleration and nothing of a higher derivative. If a sensor would be able to measure jerk, the derivative of acceleration, a third-order model can be used to possibly increase the accuracy even further. A fourth-order model can also be achieved by using a sensor able to handle snap, the derivative of jerk, etc.
There is also a possibility to generate jerk, snap, etc by numerically differentiating the acceleration and further derivatives. However, this leads to a loss of data points, as two data points are required for differentiation, and there is also an amplification of noise in the measurements. A recommendation would instead be to investigate sensors able to measure these quantities.

Noise modeling of GNSS and optimization

During this thesis, different noise models were used to test the robustness of the implemented EKF, since the GNSS noise was being unknown. However, modeling could also be made for said noise. This can be done by using a Deep Neural Network, or alternative Kalman filters which have been mentioned previously. While attempts have been made to estimate the GNSS noise, the model is still inconclusive. Finding an accurate model of the GNSS noise could yield the possibility of using optimization algorithms to find the best possible Kalman filter parameters, increasing the accuracy further.

Use of more accurate satellite coordinates

Better accuracy and more conclusions can be drawn with more precise satellite coordinates. The coordinates are used in the RMSE calculation for real-world data. In a sense, these can be viewed as the eBikes actual position that the data from the Kalman filter and the GNSS are compared to for performance evaluation. If these coordinates are misplaced, the information from the evaluation can become meaningless and too vague to make any kind of conclusion. For this study, the points were picked using Google Maps, which provides a decent accuracy with respect to the real world. However, the human factor still plays its part as these coordinate points were hand-picked from the Google Maps platform. A better alternative would be to use a more detailed GNSS with less noise, a similar sampling rate, and better precision that can be used as ground truth. This way, interpolation will not be needed since the sampling rates are synced which in turn will likely lead to more correct calculations of the RMSE.

5.4 Conclusion

From this thesis, we conclude that there is a possibility to increase the performance of the position tracking. Depending on the situation, either model could be the best alternative to track the position. However, certain scenarios also show that the lone usage of the GNSS proves to be the best option.
The GNSS is already highly accurate, giving an average error distance of less than two meters. It is already a reliable source of position tracking, with exceptions for when satellite signals are blocked. This means that in general cases, the tracking may not need improvement at all, since the GNSS measurement is in close enough proximity to the actual position that the true position can be determined.
If the performance should be improved, however, one way is to use an Extended Kalman filter. The implementation does increase the accuracy, but only barely. The dynamic model preferred would be the second-order model, which consistently improved the performance.
However, if exact position tracking is expected, the EKF is not sufficient with the pre-processing done in this thesis. Better calibration has to be done on the IMU,

noise modeling of the GNSS can help to distinguish which model is the best, and alternative Kalman filters may also need investigation.

Bibliography

Allwright, S. (2022). *RMSE vs MAPE, which is the best regression metric?* URL: `https://stephenallwright.com/rmse-vs-mape/` (visited on 2023-05-30).

Årzén, K. E. (2014). *Real-time Control Systems*. Tryckeriet i E-huset.

Becker, A. (2023). *Kalman filter overview*. URL: `https://www.kalmanfilter.net/default.aspx` (visited on 2023-03-22).

Bergstra, J. and Y. Bengio (2012). “Random search for hyper-parameter optimization.” *Journal of machine learning research* **13**:2. (Visited on 2023-06-20).

Boyat, A. K. and B. K. Joshi (2015). “A review paper: noise models in digital image processing”. *Military College of Tele Communication Engineering, Military Head Quartar of War, Ministry of Defence, Govt. of India, India*. URL: `https://arxiv.org/ftp/arxiv/papers/1505/1505.03489.pdf`.

Brockwell, P. J. (2001). “Continuous-time ARMA processes”. *Handbook of statistics* **19**, pp. 249–276. (Visited on 2023-06-20).

Campestrini, C., T. Heil, S. Kosch, and A. Jossen (2016). “A comparative study and review of different Kalman filters by applying an enhanced validation method”. *Journal of Energy Storage* **8**, pp. 142–159. ISSN: 2352-152X. DOI: `https://doi.org/10.1016/j.est.2016.10.004`. URL: `https://www.sciencedirect.com/science/article/pii/S2352152X16302031`.

