CHALMERS

Recursive Bayesian Estimation Applied to Autonomous Vehicles

Employing a stochastic algorithm on nonlinear dynamics for real-time localization

Master's thesis in Complex Adaptive Systems

ANNIE WESTERLUND
HELENA JAKOBSSON LARSSON

Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2015

MASTER'S THESIS IN COMPLEX ADAPTIVE SYSTEMS

Recursive Bayesian Estimation Applied to Autonomous Vehicles

Employing a stochastic algorithm on nonlinear dynamics for real-time localization

ANNIE WESTERLUND
HELENA JAKOBSSON LARSSON

Department of Applied Mechanics

Division of Vehicle Engineering & Autonomous Systems

CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2015

Recursive Bayesian Estimation Applied to Autonomous Vehicles
Employing a stochastic algorithm on nonlinear dynamics for real-time localization
ANNIE WESTERLUND
HELENA JAKOBSSON LARSSON

Master's thesis 2015:25
ISSN 1652-8557
Department of Applied Mechanics
Division of Vehicle Engineering & Autonomous Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
The Volvo FMX truck used in this project for testing the algorithm. Photographer: Annie Westerlund, 2015

Chalmers Reproservice
Göteborg, Sweden 2015

Recursive Bayesian Estimation Applied to Autonomous Vehicles
Employing a stochastic algorithm on nonlinear dynamics for real-time localization
Master's thesis in Complex Adaptive Systems
ANNIE WESTERLUND
HELENA JAKOBSSON LARSSON
Department of Applied Mechanics
Division of Vehicle Engineering & Autonomous Systems
Chalmers University of Technology

Abstract

This thesis presents an implementation of a sequential extended Kalman filter applied to position, velocity and attitude estimation of autonomous vehicles. The filter is self-tuning by the introduction of a particle swarm otimization which tunes the process noise covariance. The sensor fusion is adaptive through the means of corrector signals. It accepts correctors in position, velocity and attitude, in all possible combinations. The algorithm also includes an extended Kalman filter for quaternion update in order to make the estimations more robust when implementing sensor fusion flexibility. The filter architecture developed in this thesis is called the adaptive self-tuning extended Kalman filter, or the ASTEK filter. The algorithm was first tested in MATLAB/Simulink and then implemented and finalized in C++ in order to facilitate real-time performance. From testings on a truck, the RMS error for estimating position using a GPS corrector lies in the interval $[10^{-4}, 0.002]$ m, for velocity in $[10^{-4}, 0.02]$ m/s, and for the estimated attitude in $[10^{-3}, 0.21]$ degrees, depending on the road and driving mode. When using a 2D map corrector, that is correcting for x, y, and yaw, the RMS estimations of roll and pitch are higher and lying in the interval [1.1, 3.1]. However, it is kept stable as a result from the quaternion EKF, whereas the z-direction diverges as expected. The results show that the algorithm is able to produce estimations of high accuracy and that the corrector signals may vary dynamically. Moreover, the results show how different roads and driving modes influence the estimation and error evolution.

Keywords: autonomous vehicles, sequential extended Kalman filter, ASTEK filter, state estimator, localization, vehicle control, particle swarm optimization, stochastic optimization

Acknowledgements

We would like to extend our thanks to our supervisor Peter Forsberg, who guided us through this project, giving us constant ideas and feedback while still letting us decide some of the path ourselves. To CPAC Systems for letting us be a part of the company, and for helping us with hardware problems in the lab. Furthermore, to the other thesis students at CPAC for nice lunches, after works and other events.

To the chassis group at Volvo Group Trucks Technology for treating us like their own thesis workers, and of course for letting us take part of their Friday fikas. Also, for letting Helena outperform them at their prestigious running competition at Skatås, giving Annie some peace and quiet during the final weeks of the thesis. A special thanks is given to Carl-Johan Hoel at Volvo for being our informal supervisor, for giving us useful tips on the filter, and for bringing us to Hällered and help with recording the data. Moreover, a thanks to Leo Laine who lent his truck to us for these testings and giving us some of his time during the meetings at Volvo. Also, to Inge Johansson for arranging a Volvo computer which simplified the work enormously.

We would also like to thank Mattias Wahde for being our examiner. We are particularly grateful for the time and effort spent on reading and improving the report. Finally, we would like to thank all new readers. We hope you will enjoy the report as much as we did writing it!

Contents

Chapter 1

Introduction

In order to control and navigate different vehicles efficiently and safely, it is crucial to have a good estimate of all important internal states. [28] For a car, these states could include position, speed, attitude and slip. The internal states of interest are chosen depending on the vehicle, its intended use, and the environment with its specific conditions. In order to carry out proper navigation the most important states are position and orientation.

1.1 Background

There are many areas where it would be beneficial to have vehicles driving autonomously. It is therefore of great importance to be able to determine internal states of not only cars, but also other vehicles. This thesis has been carried out in collaboration with Volvo, a company dependent on having an accurate state estimator for several types of vehicles. Volvo produces trucks in various sizes and appearances, in addition to various construction equipment machines. Currently, investigations are carried out, both in academia and in industry, in order to decide under what circumstances autonomous vehicles should be used. It is clear that features like these are of great value in many contexts.

The states are typically estimated using a sensor fusion implemented in a statistical filter. Depending on the environment and setting, the sensor fusion will differ. [33] The most common sensor fusion is the combination of a GPS and an inertial measurement unit (IMU) which provides accelerometer and gyroscope data. When the GPS provides valid and complete data, the problem is rather straightforward. However, in areas where GPS signals do not reach the sensor, the solution will have to be more sophisticated and new sensor fusions have to be considered and developed. This problem will often not appear on open roads, but rather in, for example mines or very dense forests. As a remedy, one could match the estimated position and heading with map data obtained from feature extraction of the surrounding environment.

1.2 Purpose

The purpose of this master's thesis is to construct and develop simple Bayesian networks (statistical filters) for state estimation of autonomous vehicles. This thesis is part of a larger project with the goal of driving a truck autonomously in areas with frequent low GPS coverage. Therefore, the authors wishes to bring existing research closer to scenarios where certain sensors may fail to reach signals and further explore the possibility of using the estimator together with a two dimensional map or other sensor inputs.

1.3 Aim

The objective is to utilize state-of-the-art methods in order to develop a state estimator for autonomous vehicles. The final estimator is supposed to work on a general set of autonomous vehicles. Furthermore, the project should provide an implementation of a more flexible sensor structure, allowing for various sensor fusions. During the project, the new modules will be implemented and tested, both in a simulated environment and on a real truck.

1.4 Limitations

The estimator developed in this thesis estimates position, velocity, attitude, and gyroscope bias using statistical filtering techniques and stochastic optimization. The estimator does not concern tyre friction, side-slip or forces which can be estimated with tyre models. As mentioned in the aim, the filter is supposed to work on a general set of autonomous vehicles. The gain and bias of the accelerometer are assumed to be negligible after calibration. The same argument is valid for the gyroscope gain. The sensor noise is modelled as, or assumed to be, Gaussian white noise. Moreover, it is assumed in the model that all sensors are positioned on the chassis of the vehicle, ignoring the movement of for example the driver's cabin in a truck. Finally, the sensor fusion adaptivity will be restricted to only cover the corrector signals, not the input.

Chapter 2

Theory

This chapter presents the underlying theory which is required in order to understand the developed model and its applications. In Section 2.1.1, one can read about a common statistical method used for processing signals, namely the original Kalman filter and some corresponding usage areas. The sequential extended Kalman filter is presented in Section 2.1.3 as a way to use the Kalman filter on nonlinear systems. Moreover, information on particle swarm optimization can be seen in Section 2.2. Lastly, the possibility of using quaternions instead of Euler angles is considered in Section 2.3.2.

2.1 Statistical filtering techniques and localization methods

In the field of autonomous robotics, it is crucial to have a good estimate of the robot's location, in order for it to be able to navigate properly. [28] There are many methods which address this problem, ranging from rather simple to highly sophisticated. A common way to compute the position of a wheeled robot is for example to measure how much the wheels are spinning with wheel encoders and then estimate the corresponding distance using the radius of the wheel. This method is not very reliable since it does not take slip into account, and errors will propagate since the measure model always is incomplete. There are also methods using laser scans and maps in order to find a good estimate of the position. The method is precise due to the laser's precision, but is also dependent on the validity of the existing map. Consistent with these methods is the processing of different signals to create a valid estimate of the position. In this report, and in this particular review, we will consider the method of Kalman filtering and its extensions in order to provide a position that is accurate and trustworthy in environments where noise always is present.

2.1.1 Kalman filter

In 1960, Kalman [14] introduced a new method which first and foremost was used to predict random signals, to separate signals from random noise, and to detect signals in the presence of random noise. The Kalman filter is a linear filter based on statistical properties such as measurement noise and process noise. The algorithm can be seen as one of the most simple Bayesian networks, since the noise is assumed to be Gaussian. The method aims at estimating a vector with n certain states, $\boldsymbol{x} \in \mathbb{R}^n$ with a measurement $\boldsymbol{z} \in \mathbb{R}^m$, where the state is given by the difference equation,

$$\boldsymbol{x}_k = A\boldsymbol{x}_{k-1} + B\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1}. \tag{2.1}$$

The measurement is given by $\boldsymbol{z}_k = H\boldsymbol{x}_k + \boldsymbol{\nu}_k$. Here, A is the discrete system matrix, H is the measurement matrix, $\boldsymbol{y} = H\boldsymbol{x}$ the measurement model, and $\boldsymbol{w}_k$ and $\boldsymbol{\nu}_k$ are the process and measurement noise. The noise is assumed to be independent, white and Gaussian with zero-mean, and covariances Q and R respectively. The recursive algorithm consists of a prediction step followed by a correction step where more reliable data are used. Given a discrete, linear and stochastic dynamical system, the filter works as follows. [31] In the prediction step,

a problem dependent model is used to predict the a priori state estimates and the a priori covariance estimate, P_k. This is done by simply integrating the system discretely.

$$\boldsymbol{x}_k^- = A\boldsymbol{x}_{k-1} + B\boldsymbol{u}_{k-1} \tag{2.2a}$$
$$P_k^- = AP_{k-1}A^T + Q \tag{2.2b}$$

In the correction step, the Kalman gain, K_k, is derived and used to update the estimated state together with the discrepancy, $\boldsymbol{z}_k - H\boldsymbol{x}_k^-$.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.3a}$$
$$\boldsymbol{x}_k = \boldsymbol{x}_k^- + K_k(\boldsymbol{z}_k - H\boldsymbol{x}_k^-) \tag{2.3b}$$
$$P_k = (I - K_k H)P_k^- \tag{2.3c}$$

I is the identity matrix. The a priori and a posteriori state covariances can sometimes be very difficult to set properly which is a clear drawback of the particular method.

2.1.2 Extended Kalman filter

The original Kalman filter was developed for linear systems and is not applicable on nonlinear models. As a consequence, the extended Kalman filter (EKF) was introduced sequentially in the literature by several authors who wanted to adapt the original filter to nonlinear systems. [4] The EKF employs simple multi-variable calculus in order to handle nonlinearities using a first-order Taylor expansion. A nonlinear and stochastic dynamical system is given on the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \tag{2.4}$$

with states $\boldsymbol{x}$, inputs $\boldsymbol{u}$, and noise $\boldsymbol{w}$. The measurement model is also nonlinear and described by

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\nu}). \tag{2.5}$$

The prediction follows from a simple discretization scheme, for example Euler forward or Runge-Kutta. Utilizing the Euler forward would yield the following equation.

$$\boldsymbol{x}_k^- = \boldsymbol{x}_{k-1} + \Delta t \cdot \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{0}) \tag{2.6}$$

The next step, which is the the crucial part of the EKF, is deriving the discrete system matrix, A_k, and the discrete measurement matrix, H_k, through means of linearization. This is done by simply calculating the two respective Jacobians. A Jacobian is defined in multi-variable calculus as a matrix containing all partial derivatives of a vector-valued function, $\boldsymbol{F} \in \mathbb{R}^n$ with respect to the variables $\boldsymbol{x} \in \mathbb{R}^m$.

$$J(\boldsymbol{F}, \boldsymbol{x}) = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_m} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdots & \frac{\partial F_n}{\partial x_m} \end{pmatrix} \tag{2.7}$$

The discrete system and measurement matrices utilized in the extended Kalman filter are thus calculated as in Eqs. (2.8a)-(2.8b).

$$A_k = \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}_k^- \tag{2.8a}$$
$$H_k = \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{h}(\boldsymbol{x}_k^-, \boldsymbol{0}) \tag{2.8b}$$

The rest follows as in the standard Kalman filter that is, the Kalman gain is derived together with the discrepancy in order to find the corrected state vector, see Eqs. (2.3a)-(2.3c).