Dam, E. B., M. Koch, and M. Lillholm (1998). *Quaternions, Interpolation and Animation*. Tech. rep. Department of Computer Science, University of Copenhagen. URL: `https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c0dff96ef25d3b1e744b8d3d7013a76efb64e502` (visited on 2023-03-16).

Deisenroth, M. P., A. A. Faisal, and C. S. Ong (2021). *Mathematics for Machine Learning*. Cambridge University Press. URL: `https://mml-book.github.io/book/mml-book.pdf` (visited on 2023-03-15).

Drake, S. P. (2002). "Converting GPS coordinates [phi, lambda, h] to navigation coordinates (ENU)". URL: https://www.researchgate.net/profile/Samuel-Drake/publication/27253833_Converting_GPS_coordinates_phi_lambda_h_to_navigation_coordinates_ENU/links/00b7d516ca79c24fdb000000/Converting-GPS-coordinates-phi-lambda-h-to-navigation-coordinates-ENU.pdf (visited on 2023-06-20).

Dudek, G. and M. Jenkin (2010). *Computational Principles of Mobile Robotics*. 2nd ed. Cambridge University Press. DOI: 10.1017/CBO9780511780929.

EUASP (2023). *What is GNSS*. URL: https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss (visited on 2023-01-18).

Fang, B., F. Sun, H. Liu, and C. Liu (2018). "3D human gesture capturing and recognition by the IMMU-based data glove". *Neurocomputing* **277**. Hierarchical Extreme Learning Machines, pp. 198–207. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2017.02.101. URL: https://www.sciencedirect.com/science/article/pii/S0925231217314054.

Fessler, J. A. (1998). *On Transformations of Random Vectors*. Tech. rep. Dept. of Electrical Engineering and Computer Science, Univ. of Michigan. URL: http://www.eecs.umich.edu/techreports/systems/cspl/cspl-314.pdf (visited on 2023-03-02).

Geotab-Team (2020). "History of GPS satellites and commercial GPS tracking". *Fleet Management*.

Glad, T. and L. Ljung (1997). *Relgerteori*. Studentlitteratur. ISBN: 978-91-44-03003-6.

Goldman, R. (2011). "Understanding quaternions". *Graphical Models* **73**:2, pp. 21–49. ISSN: 1524-0703. DOI: https://doi.org/10.1016/j.gmod.2010.10.004. URL: https://www.sciencedirect.com/science/article/pii/S1524070310000172.

Grahm, L., H.-G. Jubrink, and A. Lauber (1996). *Modern Industriell Mätteknik*. Bokförlaget Teknikinformation. ISBN: 91-88156-05-2.

Gustafsson, F. (2010). "Particle filter theory and practice with positioning applications". *IEEE Aerospace and Electronic Systems Magazine* **25**:7, pp. 53–82. DOI: 10.1109/MAES.2010.5546308.

Hasinoff, S. W. (2014). "Photon, poisson noise". In: *Computer Vision, A Reference Guide*.

Hemingway, E. G. and O. M. O'Reilly (2018). "Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments". *Multibody System Dynamics* **44**. DOI: 10.1007/s11044-018-9620-0.

Hofmann-Wellenhof, B., H. Lichtenegger, and E. Wasle (2008). *GNSS, Global Navigation Satellite Systems*. Springer Wien New York.

Hyyti, H. and A. Visala (2015). “A DCM Based Attitude Estimation Algorithm for Low-Cost MEMS IMUs.” *International Journal of Navigation & Observation*. (Visited on 2023-06-20).

Jakobsson, A. (2020). *An Introduction to Time Series Modeling*. 3:2. Studentliteratur. ISBN: 978-91-44-13403-1.

Lindgren, G., H. Rootzén, and M. Sandsten (2013). *Stationary Stochastic Processes for Scientists and Engineers*. CRC Press. ISBN: 9781466586185.

Lindholm, A., N. Wahlström, F. Lindsten, and T. B. Schön (2022). *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press. ISBN: 9789130045624. (Visited on 2023-02-18).

Ludvigsen, K. G. A. (2022). *The 10 most energy efficient programming languages*. URL: `https://kaspergroesludvigsen.medium.com/the-10-most-energy-efficient-programming-languages-6a4165126670` (visited on 2023-05-10).

Madgwick, S. O., A. J. Harrison, and R. Vaidyanathan (2011). “Estimation of IMU and MARG orientation using a gradient descent algorithm”. In: *2011 IEEE international conference on rehabilitation robotics*. IEEE, pp. 1–7.

Månsson, J. and P. Nordbeck (2011). *Endimensionell Analys*. Studentlitteratur. ISBN: 9789144056104.

Mapscaping (2023). *How phone tracking works: position, location gps*. URL: `https://mapscaping.com/exploring-how-cell-phone-gps-tracking-works/` (visited on 2023-06-20).

Møller, N. F. (2008). “Bridging economic theory models and the cointegrated vector autoregressive model”. *Economics* **2**:1. (Visited on 2023-06-20).

Mu, H. and R. Xiong (2018). “Chapter 1 - Modeling, Evaluation, and State Estimation for Batteries”. In: Zhang, H. et al. (Eds.). *Modeling, Dynamics and Control of Electrified Vehicles*. Woodhead Publishing, pp. 1–38. ISBN: 978-0-12-812786-5. DOI: `https://doi.org/10.1016/B978-0-12-812786-5.00001-X`. URL: `https://www.sciencedirect.com/science/article/pii/B978012812786500001X` (visited on 2023-06-20).

Park, K. I. (2018). *Fundamentals of Probability and Stochastic Processes with Applications to Communications*. Springer Cham. URL: `https://link.springer.com/book/10.1007/978-3-319-68075-0` (visited on 2023-03-01).

Passaro, V. M. N., A. Cuccovillo, L. Vaiani, M. De Carlo, and C. E. Campanella (2017). “Gyroscope technology and applications: a review in the industrial perspective”. *Sensors* **17**:10, pp. 294–295. ISSN: 1424-8220. URL: `https://www.mdpi.com/1424-8220/17/10/2284` (visited on 2023-02-01).

PAVE (2020). *PAVE Poll: Americans wary of AVs but say education and experience with technology can build trust.* URL: `https://pavecampaign.org/pave-poll-americans-wary-of-avs-but-say-education-and-experience-with-technology-can-build-trust/` (visited on 2023-03-28).

Premerlani, W. and P. Bizard (2009). "Direction cosine matrix IMU: Theory". *Diy Drone: USA* **1**.

Ribiero, M. I. (2004). "Kalman and Extended Kalman Filters: Concept, Derivation and Properties". *Institute for Systems and Robotics, Instituto Superior Técnico*. URL: `http://users.isr.ist.utl.pt/~mir/pub/kalman.pdf` (visited on 2023-02-26).

Sasiadek, J. (2002). "Sensor fusion". *Annual Reviews in Control* **26**:2, pp. 203–228. ISSN: 1367-5788. DOI: `https://doi.org/10.1016/S1367-5788(02)00045-7`. URL: `https://www.sciencedirect.com/science/article/pii/S1367578802000457`.

Schiff, S. J. (2009). "Kalman meets neuron: the emerging intersection of control theory with neuroscience". *Annual International Conference of the IEEE Engineering in Medicine and Biology Society.*, pp. 3318–3321. DOI: `doi:10.1109/IEMBS.2009.5333752`. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3644303/` (visited on 2023-03-25).

Shokri, S., N. Rahemi, and N. R. Mosavi (2020). "Improving GPS positioning accuracy using weighted Kalman Filter and variance estimation methods". *CEAS Aeronautical Journal*. DOI: `10.1007/s13272-019-00433-x`. URL: `https://link.springer.com/article/10.1007/s13272-019-00433-x#citeas`.

Slabaugh, G. G. (1999). *Computing Euler angles from a rotation matrix*. Tech. rep. City Univ. London, London, U.K. URL: `https://www.researchgate.net/profile/Thomas-Blesgen/publication/276852668_On_rotation_deformation_zones_for_finite-strain_Cosserat_plasticity/links/55fbe44808ae07629e07c60c/On-rotation-deformation-zones-for-finite-strain-Cosserat-plasticity.pdf` (visited on 2023-03-14).

Sleight, M. (2021). *How do self-driving cars work?* URL: `https://www.bankrate.com/insurance/car/how-do-self-driving-cars-work/` (visited on 2023-05-10).