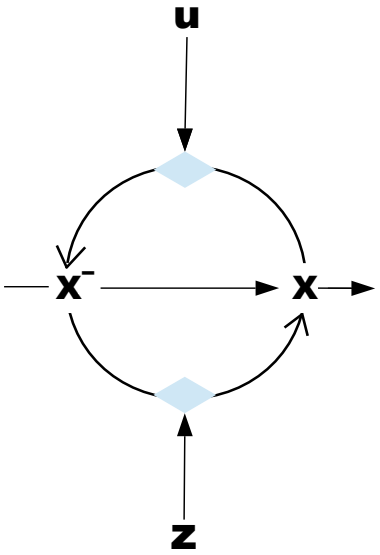

Figure 2.1: *A graphical demonstration of the SEKF. The prediction step is carried out each time an input, $\boldsymbol{u}$, is obtained, while the correction step is carried out only when the corrector, $\boldsymbol{z}$ is available. Thus, it is possible for the filter to output only the predicted states.*

2.1.3 Sequential extended Kalman filter

In connection to the EKF, a close extension was developed with the name sequential extended Kalman filter (SEKF). The foundation of this algorithm is basically the same, but with the addition of a sequential update when the input signal, $\boldsymbol{u}$, is obtained at higher rates than the correction signal, $\boldsymbol{z}$. The prediction occurs for every input signal whereas the correction merely occurs when a corrector signal is available. Figure 2.1 shows an SEKF cycle. Allowing signals to appear at different rates is a necessary feature when working with certain sensor fusions in navigation algorithms from probabilistic robotics.

2.2 Particle swarm optimization

Particle swarm optimization (PSO) is an algorithm that optimizes a given problem iteratively, without the use of gradients. PSO was introduced by Kennedy & Eberhart in [15] and is so called because the idea behind it was inspired by the collective motion of a swarm. In this algorithm, the swarm consists of particles which are described by both its position and velocity in the function landscape. A particle's change in velocity is both dependent on its own, and the others' performance with the aim to increase its value of the objective function. The algorithm is presented below. [29]

1. The algorithm starts by initializing positions and velocities randomly from a uniform distribution on the intervals given in Eqs. (2.9a)-(2.9b). The position of each particle $i = 1, \ldots, N$ is denoted $\boldsymbol{x}_i$ and the velocity is given by $\boldsymbol{v}_i$ where $\boldsymbol{x}_i, \boldsymbol{v}_i \in \mathbb{R}^n$ and n is the dimension of the variable that is to be optimized.

$$x_{ij} \in [x_{\min}, x_{\max}] \tag{2.9a}$$

$$v_{ij} \in \left[-\frac{\alpha(x_{\max} - x_{\min})}{2\Delta t}, \frac{\alpha(x_{\max} - x_{\min})}{2\Delta t} \right] \tag{2.9b}$$

2. For each particle, the objective function is evaluated, that is one computes $f(\boldsymbol{x}_i)$.

3. All particles' best positions, $\boldsymbol{x}_j^{pb}$, and the global best position, $\boldsymbol{x}^{sb}$, are updated.

4. The best position of each particle and the best position in the swarm are used in order to update the new positions and velocities.

$$v_{ij} = v_{ij} + c_1 q\left(\frac{x_{ij}^{pb} - x_{ij}}{\Delta t}\right) + c_2 r\left(\frac{x_j^{sb} - x_{ij}}{\Delta t}\right) \tag{2.10a}$$

$$|v_{ij}| \leq v_{\max} \tag{2.10b}$$

$$x_{ij} = x_{ij} + v_{ij}\Delta t \tag{2.10c}$$

Here, r and q denote two random numbers drawn from a uniform distribution on the interval $[0, 1]$, and c_1, c_2 are two parameters.

The algorithm proceeds with the next iteration, starting from step 2 until a convergence criterion is reached. The criterion is usually defined by a threshold using an acceptable performance, for example in a maximization problem where the objective function is non-positive everywhere, the threshold may be chosen a negative value close to zero.

2.3 Spatial rotations

When representing a rotation of a coordinate system or a vector within it one will find that there exist many different methods, for example Euler angles, quaternions, Rodrigues parameters, and Gibbs representation. For the purpose of this thesis, the following sections present how to use Euler angles and quaternions to describe orientation. This section also briefly considers dual quaternions as a tool for rotating and translating a vector.

2.3.1 Euler angles

The most basic method is using Euler angles, which is a description of how coordinate systems are rotated around their orthogonal axes. In the case of three dimensions, and orthogonal system XYZ, we have the angles that rotate around corresponding axes; ϕ, θ and ψ. The rotation around the x-axis is referred to as roll (ϕ), the rotation around y-axis as pitch (θ), and the rotation around the z-axis as yaw (ψ). The rotations are then described by the rotation matrices in Eqs. (2.11)-(2.13).

Rotation around the x-axis

$$R_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix} \tag{2.11}$$

Rotation around the y-axis

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{2.12}$$

Rotation around the z-axis

$$R_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.13}$$

A rotation of a vector is obtained through a simple matrix multiplication. Although easy to comprehend, this representation is ambiguous, singular at certain values, and computationally ineffective. Thus, there is need for other methods to compute rotations.

2.3.2 Quaternions

Quaternions can be used as an alternative to Euler angles in order to describe rotations in three dimensions. Unlike Euler angles, quaternions are more computationally effective and additionally, they are not ambiguous nor singular. Quaternions were discovered and explained by Hamilton [7]. The quaternion is an extension of complex numbers and can be seen as an element in $\mathbb{R}^4$. There are two common ways of defining a quaternion. In this report the following notation will be used, where q_0 denotes the scalar part and q_1, q_2, q_3 denotes the vector part of the quaternion.

$$\boldsymbol{q} = q_0 + q_1 i + q_2 j + q_3 k. \tag{2.14}$$

The basis i, j, k is similar to the regular complex base i, with some additional algebraic rules as will be presented below. From this description, the quaternion can be considered as a quadruple of real values, $\hat{\boldsymbol{q}} = (q_0,\, q_1,\, q_2,\, q_3) \in \mathbb{H} = \mathbb{R}^4$, but since the quaternion is constructed from a scalar part and a complex vector part, one can also view this construction as $\hat{\boldsymbol{q}} = q_r + \boldsymbol{q_c}$. [1]

As mentioned, the quaternions extend the complex numbers wherein only one imaginary unit exists with the property $i^2 = -1$. In quaternion space, there are three imaginary units, i, j, and k with the following properties.

$$i^2 = j^2 = k^2 = -1 \quad \text{and} \quad jk = -kj = i, \quad ki = -ik = j, \quad ij = -ji = k \tag{2.15}$$

Operations and definitions

Quaternion algebra differs in some cases from the algebra valid from complex numbers. The vector part $\boldsymbol{q_c}$ can however be seen as a vector where all common operations are valid such as dot product, cross product, scaling, and addition. The operations in quaternion algebra that will be used in the algorithms presented in this report are listed below.

Multiplication

$$\begin{aligned}\hat{\boldsymbol{q}} \otimes \hat{\boldsymbol{r}} =& (q_0 + q_1 i + q_2 j + q_3 k) \otimes (r_0 + r_1 i + r_2 j + r_3 k) = \\ & i(q_2 r_3 - q_3 r_2 + r_0 q_1 + q_0 r_2) + \\ & j(q_3 r_1 - q_1 r_3 + r_0 q_2 + q_3 r_2) + \\ & k(q_1 r_2 - q_2 r_1 + r_0 q_3 + q_0 r_3) + \\ & q_0 r_0 - q_1 r_1 - q_2 r_2 - q_3 r_3 = \\ & (\boldsymbol{q_c} \times \boldsymbol{r_c} + r_0 \boldsymbol{q_r} + q_0 \boldsymbol{r_r}, q_0 r_0 - \boldsymbol{q_v} \cdot \boldsymbol{r_r})\end{aligned} \tag{2.16}$$

Addition

$$\hat{\boldsymbol{q}} \oplus \hat{\boldsymbol{r}} = (q_r, \boldsymbol{q_c}) \oplus (r_r, \boldsymbol{r_c}) = (q_r + r_r, \boldsymbol{q_c} + \boldsymbol{r_c}) \tag{2.17}$$

Conjugate

$$\hat{\boldsymbol{q}}^* = (q_r, \boldsymbol{q_c})^* = (q_r, -\boldsymbol{q_c}) \tag{2.18}$$

Norm

$$\|\hat{\boldsymbol{q}}\| = \sqrt{\boldsymbol{q} \cdot \boldsymbol{q}^*} = \sqrt{\boldsymbol{q_c} \cdot \boldsymbol{q_c} + q_0^2} = \sqrt{q_0^2 + q_2^2 + q_3^2 + q_4^2} \tag{2.19}$$

Identity

$$\hat{\boldsymbol{q}}_{\boldsymbol{i}} = (1, \boldsymbol{0}) \tag{2.20}$$

Time derivative

$$\frac{d}{\mathrm{d}t}\hat{\boldsymbol{q}} = \frac{1}{2}\hat{\boldsymbol{q}} \otimes [0,\ \dot{\phi},\ \dot{\theta},\ \dot{\psi}] \tag{2.21}$$

Here, $[0,\ \dot{\phi},\ \dot{\theta},\ \dot{\psi}]$ denotes a quaternion with zero scalar part and a vector part defined by Euler angular velocities.

Unit quaternions and rotations

Quaternions span $\mathbb{R}^4$ since there is no restriction on the quadruple $(q_0, q_1, q_2, q_3) \in \mathbb{H}$. For the subject of this thesis, there is however an important subset of $\mathbb{H}$ that represents unit quaternions. This set corresponds to a unit hyper sphere in $\mathbb{R}^4$, that is, a four-dimensional sphere with radius one. The norm of a unit quaternion is always one, which is a restriction from the regular quaternions. From now on, the quaternion $\hat{\boldsymbol{q}}$ is assumed to be of unit length. From the property, $\|\hat{\boldsymbol{q}}\| = 1$, the quaternion can be rewritten in terms of an arbitrary angle, ϕ as

$$\hat{\boldsymbol{q}} = (\cos\phi, \sin\phi \boldsymbol{u_q}) = \cos\phi + \sin\phi \boldsymbol{u_q}, \tag{2.22}$$

since the square of the quaternion components $\hat{\boldsymbol{q}}$ adds up as $\|\hat{\boldsymbol{q}}\|^2 = \cos^2\phi + \sin\phi$ for some three-dimensional vector $\boldsymbol{u_q}$, with $\|\boldsymbol{u_q}\| = 1$.

Unit quaternions can represent three-dimensional rotations in a compact notation and in addition, the computations are unambiguous. A general vector in three dimensions can easily be rotated by a unit quaternion. For a point $\boldsymbol{p}$ with components (p_x, p_y, p_z), a quaternion is declared by the same components in the complex vector part and a zero real part. This point, or vector, is to be rotated with a unit quaternion $\hat{\boldsymbol{q}} = (\cos\phi, \sin\phi \boldsymbol{u_q})$. In order to perform a rotation, the so called sandwiching map is applied to the quaternion $\hat{\boldsymbol{q}}$ as

$$\hat{\boldsymbol{p}}' = \hat{\boldsymbol{q}} \otimes \hat{\boldsymbol{p}} \otimes \hat{\boldsymbol{q}}^* = \hat{\boldsymbol{q}} \otimes \hat{\boldsymbol{p}} \otimes \hat{\boldsymbol{q}}^{-1}. \tag{2.23}$$

This mapping will rotate the point $\boldsymbol{p} = (p_x, p_y, p_z)$ around the arbitrarily axis $\boldsymbol{u_q}$ an angle of 2ϕ radians. The conjugate of a quaternion will represent a orientation in the other direction, meaning that if $\hat{\boldsymbol{q}}$ correspond to the orientation relative frame A to frame B, the conjugate $\hat{\boldsymbol{q}}^*$ will correspond to the relation of frame B to frame A.

2.3.3 Dual quaternions

A quaternion is a compact representation of rotations, but it is not sufficient for translations. There is however an extension to quaternions, called dual quaternions that represents both translation and rotation without going back to Euler angles as an intermediate step, which would be the case if regular quaternions were used. In this section, dual quaternions will be thoroughly presented in such scope the method requires. The dual quaternions will be used for representing transformation of rigid bodies, which in this case corresponds to the different frames of a vehicle.

Dual numbers and the definition of dual quaternion

A dual quaternion is constructed from two regular quaternions, where one is multiplied with the dual number ε. [23] The dual quaternion is hence an eight-dimensional vector defined as

$$\hat{\boldsymbol{Q}} = \hat{\boldsymbol{q}} + \varepsilon\hat{\boldsymbol{q}}_{\boldsymbol{\varepsilon}} = q_0 + q_1 i + q_2 j + q_3 k + \varepsilon(\tilde{q_0} + \tilde{q_1} i + \tilde{q_2} j + \tilde{q_3} k) = (q_0,\, q_1,\, q_2,\, q_3,\, \tilde{q_0},\, \tilde{q_1},\, \tilde{q_2},\, \tilde{q_3}). \tag{2.24}$$

The construction of two quaternions combined with a dual number yields that the algebra of dual numbers should be considered. Briefly, the dual number algebra again is very close to what is valid for complex numbers. A dual number can be written as $d = a + \varepsilon b$, where a and b are real numbers and $\varepsilon^2 = 0, \varepsilon \neq 0$. The conjugate is defined, similar to complex numbers, as $d^* = a - \varepsilon b$. The notation $d = (a, b)$ will be used from now on to represent dual numbers.