Southallzy, B., B. Buxtony, and J. Marchant (1998). "Controllability and observability: Tools for Kalman filter design". In: *British Machine Vision Conference*. Vol. 98, pp. 164–173. (Visited on 2023-06-20).

Toll, M. (2022). *I just had (another) expensive electric bike stolen. here's what would have stopped it.* URL: `https://electrek.co/2022/09/20/i-just-had-another-expensive-electric-bike-stolen-heres-what-would-have-stopped-it/` (visited on 2023-05-12).

Trussell, H. J. and R. Zhang (2012). “The dominance of poisson noise in color digital cameras”. In: *2012 19th IEEE International Conference on Image Processing*. IEEE, pp. 329–332. (Visited on 2023-06-20).

Vectornav (2023). *Whas is an inertial measurement unit?* URL: `https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu` (visited on 2023-01-18).

Wan, E. and R. Van Der Merwe (2000). “The unscented Kalman filter for nonlinear estimation”. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pp. 153–158. DOI: `10.1109/ASSPCC.2000.882463`. URL: `https://ieeexplore.ieee.org/document/882434` (visited on 2023-02-26).

Watson, J. (2020). “MEMS Gyroscope Provides Precision Inertial Sensing in Harsh”. *High Temperature Environments*. URL: `https://www.analog.com/en/technical-articles/mems-gyroscope-provides-precision-inertial-sensing.html` (visited on 2023-06-20).

Weisstein, E. W. (2009). “Euler angles”. *https://mathworld. wolfram. com/*. (Visited on 2023-06-20).

Welch, G. F. (2020). “Kalman filter”. *Springer Nature Switzerland*.

Yeong, D. J., G. Velasco-Hernandez, J. Barry, and J. Walsh (2021). “Sensor and sensor fusion technology in autonomous vehicles: a review”. *Sensors* **21**:6. ISSN: 1424-8220. DOI: `10.3390/s21062140`. URL: `https://www.mdpi.com/1424-8220/21/6/2140` (visited on 2023-05-10).

Zabinsky, Z. B. et al. (2009). *Random search algorithms*. Tech. rep. Department of Industrial and Systems Engineering, University of Washington, USA. URL: `https://courses.washington.edu/inde510/516/AdapRandomSearch4.05.2009.pdf` (visited on 2023-05-11).

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER'S THESIS *Date of issue* June 2023 *Document Number* TFRT-6212
Author(s) Richard Byström William Sjödin	*Supervisor* Martin Heyden, Bosch Stefan Noll, Bosch Yiannis Karayiannidis, Dept. of Automatic Control, Lund University, Sweden Björn Olofsson., Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

GNSS Precision for Mobile Devices with Sensor Fusion Techniques: A Case Study on eBike Tracking Using State Estimation

Abstract

Electric bicycles have over time become a common method of transportation. With a rapidly increasing user base and expensive prices, there is also an increasing demand for safety and insurance. Bike safety can be used to notify the bicycle owner of a potential theft, or to locate the position of the bike when it is lost. Improving the autonomous vehicle's position tracking when its location is unknown to the owner, simplifies the search for a lost or stolen bike.
With the use of an accelerometer and gyroscope as input, coupled with a Global Navigation Satellite System, a prediction algorithm, or filter, was developed to predict the trajectory. Three different dynamical models for the filter were tested for robustness and optimization of filter parameters to yield desired results. An Extended Kalman Filter was used for the predictions, while the tested dynamic models were of linear, first-order and second-order types. The simulations used for the model evaluation utilized four different noise models. While the Inertial Measurement Unit contained known noise, the GNSS noise remained unknown.
When the tests with simulations indicated which models had the best performance, the filter was used on data from real-world measurements. Calibration was made on the Inertial Measurement Units' inner coordinate frame to get position estimates for comparisons with satellite points.
In some cases, the lone use of a GNSS for position tracking proved to be the best, while in other cases the filter output had a higher accuracy of predicting the position correctly. On average, the second-order model proved to have the best performance, concluding that it also was the most robust model. The model had an average error of 1 meter from the true position at best, and 1.4 meters at worst.

Keywords		
Classification system and/or index terms (if any)		
Supplementary bibliographical information		
ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-69	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/