A dual quaternion is hence a dual number where the components of it are regular quaternions as $\hat{\boldsymbol{Q}} = (Q, \tilde{Q})$, with $Q = (q_0 + q_1 i + q_2 j + q_3 k)$, and $\tilde{Q} = (\tilde{q_0} + \tilde{q_1} i + \tilde{q_2} j + \tilde{q_3} k)$. The extension to an 8-tuple is valid and presented

as $\hat{\boldsymbol{q}} = (q_0, q_1, q_2, q_3, \tilde{q_0}, \tilde{q_1}, \tilde{q_2}, \tilde{q_3})$. The method of using dual quaternions requires some specific algebra rules. Firstly, the multiplication of two dual quaternions is given as

$$\hat{\boldsymbol{P}}\hat{\boldsymbol{Q}} = \left(P,\, \hat{P}\right)\left(Q,\, \hat{Q}\right) = \left(PQ,\, \tilde{P}Q + P\tilde{Q}\right), \tag{2.25}$$

and the square root for a dual number is defined as

$$\sqrt{\boldsymbol{d}} = \sqrt{(a,\, b)} = \left(\sqrt{a},\, \frac{b}{2\sqrt{a}}\right). \tag{2.26}$$

These operations will be used when computing the norm of a dual quaternion and just as in the case of regular quaternions, the norm should be unity. The quaternion conjugate is used to compute the norm. There are several ways to define the conjugate of a dual quaternion, all with its own drawbacks and usage areas. In the field of rigid body transformations, the following notation will be used

$$\hat{\boldsymbol{Q}}^{\boldsymbol{*}} = \left(Q^*,\, \tilde{Q}^*\right) = (q_0,\, -q_1,\, -q_2,\, -q_3,\, \tilde{q}_0,\, -\tilde{q}_1,\, -\tilde{q}_2,\, -\tilde{q}_3)\,. \tag{2.27}$$

This definition satisfies the properties $(\hat{P}\hat{Q})^* = \hat{Q}^*\hat{P}^*$ and $(\hat{Q}^*)^*$ as expected from the rules of complex numbers. From the conjugate, we can define the norm of a dual quaternion

$$\|\hat{\boldsymbol{Q}}\| = \sqrt{\hat{\boldsymbol{Q}}\hat{\boldsymbol{Q}}^{\boldsymbol{*}}} = \sqrt{\left(|\hat{Q}|^2, \tilde{Q}Q^* - (\tilde{Q}Q^*)^*\right)} = \left(|\hat{Q}|, \frac{\tilde{Q}Q^* - (\tilde{Q}Q^*)^*}{|\hat{Q}|}\right). \tag{2.28}$$

From this definition, one can conclude that for a dual quaternion to have unit norm, the real part Q must be unity and the quaternion $\tilde{\boldsymbol{Q}}\hat{\boldsymbol{Q}}^*$ must be real which makes the dual part zero. The unit dual quaternion is used for representing rotations and translation as will be described in the next section.

2.3.4 Translation and rotation using dual quaternions

A unit quaternion describes a rotation in three dimensions and can be rewritten as a dual quaternion, also with unit norm, as $\hat{\boldsymbol{q}} = (q, 0)$. The point p as a dual quaternion is also easily written as $\hat{\boldsymbol{P}} = (1, 0, 0, 0, 0, p_x, p_y, p_z)$. The rotation is taken care of with the same procedure as with a regular quaternion. The spatial translation is constructed from a three dimensional translation vector $t = (t_x, t_y, t_z)$. This vector is transformed to the quaternion $(0, t/2)$, which as a dual quaternion is given as $\hat{\boldsymbol{q}} = \left(1, 0, 0, 0, 0, \frac{t_x}{2}, \frac{t_y}{2}, \frac{t_z}{2}\right)$. In order to perform the translation on the point p, the sandwiching product will again be utilized as

$$\hat{\boldsymbol{P}}' = \hat{\boldsymbol{T}}\hat{\boldsymbol{P}}\hat{\boldsymbol{T}}^{\boldsymbol{*}} = (1, P + 2T). \tag{2.29}$$

The definition of conjugate yields $\hat{\boldsymbol{T}}^{\boldsymbol{*}} = (1, -T^*) = (1, T) = \hat{\boldsymbol{T}}$. The transformed point p' is now retrieved from the dual part $\hat{\boldsymbol{P}}'$, and $p' = (p_x + t_x, p_y + t_y, p_z + t_z)$ as expected.

2.4 Related work and applications

In order to address the localization and navigation problem, many methods have been explored in the literature using variations of Kalman filters. This section will provide a brief overview of related work and applications to the theory presented in section 2.1-2.3, thus where Kalman filtering techniques have been applied to navigation. Most of the applications in this overview consider state estimation of cars and robots. However, they are easily related to trucks since the algorithm is, or could be made, general. The joint method of all authors presented here is some variation of Kalman filters. However, the procedure varies depending on the application. This is

clear when studying the choice of state vectors. The states one estimates differ depending on the application. Therefore, the literature presents a variety of different state vectors, for example only attitude [16, 19], position and heading in 2D [10] or 3D [8, 4, 17, 32], and other states such as acceleration [32], or gyroscope-accelerometer bias and gain [4]. Position and velocity are usually predicted using an accelerometer while attitude usually is predicted by the use of a gyroscope. More on sensors and sensor fusions are discussed below. When bias and gain are present in the state vector, they are modelled as white noise in the prediction step. There is no corrector signal for bias and gain, meaning that they are silent states that will be implicitly updated.

The literature also introduces a few different possible sensor fusions. The most common approach to estimate the decided states is to use an EKF together with a GPS and an IMU. This has been done in [6, 4, 13, 3].The IMU is a common sensor in these applications and the device measures acceleration and rotational changes, which is the foundation for estimating the movement of the vehicle in these algorithms. In [4], the EKF was also made sequential which allows the sensor data to come at different rates. An IMU often incorporates an accelerometer and a gyroscope, but the addition of a magnetometer could also be done. Some work have included this variety of sensor fusion by utilizing a magnetometer together with an IMU or a GPS, depending on the application. Such methods are found in [32, 19, 16]. In [19], both an EKF and a newly developed complementary filter were used and computationally compared.

Moreover, other methods involving EKF as estimator have been used. An example is found in [17] where dead-reckoning was used together with an IMU to estimate the pose of a robot in terrain. The dead-reckoning assumed a no-slip condition, something that is not valid in real terrain. Thus, the results were corrected using IMU data. Another method which have been considered is to simplify a four-wheeled vehicle to a two-wheeled kinematic model with steering angle, wheel angular velocities which were then processed in an EKF [10]. Furthermore, the concept of using a regular Kalman filter in order to reduce computational time have also been explored in [8]. In this paper, the data obtained from the sensors were preprocessed and linearized before entering the filter, which created a computationally efficient method.

When estimating the position of an object with an elongation, an important state to estimate is the attitude. There are several valid mathematical definitions of spatial rotations and different articles have employed different ways of representing them. Often the works have involved Euler angles [4, 6, 10] or quaternions [5, 22, 16, 19]. The inclusion of quaternions has been done in several papers and in various fields due to their property of unambiguity, which is further explained in Section 2.3.2. Quaternions are especially useful when the obstacle in consideration is allowed to make full rotations in all directions. A spacecraft is a perfect example of such a vehicle and was studied extensively in [5], where attitude in quaternion form was included in the state vector. A similar application could also be found in [22], where the algorithm was implemented for a small-sized helicopter.

The idea of using a particle swarm optimization in an EKF to gain more accurate results is slightly touched upon in the literature [13, 3]. One way to utilize this combination is to have the PSO algorithm tuning the process noise covariance, given that some accuracy, defined by statistical samples, falls below a certain threshold [13]. The objective function in this paper was chosen using the trace of system and process matrices. Another way to use this optimization algorithm is to try to find the optimal estimated point [3]. In [3] this was achieved given the state covariance, measurements, predictions, and measurement noise covariance.

Extensive research has been done in the field of vehicle state estimation, some of which has been presented above. In this project, some of the ideas and methods will be applied in order to adjust and develop an algorithm suitable for the purpose of the thesis. The sensor fusion is inspired by [6, 4, 13, 3], where the authors use an IMU and a GPS as their primary data source. The idea of using sensor fusion at different rates in an SEKF [4] will be used, since the rates of the provided devices differ when obtaining data with high accuracy. By representing rotations with quaternions as in for example [5, 19], the filter will be more general and applicable on vehicles that move without angular restrictions. Since the vehicle in consideration for this thesis is expected to move in different areas, where GPS signals not always are reachable, it is also desirable for the sensor fusion to be flexible.

Chapter 3

The adaptive self-tuning extended Kalman filter

3.1 The basic model

The first model considered here is a sequential extended Kalman filter which assumes that the prediction step is carried out using data from an IMU, and the correction using a three-antenna GPS. Denoting the vehicle position by $\boldsymbol{p} = [p_x, p_y, p_z]$, the velocity by $\boldsymbol{v} = [v_x, v_y, v_z]$, the orientation is described by the quaternion $\hat{\boldsymbol{q}} = q_0 + q_1 i + q_2 j + q_3 k$, and $\boldsymbol{b}_g = [b_x, b_y, b_z]$ is the gyroscope bias. Then, the state vector can be written

$$\boldsymbol{x} = [\boldsymbol{p}, \quad \boldsymbol{v}, \quad \hat{\boldsymbol{q}}, \quad \boldsymbol{b}_g]. \tag{3.1}$$

Before the prediction step, the IMU data are rotated into earth frame using the rotations in Eqs. (3.2a)-(3.2b), where $\bar{\boldsymbol{a}}$ is the accelerometer data and $\bar{\boldsymbol{\omega}}$ is the gyroscope data. Furthermore, the angular velocities obtained from the gyroscope are converted to quaternion time derivatives with Eq. (2.21).

$$\boldsymbol{a} = \hat{\boldsymbol{q}} \otimes \bar{\boldsymbol{a}} \otimes \hat{\boldsymbol{q}}^* + [0,\, 0,\, 1]g \tag{3.2a}$$

$$\boldsymbol{\omega} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{pmatrix} \cdot \bar{\boldsymbol{\omega}} - \boldsymbol{b}_g \tag{3.2b}$$

The prediction is carried out with a simple Euler forward discretization scheme, using the quaternion time derivatives and the acceleration. The time derivative of the gyroscope bias is usually modeled as white noise. The best approximation when simulating is to assume that the bias is constant, since the spectrum of the process is flat.

$$\boldsymbol{v}_t = \boldsymbol{v}_{t-1} + \Delta t \cdot \boldsymbol{a}_t \tag{3.3a}$$

$$\boldsymbol{p}_t = \boldsymbol{p}_{t-1} + \Delta t \cdot \boldsymbol{v}_t \tag{3.3b}$$

$$\hat{\boldsymbol{q}}_t = \hat{\boldsymbol{q}}_{t-1} + \Delta t \cdot \dot{\hat{\boldsymbol{q}}}_t. \tag{3.3c}$$

Since a three-antenna GPS is assumed to be the corrector, the measurement model is taken as the estimated position, velocity and attitude in earth frame. The position, velocity and attitude can be transformed to the right frame using dual quaternions. In this case, the dual quaternion, $\hat{\boldsymbol{Q}}$ equals $(1,\, 0,\, 0,\, 0) + \varepsilon(0,\, 0,\, 0,\, 0)$. The process system and measurement matrices are derived as explained in Eqs. (2.8a)-(2.8b), and the correction is carried out as in Eqs. (2.3a)-(2.3c).

3.2 Extensions to the basic model

The basic model is formulated for a problem where the process noise covariance Q is known or chosen and the corrector signals include complete position, velocity and attitude. This requires a three-antenna GPS which may not be used as a corrector for all purposes. As mentioned in Chapter 1, a GPS cannot give valid signals when obscured. In this chapter, a suggestion on how to use PSO in order to estimate Q is given. Different solutions to the sensor fusion problem is also presented. These extensions together with the basic model form the adaptive self-tuning extended Kalman filter (ASTEK filter).

3.2.1 Tuning the process noise covariance

As mentioned before, a large problem concerning Kalman filters is the parameter tuning, and especially tuning the process noise covariance matrix, Q. In an attempt to make the filter more general and robust, a minimization problem is formulated in order to find the diagonal elements of the covariance matrix, $Q_1, \ldots, Q_n$. The minimization problem is solved with the help of a particle swarm optimization algorithm, which is described in more detail in Section 2.2. An objective function was formulated that depends on Q and is based on samples obtained from the N last correction steps of the EKF. The objective is to minimize the sum of square residuals of the correction signal and the corrected signal. This function is described mathematically as

$$f(Q) = \sum_{j=1}^{N} \sum_{i=1}^{m} \left(z_i^{(j)} - h\left((A_k P_{k-1} A_k^T + Q) \cdot H_k \cdot (H_k^T (A_k P_{k-1} A_k^T + Q) H_k + R)^{-1} \cdot (\boldsymbol{z}_k - \boldsymbol{y}_k), \boldsymbol{\nu} \right)_i^{(j)} \right)^2 \tag{3.4}$$

The notation is the same as in Section 2.1.1 and $\boldsymbol{z} \in \mathbb{R}^m$. The final optimization problem can thus be formulated with constraints. It is given by Eqs. (3.5a)-(3.5c).

$$\begin{aligned} \text{minimize} \quad & f(Q) && \text{(3.5a)} \\ \text{subject to} \quad & Q_1, \ldots, Q_n \geq 0 && \text{(3.5b)} \\ & Q_1, \ldots, Q_n \leq 1 && \text{(3.5c)} \end{aligned}$$

Since the algorithm is quite computationally heavy, the PSO is not run in every correction step. Instead, the PSO can for example run every 10^{th} second or constantly in a parallel system, returning new covariance matrices to the filter.

3.2.2 Flexible sensor structure

Different actions are taken depending on the sensor inputs to the filter. When running the algorithm, the input sensors are allowed to vary and the action taken in order to correct is determined dynamically in real-time. This section describes how the filter operates given different inputs.

Gyroscope-accelerometer EKF

The main filter includes an additional EKF, where the gyroscope and accelerometer data are used as input and corrector signal, respectively. This makes it possible to always be able to correct the quaternion, even if no more exact correction signal is given. Furthermore, the filter has merely the quaternion in its state vector.

The state prediction is carried out as in Eq. (3.3c). In order to construct a measurement model, it is assumed that the accelerometer is at rest, with no linear acceleration. This means that it gives $[0, 0, -g]$ as data in the earth coordinate system. By rotating this with the conjugate of the estimated quaternion, one should obtain the accelerometer data which is given as a corrector signal to the filter. There is a need to account for linear acceleration in this filter since it will occur in some driving modes, for example during a severe brake, and

thus will cause the measurement model to become inaccurate. A remedy for this is to dynamically update the measurement noise covariance matrix, R, depending on the amount of linear acceleration, l. The measurement covariance for the accelerometer should be increased in order to reduce the importance and reliability of the corrector signal. The amount of linear acceleration is determined by the magnitude of the accelerometer data, so that a larger deviation from g means more linear acceleration and larger values in the covariance matrix. The matrix is updated exponentially by the deviation.

$$l = \|\boldsymbol{a}\| \tag{3.6a}$$

$$R' = R \cdot \mathrm{e}^{\mathrm{k}|\mathrm{g}-\mathrm{l}|} \tag{3.6b}$$

The parameter k is chosen so that the predicted states are left unchanged after correction for cases when the linear acceleration is high enough.

Dynamic sensor fusion

The basic SEKF is developed for a three-antenna GPS as corrector. There are, however, other possible correctors which provide different amount of corrector signals. These need to be accounted for. Examples of such correctors are two-antenna GPS and 2D map.

The difference between a two-antenna GPS and a three-antenna GPS is that one can obtain all three Euler angles from a three-antenna GPS, while only two Euler angles can be obtained from a two-antenna GPS. Depending on how the GPS is mounted on the vehicle, the two-antenna GPS can either provide yaw and roll (ψ, ϕ) or yaw and pitch (ψ, θ). For the 2D map case, the correction signals are obtained from an already built map. The data available are position in xy-plane, velocity in the xy-plane, and the yaw angle (ψ).

In this project, the filter is developed so that not only the three above mentioned cases are allowed as correction signals. Instead any combination of correction signals that are available will be used to update the prediction. This is done with the help of a Boolean vector which marks the signals that are available as 1 and the signals that are not available as 0. The Boolean vector is defined as

$$\boldsymbol{s}_{\text{correction}} = [\boldsymbol{p}, \quad \boldsymbol{v}, \quad \boldsymbol{\phi}], \tag{3.7}$$

where $\boldsymbol{\phi} = [\phi, \theta, \psi]$ denotes roll, pitch, and yaw. The Euler angles are converted to a quaternion. If not all three angles are given, the additional EKF will be used to correct the rest. Using this notation the vector $\boldsymbol{s}_{\text{correction}} = [1, 1, 1, 0, 0, 0, 0, 0, 0]$ corresponds to correction of position in x-, y-, and z-direction. In the same fashion, map matching is recognized as $\boldsymbol{s}_{\text{correction}} = [1, 1, 0, 1, 1, 0, 0, 0, 1]$. The rows in the covariance measurement matrix (H_k) that corresponds to the available signals are selected, together with their measurement noise covariances. The algorithm will then operate as usual when updating the state vector. This automatically makes it possible to correct the states for 2^9 different correction cases, that is, the filter becomes more general and easily allows for a dynamic change of correction device or loss of signals.

Propagating variances

Since the corrector signals can be obtained from different devices, the signal variances in the measurement noise covariance matrix may vary. Updating the quaternion could for example sometimes be carried out using the additional EKF or the main SEKF. In order to weight the possible updates, the algorithm takes the different variances into account and the filter will be updated according to the variances. Since the accuracy of the corrector signals from different devices can vary, both between the devices but also from time to time, it is possible to change the measurement noise covariance dynamically. The variances will propagate through the filter and the correction will be carried out depending on it. This allows for more flexibility in the system.

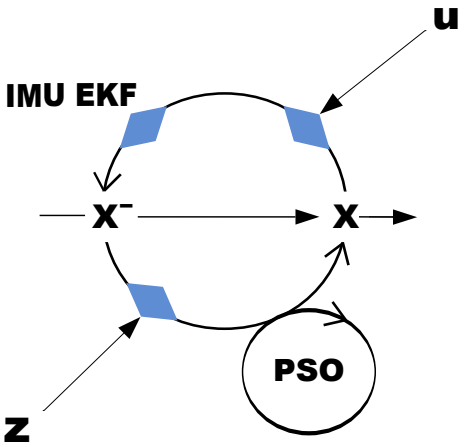

Figure 3.1: *A visualization of the ASTEK filter. The input, u, to the filter corresponds to IMU data. The gyroscope-accelerometer (IMU) EKF is carried out each time an input is provided. When a corrector signal is present, in any form through the flexible sensor structure, the correction step in the SEKF is performed. The PSO tunes the process noise covariance in a user defined time interval, as for the result presented here, every* 10^{th} *second.*

3.3 Overview of the ASTEK filter

The complete filter architecture is shown in Figure 3.1. As described before, the input, $\boldsymbol{u}$, to the filter corresponds to gyroscope and accelerometer data from the IMU. The prediction step and the quaternion correction from the gyroscope-accelerometer EKF, in the overview denoted as IMU EKF, are carried out when input is provided to the filter. When a corrector signal, $\boldsymbol{z}$, is present, the correction step in the SEKF is performed. If no corrector is available, the estimate is taken as the predicted and quaternion corrected state vector. The main corrector signal may vary dynamically as explained in Section 3.2.2. Depending on the environment and settings, the PSO is performed in order to tune the process noise covariance. For the implementation presented in this report, the PSO was executed every 10^{th} second.

Chapter 4

Hardware and signal processing

4.1 Inertial measurement unit

In this project, an IMU provided the filter with input in form of raw accelerometer and gyroscope signals. When the motor of the vehicle is on, the signal noise will be strongly amplified, which affects the process noise. The IMU is widely used in this way due to its high rate, 250 Hz. The high rate of the IMU makes it possible to predict the current position, velocity, and attitude in between the corrector signals which are provided by a more accurate sensor, i.e. the GPS.

4.1.1 Logging gyroscope and accelerometer data

During logging of data with the Volvo FMX truck, the IMU device was connected to a dSPACE MicroAutoBox, a device where several different processors and I/O boards can be mounted and process signals from various sensors. The IMU was connected to the Autobox with CAN, hence the signals are sent via CAN-buses. The raw signals obtained from the Autobox were saved as MAT-files which were post-processed in MATLAB. Since it was difficult to mount the IMU so that its coordinate frame would be parallel or orthogonal to the truck body frame, the IMU was calibrated by logging when the vehicle was standing still, pointing in different directions. From the data logs, a bias in each direction was computed and subtracted from the raw data. This was done in the body frame for the IMU, the rotation of the data is described in the following section.

4.1.2 Translation of data between frames

The IMU device has an internal coordinate system which must be transformed to the body frame of the vehicle. The vehicle's body frame is a standard NED-frame (north-east-down), and depending on how the IMU device is mounted, the data have to be rotated. The mounting on the FMX truck was made behind the cab on the chassis in order to keep the noise level as low as possible. The attachment corresponded to a rotation of the device in ZYX-order Euler angles as: $\phi = \pi$, $\theta = 0$, $\psi = \pi/2$. This rotation will be represented as a quaternion and the accelerometer data are rotated by the quaternion product as described in Section 3.1. The gyroscope data are rotated with a rotation matrix constructed from the Euler angles, similar to what was done in the implemented Kalman filter.

4.2 GPS

A global positioning system (GPS) can produce quite accurate positioning, velocity, and attitude signals. This is done with a satellite navigation system. The GPS used in this project was quite accurate given that a base

Name	Type	Length [bytes]
Number of satellites	Byte	1
GPS week	Unsigned short	2
GPS time of week	Double	8
Latitude	Double	8
Longitude	Double	8
Height	Float	4
Velocity north	Float	4
Velocity east	Float	4
Velocity up	Float	4

Table 4.1: *A table presenting the most useful binary message components logged from the GPS receiver. These were transferred by serial RS232 and translated via a C-script in order to interpret the messages and converting them. The GPS data are used as corrector in the ASTEK filter, denoted as z in Figure 3.1.*

station is used together with it. When a base station is not used, the accuracy of the GPS becomes a lot worse, especially when measuring height. As mentioned in Section 3.2.2, the number of Euler angles that can be measured, depends on the number of antennas used in the GPS. In this project, the GPS receiver contains two antennas and it was used together with a base station during data logging.

4.2.1 Logging GPS messages

The GPS receiver is communicating via an RS232 serial port. The receiver sends the messages requested by the user, each containing a factory defined set of data. For this application, when position, velocity and attitude data are needed, there are several options. However, in order to log as few messages as possible, a binary message containing most of the necessary components is logged. For the ones that are missing, a complementary ASCII-message is logged, which includes the attitude data.

During the logging, the GPS receiver was connected to the Autobox, which is convenient since this software makes sure that the timestamps of the signals coincide. The rate at which the signals are logged is for the GPS 10 Hz. Sometimes the signals is lost, which will yield a zero-logging. This is however taken care of during the post-processing of the data, to prevent the filter to follow signals that are erroneous.

4.2.2 Interpretation of binary messages

The binary messages contain numerous components, some of which are important and some not. The length of the binary message being read is 52 bytes excluding the header of 8 bytes and the epilogue of 4 bytes. The message of 52 bytes includes components as described in Table 4.1, where all the positional data as well as time data is of interest.

The data is logged as a string, where the data are represented as in hex, which then is cut into corresponding sub messages. These components contain floats, doubles, unsigned chars and unsigned shorts in binary strings which are to be decoded. This is done by a C-script which is converted to a MATLAB executable (MEX) function.

Chapter 5

Testing the implementation

The model has been tested in different environments and settings during the project in order to verify the performance in various scenarios. To reduce the complexity of the system, the model was first tested on data generated from a simple accelerometer model. The filter was then tested on data obtained from a virtual truck model (VTM), which was developed and provided by Volvo. When the filter performed well enough in simulations, additional data were logged from a real truck. This chapter thoroughly describes the generation of data and how the testing was performed.

5.1 Simulated environment

The testing in simulated environment has been carried out by increasing the complexity of the underlying model and is presented in the section below. During evaluation of the models, the order was slightly reversed as the virtual truck model (VTM) data set was the first to be used.

5.1.1 Generating simulated IMU data

In order to have full control of the input values to the filter, a simple IMU emulator was developed with the purpose of generating accelerometer and gyroscope data given a certain road. The advantage of this simple model is that the system complexity is scaled down. This will make it easier to detect possible bugs and logical errors in the different implementations. The emulator takes a three dimensional vector p with x, y and z coordinates as inputs. These points are logged with a time difference δt. From this data set, one can estimate the velocity and acceleration of a point that moves in a trajectory interpolated from the coordinate set. The velocity is generated from a simple discrete differentiation scheme of the positions in p as

$$\boldsymbol{v}_{t-1} = \frac{\boldsymbol{p}_t - \boldsymbol{p}_{t-1}}{\delta t}. \tag{5.1}$$

In order to obtain acceleration, the velocity is differentiated with respect to time. The gravity component is then subtracted from the measured acceleration in z-direction to get the proper value of an IMU.

$$\boldsymbol{a}_{t-1} = \frac{\boldsymbol{v}_t - \boldsymbol{v}_{t-1}}{\delta t} - [0,\, 0,\, g]. \tag{5.2}$$

These values are correct with respect to earth frame, i.e. a coordinate system pointing towards north, east and down (NED-frame). However, the IMU is mounted so that the coordinate system of the device coincide with the body frame of the vehicle. This rotation is given by the attitude of the vehicle, which must be estimated as

well. The assumption that the nose of the vehicle always is moving forward is done for convenience. By using that $dx = v_t^x - v_{t-1}^x$, $dy = v_t^y - v_{t-1}^y$, and $dz = v_t^z - v_{t-1}^z$, the attitude of the vehicle is defined as

$$\phi_{t-1} = \tan^{-1} \frac{dz}{\sqrt{dx^2 + dy^2}}, \tag{5.3a}$$

$$\theta_{t-1} = \tan^{-1} \frac{dy}{\sqrt{dx^2 + dz^2}}, \tag{5.3b}$$

$$\psi_{t-1} = \tan^{-1} \frac{dy}{dx}. \tag{5.3c}$$

The definition of attitude is dependent on which sequence the object is rotated in when using Euler angles. This definition uses the sequence ZYX, which also is the case, without exceptions, in the further implementation when Euler angles is used. To get the rotated measure of acceleration, corresponding to the one given from the IMU, the acceleration in earth frame is rotated by the quaternion conjugate of the vehicle attitude as

$$\boldsymbol{a}_{\text{IMU}} = \hat{\boldsymbol{q}}^* \otimes \boldsymbol{a} \otimes \hat{\boldsymbol{q}}. \tag{5.4}$$

The gyroscope will measure the angular rotation of the device, and will be modelled as the rate of change of the three rotation angles, analogously with spatial velocity and acceleration.

$$\boldsymbol{\omega}_t = \frac{\boldsymbol{\phi}_t - \boldsymbol{\phi}_{t-1}}{dt}, \tag{5.5}$$

analogously for θ and ψ.

This model will give a rough estimate of the output of an IMU that moves along a trajectory given from a data set of spatial coordinates. The data serve as input and corrector to the filter, and is used as a faster generator of different roads. An example of a path and the corresponding emulated IMU data is shown in Figure 6.2.

5.1.2 Virtual truck model

Virtual truck model (VTM) is a Simulink model, made to simulate the behaviour of a truck. The model is very detailed and there are existing plant models for most of the present trucks. To simulate road and sensor data, the FMX block has been used to log the corresponding signals. Roads are defined by steering angles and height profiles. The output from the IMU block is used as input to the filter and the simulated GPS signals work as correctors. The corresponding configuration was also done for the 2D map corrector.

VTM will be closer to reality since it models a full truck, including for example suspension and other dynamics. Hence, these data will serve as a bridge between the simple point mass moving along a trajectory in the IMU emulator and the reality when data is collected from a truck driving on public roads.

5.2 Testing on a truck

In order to test the filters on real data, a test on a Volvo FMX truck was carried out at Volvo's test track at Hällered. Figure 5.1 shows how the IMU and GPS were mounted during the tests. The GPS was mounted on the driver's cabin while the IMU was mounted under the truck bin. The right panel of the figure shows the GPS used during the testing.

While deciding and specifying the different tests, the purpose was to, on the one hand, catch the normal behaviour of a truck that executes tasks that are in line with the ones specified by the applications of the project. On the other hand, it is also important to test the robustness and limitations of the filter by running

Figure 5.1: *The GPS was mounted on the driver's cabin as shown by the black arrow in the left panel. The white arrow points at the location of the IMU. The right panel shows the GPS used at the loggings.*

it on some extreme and constructed cases. The first set corresponds to the normal operating modes for an off-road truck and the second will test the robustness of the filter by constructed and hard tasks. A more thorough motivation for the different driving modes are given below. The processed results from the different data sets are presented in Chapter 6.

Normal driving

Regarding the goal of making the filter functioning on general vehicles and general cases, it is of course important to run it at normal driving. For a truck, normal driving can correspond to different scenarios depending on the purpose. Moreover, in the test logs of this class, the focus is chosen to be on normal driving on country roads and high ways, with a speed range from 20 to 90 km/h, but also on slower speeds in terrain and on very uneven roads. The uneven roads are chosen to get noisy data in all and to make the vehicle slip on the surface. Another goal is to get variations in all three dimensions, by driving along a road with hills and turns.

Extreme driving

This class of test logs was constructed to test the performance of the filter at very specific scenarios. The goal was to create cases where the limitations of the filter become more visible. Another goal was to let the filter process data of extreme cases to see how it handles the worst cases for example in angular movements. The filter that corrects the states in two dimensions is sensitive to large acceleration so this was also tested for during the logging.

Chapter 6

Results

The results obtained from this thesis are divided into the various correction and test cases. The particle swarm optimization is incorporated in all results and will be discussed continuously along the chapter. The filter with options of full correction, 2D map correction and PSO was first implemented in MATLAB. To recall, the 2D map correction is the case when correction is done for position, velocity and heading in two dimensions. The basic filter was also built as a Simulink function in order to have a first version running on a truck in real-time. The complete filter, which was used to generate the results, was implemented in C++.

6.1 Results using data from IMU emulator

Using the developed IMU emulator, the accelerometer and gyroscope data are derived given a certain road input. The algorithm for this is presented in Section 5.1.1. In this section, one of these roads are chosen to present the results from the full-correction algorithm. The chosen object is a spiral road, with a constant turn to the left and a constant elevation, which could correspond to a car park. This is a quite extreme road input with large acceleration. All simulated data have added Gaussian noise with standard deviation corresponding to that of an IMU. The data also include added gyroscope bias since it is an important feature of the full-scale and real-time version of the filter. This way it is possible to test the bias estimation and robustness of the filter.

A simple performance test of the gyroscope-accelerometer EKF is carried out on a straight road with constant speed, i.e. no linear acceleration. Noise is added to both signals, and bias is added to the gyroscope data. Figure 6.1 shows the difference between not using the updating and using the update of the quaternion. No update is carried out with the SEKF.

Figure 6.2 shows results obtained when running the complete filter on a spiral path. In order to analyze the behavior in more detail, the error of the correction signal and the corrected states are computed. The error from this simulation and how it is evolving with time is presented in the middle panel.

In Figure 6.3 one can see a result obtained from using map matching correction. Some results from switching dynamically between different correction cases are shown in Figure 6.4. First, the correction is done merely for position, then for velocity, for the quaternion and finally for all states.

6.2 Results using data from truck

To fully test the performance of the filter, a number of tests were carried out at a test track. The complete description of the different cases is presented in the previous chapter. The results include both GPS correction and 2D map correction. Below, the results from three data sets are presented where GPS correction is applied on all correction steps. They are shown here for comparison when the more advanced flexible sensor fusion

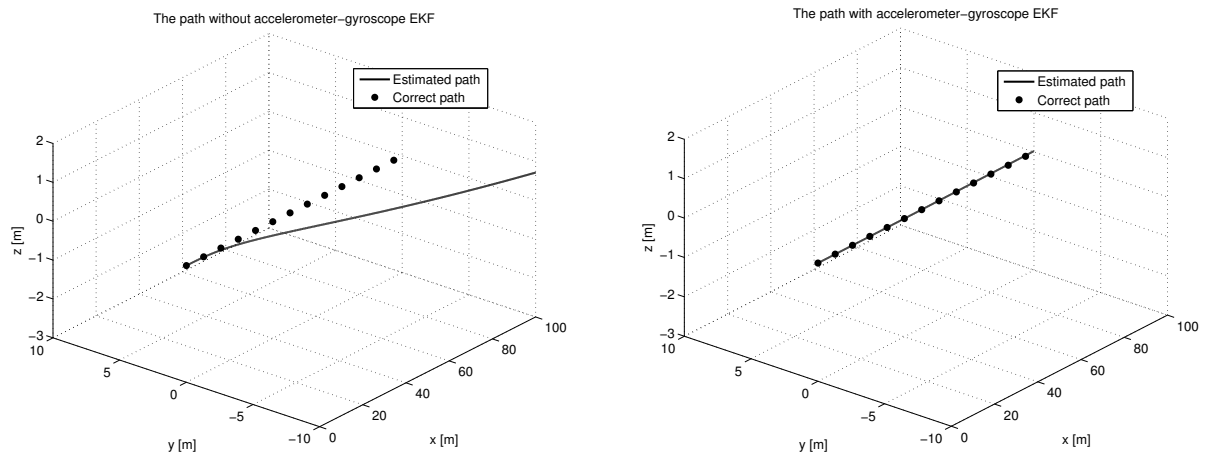

Figure 6.1: *Filtered data from a simulated straight road using the IMU emulator. To the left: the result from only using the prediction step of the SEKF. To the right: the result from using only the additional gyroscope-accelerometer EKF. Hence, the additional EKF improves the result and makes the drift smaller. The bias added to the gyroscope data was in this run* $\boldsymbol{b_g} = [0.4, 0.5, 0.6]$.

algorithm is used.

The first data set corresponds to the construction track at Hällered. There, a number of different obstacles are present to create as much bumps and noise as possible. This is a challenge for the filter since movements are present in all directions, as well in position and attitude. The results are given in Figure 6.5. The next data set corresponds to large acceleration in linear motion followed by harsh braking, driving in a straight line. This provides large linear accelerations. The results when using GPS correction is shown in Figure 6.6 while results when using a 2D map corrector are displayed in Figure 6.7. Furthermore, data corresponding to a country road is processed, showing the results when using GPS and 2D map correction in Figure 6.8.

Finally, the country road data set is also used for a statistical evaluation of the algorithm. This is done by running the filter on the same data 100 times and then do a statistical analysis of the results. Visualizations of the standard deviations when running the country road 100 times on two correction cases is shown in Figure 6.9. The upper panel showing the nine states representes the GPS corrector, and the lower panel shows the 2D map corrector. The standard deviations are plotted through time in order to show how the different estimations differ from each other during different parts of the drive. This will give a hint about how robust the estimator is.

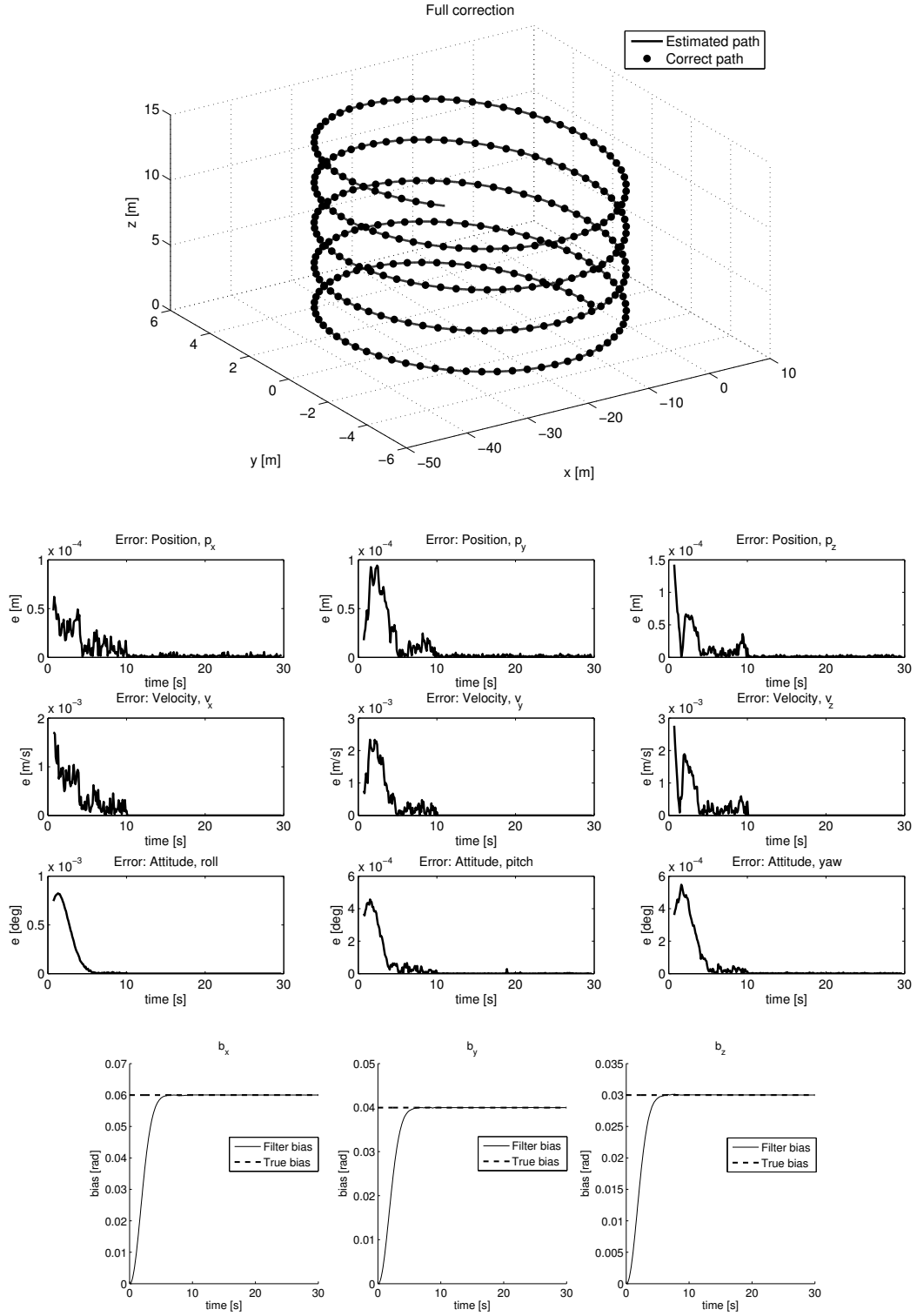

Figure 6.2: *Results from full GPS corrector on simulated data from the IMU emulator. The vehicle turns at a constant rate to the left and the velocity magnitude is constant. Upper panel: the estimated path as a solid line and the correct path as filled dots. From a large scale perspective, the estimation follows the correct path. Middle panel: the error between filter estimation and the GPS signal for position, velocity and attitude in three dimensions. The error drops significantly in position and velocity after ten seconds, when a PSO is performed to optimize the process noise covariance. The error also decreases before due to the properties of the Kalman filter. Lower panel: the estimation of the added bias to the gyroscope data and the corresponding state convergence. The added bias for this run was* $\boldsymbol{b_g} = [0.06, 0.04, 0.03]$.

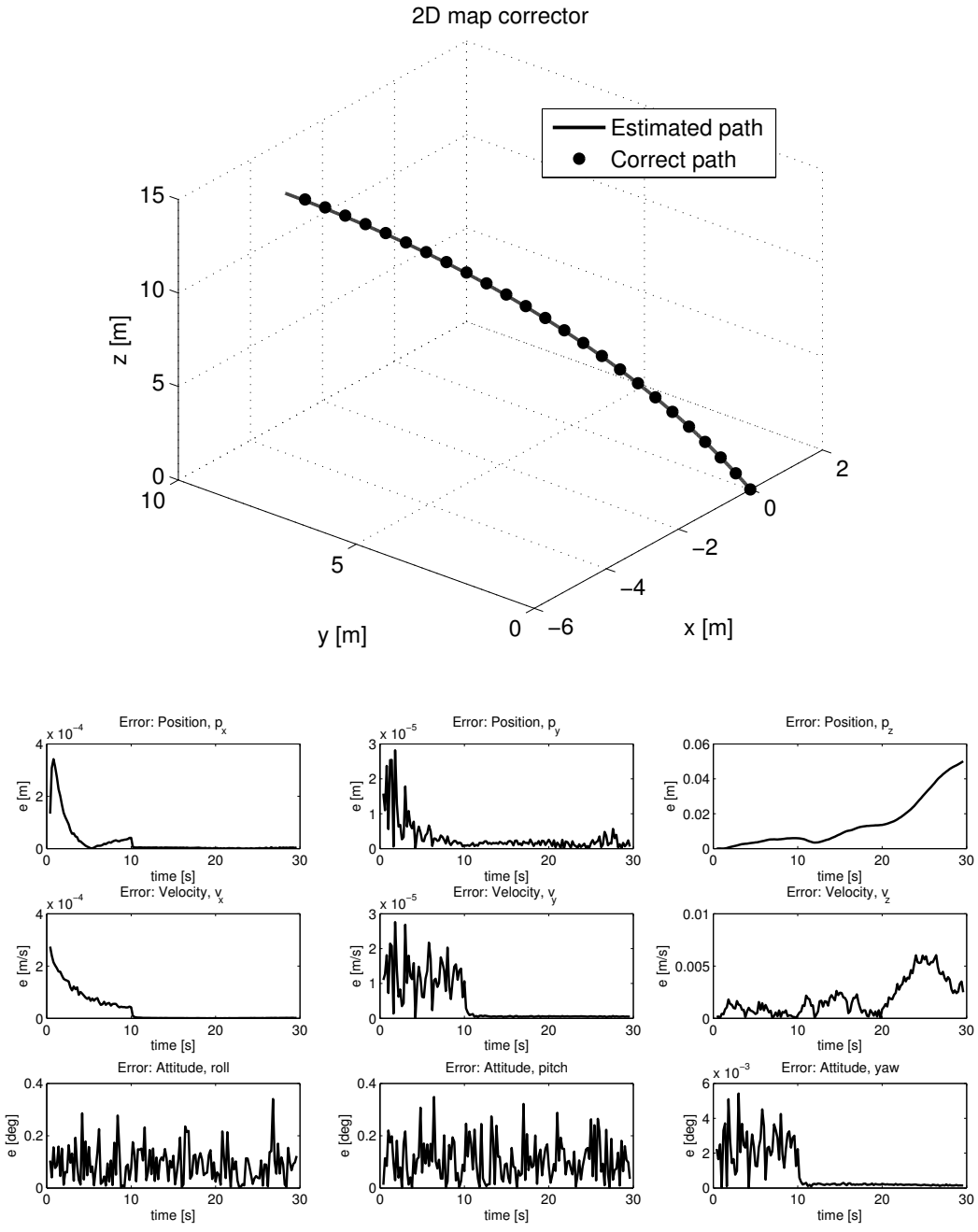

Figure 6.3: *Results obtained when using a 2D map as corrector on simulated road data. Upper panel: the correct path as filled dots and the estimate as a solid line. Correction was carried out every* 25^{th} *IMU sample. A small error of approximately 5 cm is seen in z-direction. In the lower panel: the error in position, velocity and attitude in three dimensions. Since the correction is done in 2D, a drift is visible in the* $z-$*direction as well as in roll and pitch. The error is however rather small in this run because of the slow velocity and the gyroscope-accelerometer EKF. The added gyroscope bias was* $\boldsymbol{b_g} = [0.06, 0.04, 0.03]$.

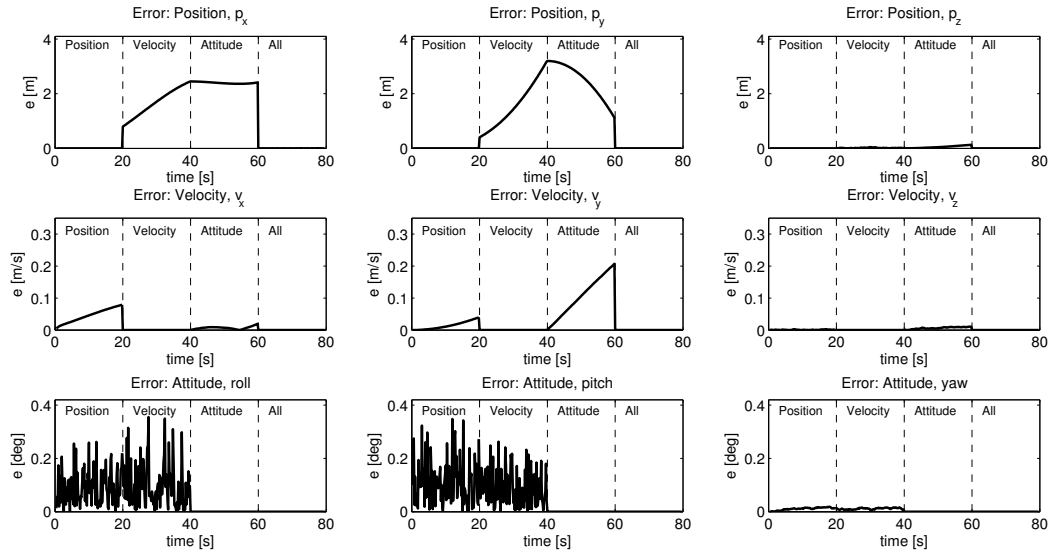

Figure 6.4: *The error evolution when the corrector signals are dynamically changing. Every* 20^{th} *second, the correction signals are changed in the order: position, velocity, attitude, and all signals present. The error is displayed for position,velocity and attitude and it is clear that the error drops significantly for the state which is being corrected for. The correction is done in all three dimension for each slot. This result was made from a simulated road using the IMU emulator and the added gyroscope bias was* $\boldsymbol{b_g} = [0.06,\ 0.04,\ 0.03]$.

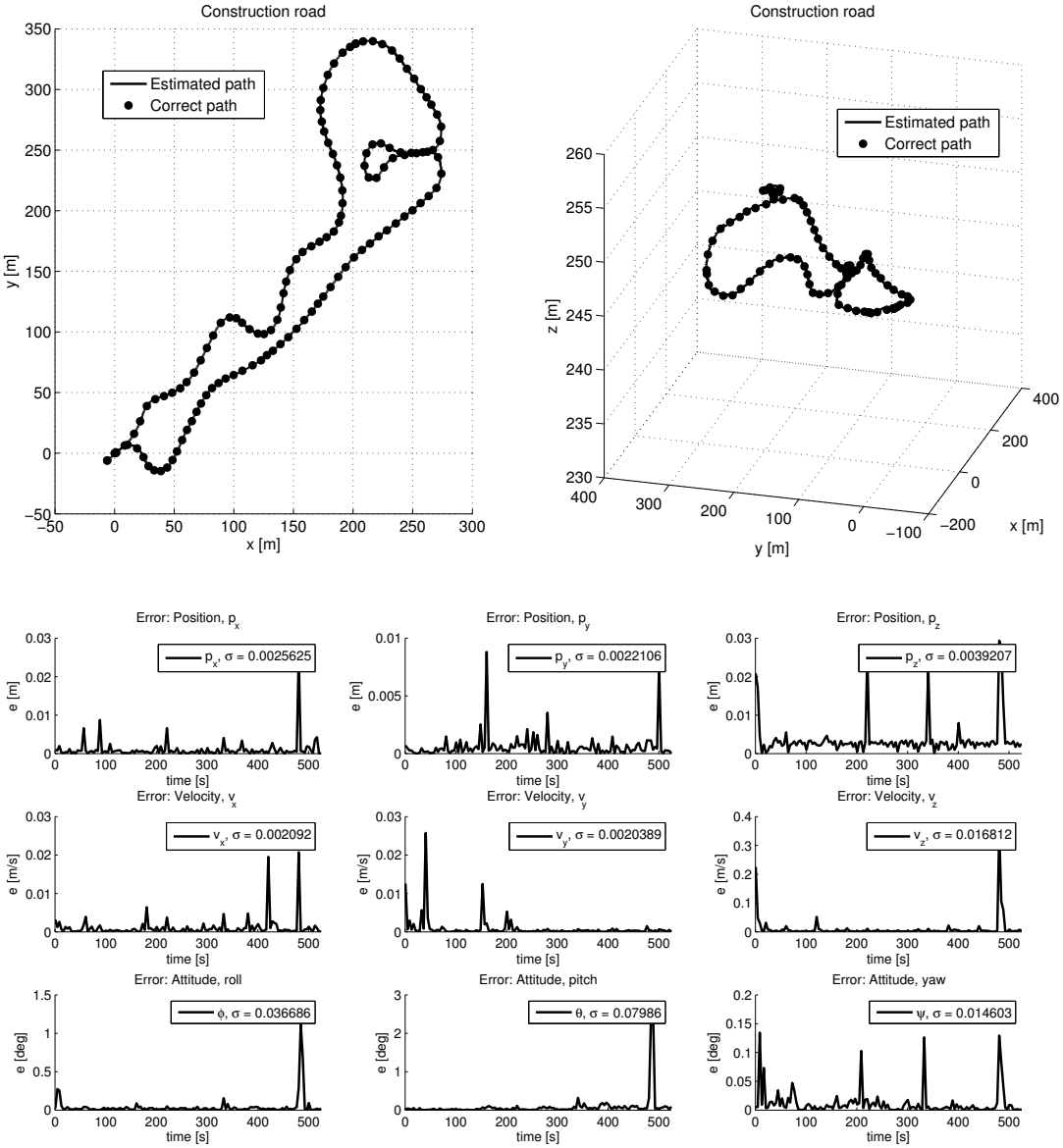

Figure 6.5: *The results from full GPS correction using the data set that represents the construction road at Hällered. Correction at approximately every* 25^{th} *IMU sample. Upper panel: the estimated path from the filter as a solid line and the raw GPS path as filled dots. Visualised in two views for comparison. To the left: xy-view. To the right: full 3D view to see the height difference. Lower panel: the estimation error obtained from the filter compared to the GPS signal. For each state, the average error with respect to time is presented as* σ *as a measure of accuracy. The error varies at the magnitude* $10^{-3} - 10^{-2}$ *for the different states which is acceptable, since the sensor yields very noisy data on this specific road. The process noise covariance is tuned every* 10^{th} *second by the PSO.*

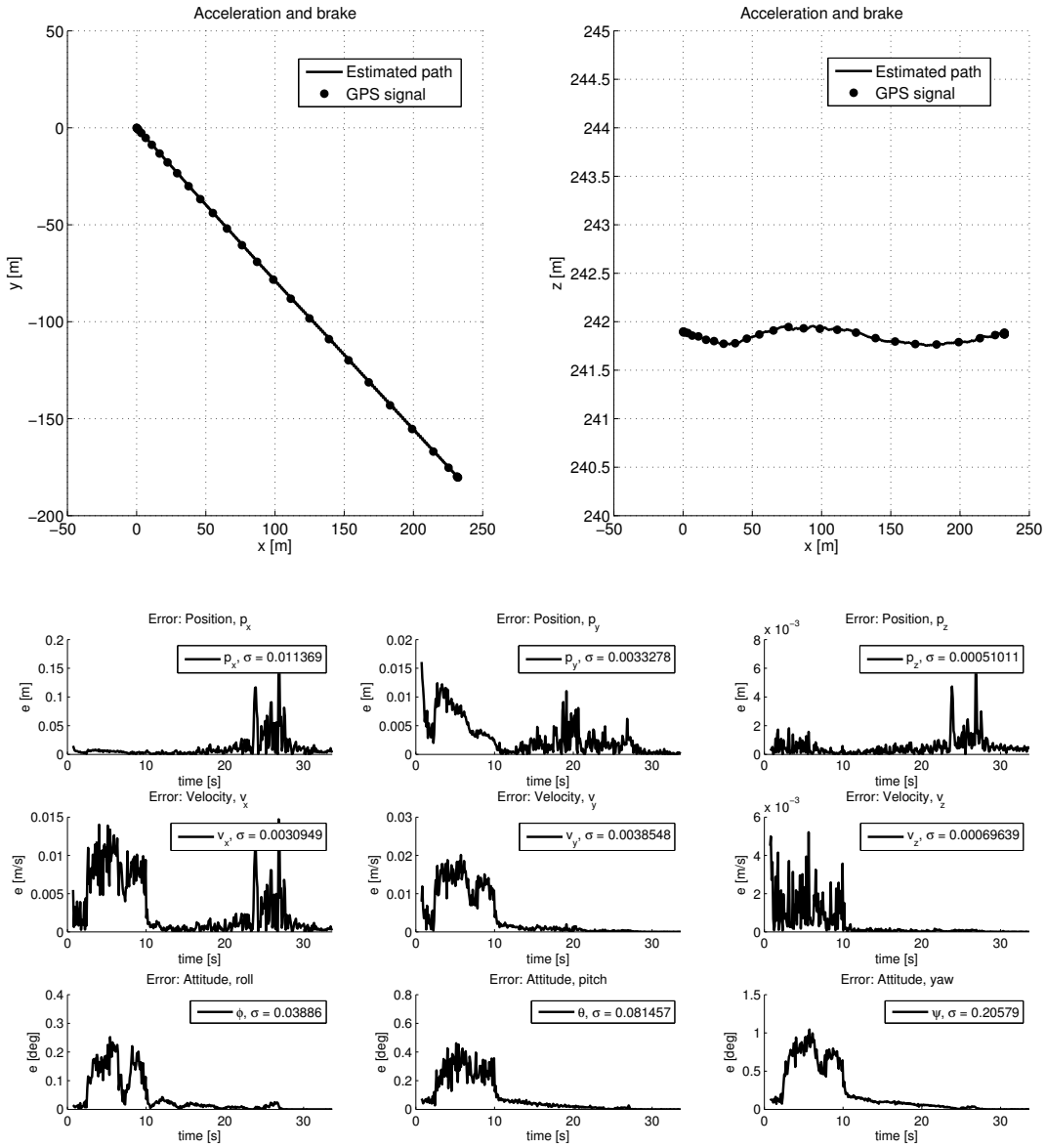

Figure 6.6: *The results from GPS correction using the data set representing large acceleration and hard braking. Upper panel: the estimated path and the GPS path in two views. Left: planar coordinates. Right: xz-plane where small variations in height are visible . Lower panel: the estimation error in three dimensions obtained from the filter for GPS correction. For each state, the average error with respect to time is presented as σ as a measure of accuracy. The error varies at $10^{-4} - 10^{-2}$ m in position, $10^{-4} - 10^{-3}$ m/s in velocity and $10^{-2} - 10^{-1}$ degrees in attitude. The error drops at the 10 second mark due to the performing of a PSO of the process noise covariance. The error increases and gets noisier in position after 25 seconds due to the hard braking of the truck.*

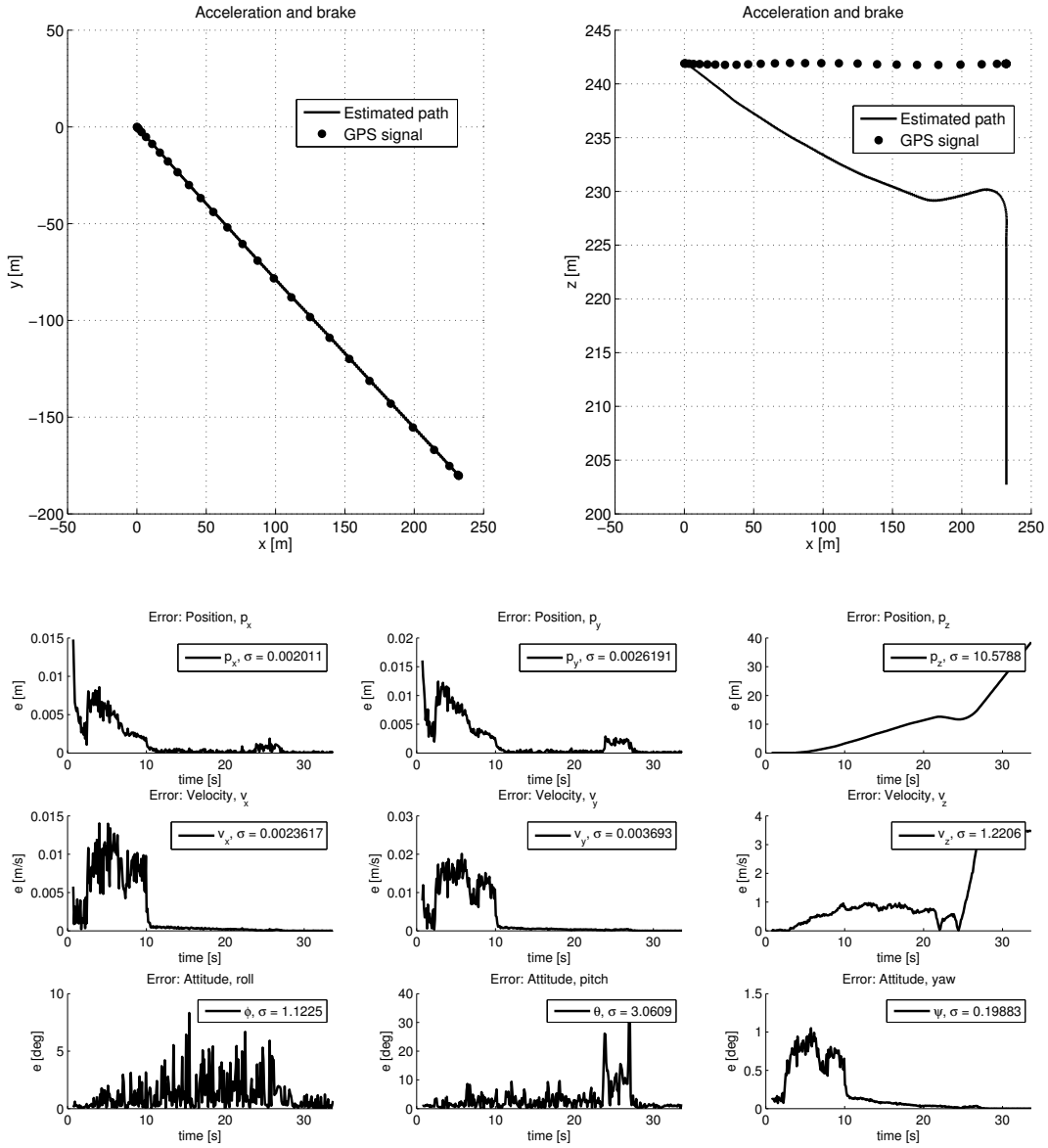

Figure 6.7: *The results from 2D map correction using the data set representing large acceleration and hard braking. Upper panel: the estimated path and the GPS path in two views. Left: planar coordinates. Right: xz-plane where a large drift in height is visible . Lower panel: the estimation error in three dimensions obtained from the filter for GPS correction. For each state, the average error with respect to time is presented as σ as a measure of accuracy. For the x and y direction, the error varies around 10^{-3} m in position, around 10^{-3} m/s in velocity and 0.1 degrees in yaw. The error in z-direction is larger as expected, as well as roll and pitch. The error drops after 10 seconds due to the performing of a PSO of the process noise covariance. The error increases and gets noisier in position and pitch after 25 seconds due to the hard braking of the truck.*

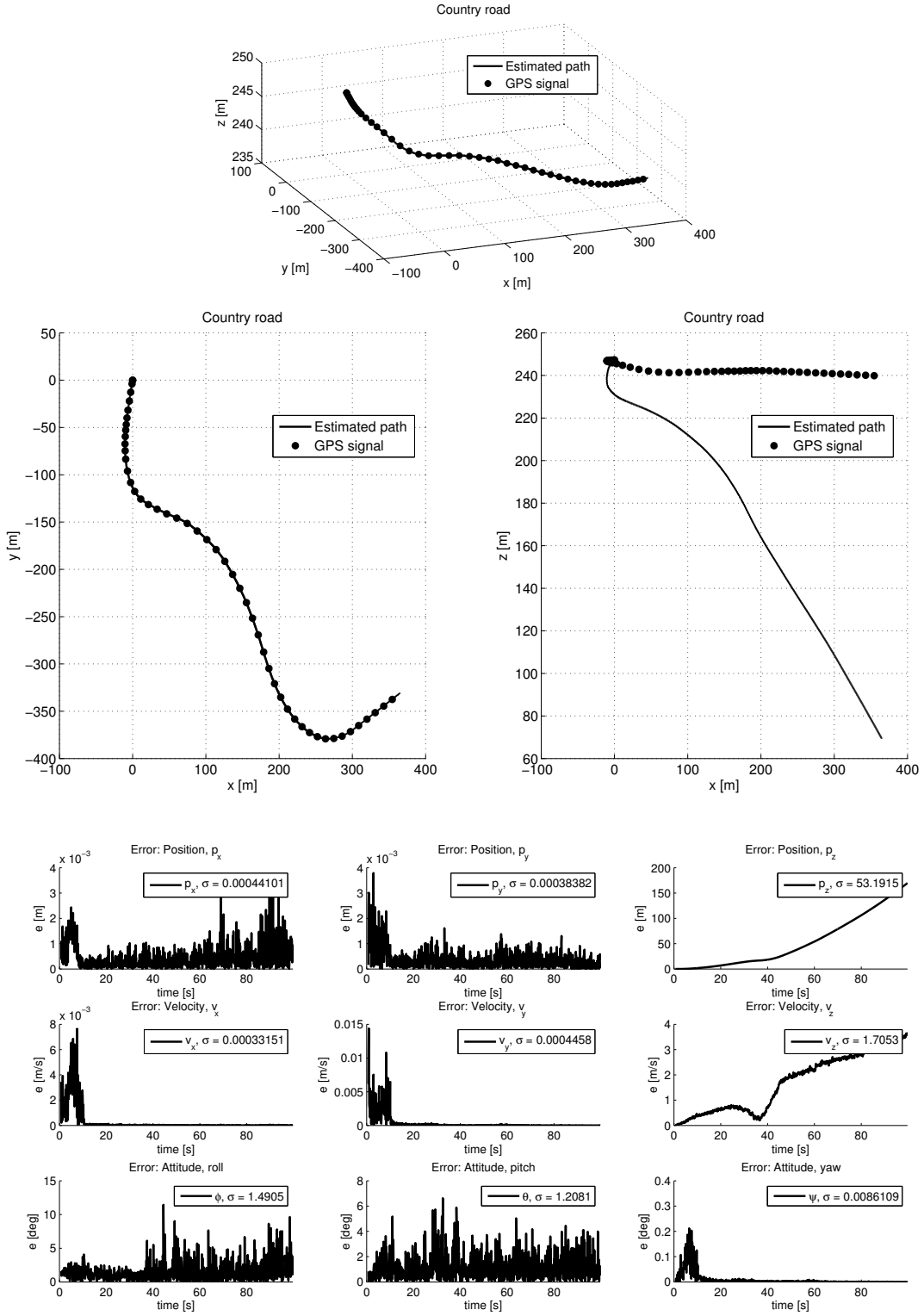

Figure 6.8: *The results from the data set representing a country road. Upper panel: the estimated path and the correct path from GPS data using full GPS correction. Middle panel: the estimated path and the GPS path using 2D map correction in two views. The estimator follows the GPS signal on a large scale perspective in planar coordinates, but not in height as visible in the right figure. Lower panel: the estimation error obtained from the filter when using 2D map correction for position, velocity, and attitude in three dimensions. For each state, the average error with respect to time is presented as σ as a measure of accuracy. The error in position of x and y is of magnitude 10^{-4} m, and in yaw 10^{-3} degrees. For the z-direction, the error drifts as expected. PSO is carried out every 10^{th} second to tune the process noise covariance.*

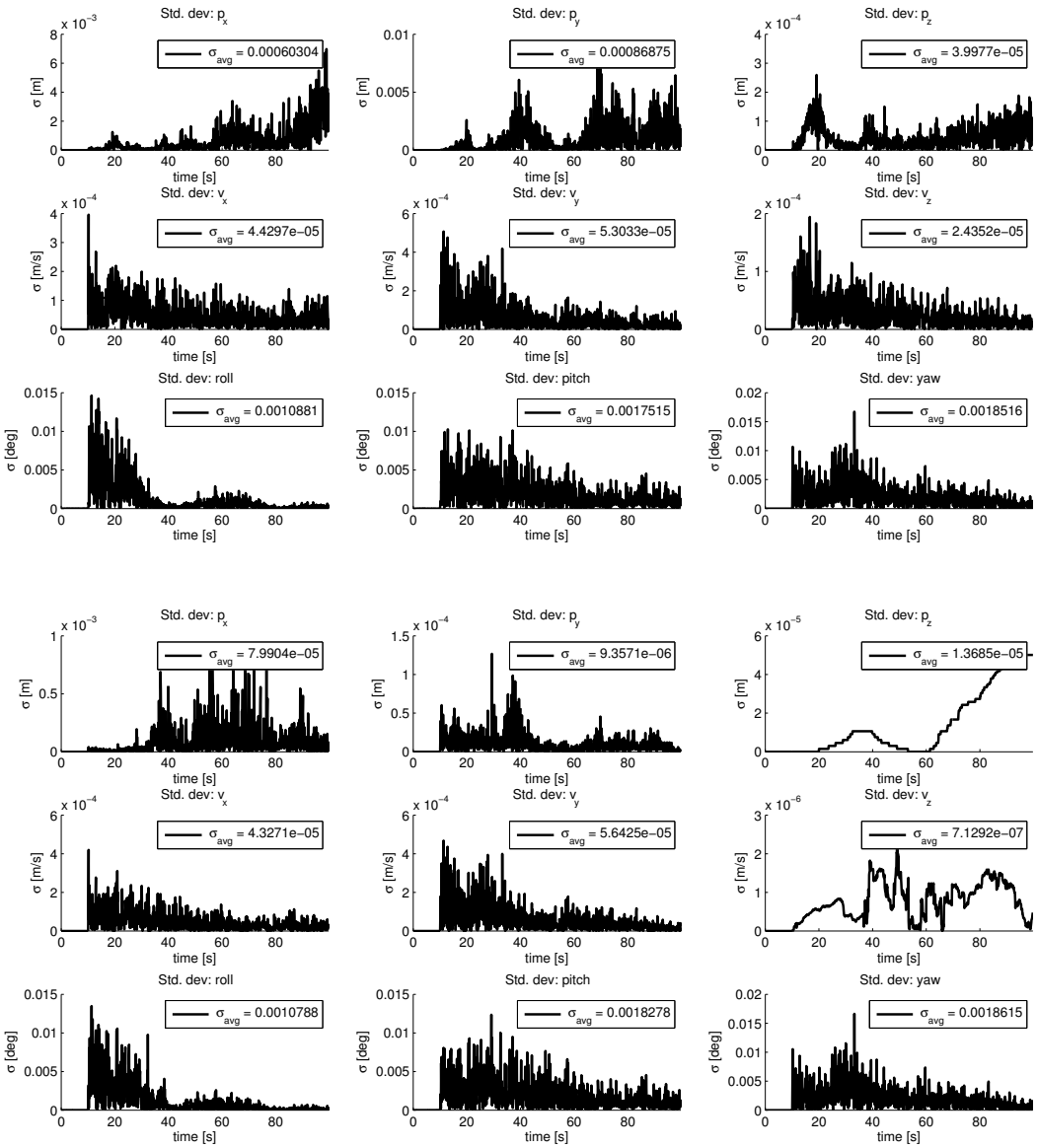

Figure 6.9: *Standard deviation of error in position, velocity, and attitude generated from running the algorithm 100 times for two different correction cases. The filter estimations are compared with GPS data to analyse the performance. For each case, the time averaged standard deviation is included for every state and represented as σ. Upper panel: GPS corrector applied to country road with the corresponding standard deviations for the correctable states. Lower panel: 2D map corrector for country road, i.e. the same input data as above, but with correction only in x, y, and yaw.*

Chapter 7

Discussion

This chapter will thoroughly discuss the generated results presented in the previous parts of the report. The discussion will first consider the performance on simulated data, generated from the IMU emulator. Then, the results from correcting the filter with 2D data as well as dynamically changing the correction signals will be evaluated. To sum up, the performance on real data logged from a truck is discussed before the overall conclusion is presented as well as future work on the topic in subject.

7.1 Evaluation of the ASTEK filter on simulated data

Figure 6.1 shows how the gyroscope-accelerometer EKF corrects the quaternion on a straight road. The left panel shows how the estimated states diverge fast, whereas the right panel shows a better estimation of the path. There is no correction of velocity or position here, the estimation is retrieved from the prediction step of the filter. The divergence in the left panel is mainly due to the lack of correction and the addition of bias in the gyroscope data. In the right panel, where the gyroscope-accelerometer EKF is used, the estimation follows the path without diverging. This simple test clearly shows that the EKF is able to handle and estimate bias, which is important since it always is present from a hardware point of view, i.e. there is no way of calibrating the sensor to get values with no bias. Consequently, the accelerometer-gyroscope EKF is better than only using prediction for estimating attitude.

When using the complete filter on the spiral road, Figure 6.2 shows that the estimation seems to follow the true path closely. The states are estimated with high accuracy, and the lower panel shows that the bias of the gyroscope also is tuned in well. The good behavior of the estimations is expected when all possible correction signals are given. Given the error, it is clear that the performance of the filter is acceptable, since the error is at a magnitude of $10^{-6} - 10^{-3}$ meters, meters per seconds or degrees. A particle swarm optimization is carried out every 10^{th} second. This evident in the plots, especially at the 10 second mark, where the error drops significantly for all states.

7.2 2D map corrector

The correction using a 2D map works as expected, which can be seen in Figure 6.3, 6.7 and 6.8. The first thing to notice is that the estimation is considerably worse in z-direction. This is because there is no correction in z-direction and when no corrector is present, the error will drift. On the other hand, it follows well in x- and y-direction. The z-direction is dependent on the angles since the accelerometer data is rotated using the quaternion. Therefore it is important to have a good estimate of roll and pitch so that the z-direction does not diverge faster. Roll and pitch are corrected using the gyroscope-accelerometer EKF which works well when the acceleration is modest, as in Figure 6.3. However, when the vehicle is exposed to large linear acceleration, the

additional filter is given larger measurement noise covariance and its importance will be reduced. Therefore, the z-direction will drift more. Thus, the 2D map correction will produce worse results in such circumstances. This can be seen in Figure 6.7. The roll is highly affected since the linear acceleration is larger which affects measurement covariance matrix so that the accelerometer is not trusted as much when correcting for roll and pitch. This, together with prediction errors from gyroscope bias give the larger errors. Also, at the time of braking, the pitch estimate is worsened, since the cabin of the truck where the GPS is mounted, will suffer from more noise and movement which yield a worse corrector signal. This will be discussed more further on. However, in applications such as driving in restricted areas or driving during short periods where GPS-satellites cannot be reached; the z-direction, roll, and pitch are perhaps not the most important states to estimate. The most crucial estimates are position in $x - y$ and the heading. These states are well estimated in the cases presented here. Moreover, which will be discussed below, the estimation becomes better once full correction is reestablished.

7.3 Dynamically changing the corrector signals

In Figure 6.4 one can see how the estimations instantly increase in error when there is no correction present, and how the error is decreased after receiving corrector signals yet again. During the first 20 seconds, the filter only corrects for position which yields an increasing error in velocity and attitude, while the error of position is kept low. When instead the correction shifts to velocity, the position error is increasing while the velocity error drops etc. One can note the sudden change in position just as the correction changes from position to velocity. This is due to the harsh correction of velocity which influences the position as well. Since the position is well estimated and has no corrector signal present at this point, a large change in position will result in a sudden and aggressively increasing error. The attitude is kept constant during this break point due to the gyroscope-accelerometer EKF. When the attitude is corrected for, the velocity slowly starts to increase in error. When all states are corrected for, all errors drop significantly. This serves as a verification that the dynamic change in corrector signals works as expected.

7.4 Performance of the ASTEK filter on a Volvo FMX truck

The results from logged data at Hällered are seen in Figures 6.6-6.8, where it is evident that the estimator follows the path from a large scale view when using the GPS correction.

Figures 6.6 and 6.7 displays the results from large acceleration and harsh braking. As seen, the error is larger in position and attitude at the 25 second mark. This is where the hard braking occurs. The truck cabin moved significantly during the brake and provided a large roll and pitch angle while the rest of the truck held the same angle throughout the braking. The GPS was mounted on the cabin whereas the IMU was mounted on the chassis, below the truck bin. Therefore, the GPS signal recorded a much larger pitch than the IMU predicted and hence, the difference between the estimated signal and the GPS signal is larger. The difference between the acceleration-braking when using GPS corrector or 2D map corrector is seen especially in the RMSE of the attitude, where the RMSE is [0.0388, 0.0815, 0.206] degrees for the GPS corrector and [1.12, 3.06, 0.199] degrees for the 2D map corrector. The RMSEs for the yaw angles are more alike in the two corrector cases, whereas the RMSEs for roll and pitch angles are larger in the 2D map case. The biggest difference is seen at the point of braking, where the error between GPS and estimation attitude is grows more for the 2D map corrector than for the GPS corrector. This is, as mentioned, because the cabin moves and creates other pitch and roll angles, than those of the truck body. Hence, the difference between the GPS signal and the estimated signal should grows. In scenarios where the cabin moves a lot, the gyroscope-accelerometer EKF could actually produce a more accurate estimation than the GPS. In a completely autonomous vehicle, however, the cabin will not move and this will therefore not be a problem when mounting the corrector.

The construction track in Figure 6.5 is one of the most difficult tracks to estimate due to the large variations of the road. The layout of the road will cause movements of the cabin, and thus, obstruct the estimation. Since the location of the track is within a forest section, the GPS satellites will sometimes be out of reach, and the states will drift. In spite of these reasons, the over all estimation is quite good. Some spikes appear

in the error as a result from loosing the correction signals. The RMS errors are [2.56, 2.21, 3.92] mm for position, [2.10, 2.04, 16.8] mm/s for velocity, and [0.0367, 0.0799, 0.0146] degrees for attitude. This is because the truck moved slowly, only 5 km/h, which made it possible to perform good estimations. Due to the cabin movement, however, the measurement noise should in reality be larger than the GPS signal which should affect the difference between the GPS signal and the estimated states, as described above.

The country road seen in Figure 6.8 corresponds to normal driving on a road. The RMSEs when using a 2D map corrector are $[4.41 \cdot 10^{-4}, 3.84 \cdot 10^{-4}, 53.1915]$ m for position, $[3.32 \cdot 10^{-4}, 4.46 \cdot 10^{-4}, 1.7053]$ m/s for velocity and [1.49, 1.21, 0.00861] degrees for attitude. Note again that the z-direction diverges and how the error evolution of roll and pitch are quite constant although the RMSE are much larger than for yaw. This road shows how the filter works on trucks when neither the setting, the environment nor the driving mode is extraordinary. Therefore, these results serve as validation that the filter is functioning on real vehicles as well.

In Figure 6.9, the result from a statistical simulation is shown. The plots show the standard deviation of the estimations from 100 runs. The mean standard deviations for the 100 runs are $[6.03 \cdot 10^{-4}, 8.69 \cdot 10^{-4}, 4.00 \cdot 10^{-5}]$ m for position, $[4.43 \cdot 10^{-5}, 5.30 \cdot 10^{-5}, 2.44 \cdot 10^{-5}]$ m/s for velocity, and $[1.09 \cdot 10^{-3}, 1.75 \cdot 10^{-3}, 1.85 \cdot 10^{-3}]$ degrees for attitude. This implies that even though the algorithm is stochastic, it is rather robust in its performance and the results are stable. Note also that the standard deviation is zero until the ten second marks. This is because the filter generates the same results until the first optimization.

7.5 Conclusions

Given the purpose of the thesis, it is highly important to verify that the 2D map corrector algorithm is working properly. This will widen the usage of the state estimator, since it will not be dependent on a three-antenna GPS, but be more flexible. Therefore, the ASTEK filter will be compatible for usage in for example restricted areas where only maps are present as corrector. The estimator will however diverge in z-direction and generate a larger RMSE in roll as well as pitch, which is important to conclude, but also expected.

The data set representing acceleration and braking shows that good estimations are difficult to maintain when the acceleration is large, especially when using 2D map corrector. These results are general and expected, but also generated from an extreme case. Normal driving does not often include these types of accelerations, and if so, they are executed during shorter periods.

The construction road represents the most difficult case for the filter to perform well. Several aspects make this road a difficult data set. Due to location, the GPS signals could be out of reach. This creates spikes in the error, since the estimation drifts when no correction signals are present. In addition, the GPS, which is mounted on the cabin, is sometimes out of sync with the chassis-mounted IMU due to the large bumps and obstacles. In spite of these reasons, the accuracy of the filter is rather high, which probably is a result from the very slow speed of the vehicle.

As an attempt to emulate a general case of driving, data was logged at a country road at the speed 20 km/h. For both GPS correction and 2D map correction, the results are accurate and reliable. When the other data sets correspond to rather extreme driving, this case is applicable to normal driving. This concludes that one can expect good performance from the filter on a general road, where corrector in at least two dimensions is available.

7.6 Future work

There are some areas in this project that can be fine tuned or explored further. Some subjects of improvement are listed below, with accompanying explanations.

The particle swarm optimization for process noise covariance was implemented sequentially, called every 10^{th} second, in this thesis. Since it is computationally demanding, it would be beneficial to rearrange the system architecture and put the optimization in a parallel system. This way, the covariance could be updated whenever

a new and better solution is found, independently of the current filter operation. The particle swarm algorithm could also be improved by separating the particle performance into the different states. In the present version, although rare, one state could actually become worse after the optimization. This is due to the joint sum over the states. If some states decrease their errors enough, the sum will still be less even if one state suffers from a worse estimation. This is usually tuned in the following optimization, but in order to have a more robust algorithm, an idea could be to make the state performances independent from one-another.

Furthermore, one could expand the project even further by including tyre modelling. In such an expansion one would for example want to estimate and derive tyre friction, side slip and wheel torques. In this setup, wheel velocities could be used together with the steering angle of each wheel to estimate such states. This would, however, not be vehicle independent anymore since one would need length and width measurements from the specific vehicle and tyre data. Thus, the estimator would lose its generality. In order to slightly account for this one could make the expanded part of the estimator modular and introduce a flag to denote the current vehicle from a data base. Introducing such an extension would provide most of the interesting information needed to navigate the vehicle.

Finally, the main estimator developed in this thesis could be more robust by implementing an unscented Kalman filter (UKF) instead of the SEKF which was used here. [27] The UKF uses statistical sampling instead of Jacobians when approximating the non-linear system. Hence, this yields a more robust estimation for highly non-linear dynamics and decreases the risk of divergence. The UKF can also be used on dynamics with non-Gaussian noise. Using a UKF would, however, increase the computational complexity and having a large state vector would increase the risk of divergence. Therefore, one could divide the system into one highly non-linear part and one less non-linear part, and then estimate only the highly non-linear part with an UKF and the other with an EKF. One could also consider using a different differentiating scheme instead of Euler forward as was used in this thesis. This could for some cases make the prediction step more stable, for example if the IMU samples come in sparsely, the prediction may diverge more than when using for example Runge-Kutta. It seems however, as if the SEKF together with Euler forward is a good starting point when estimating states with the sensors used in this context.

References

[1] T. Akenine-Möller and E. Haines. *Real-time rendering*. 3rd ed. A K Peters, 2008. ISBN: 9781568814247.

[2] S. Antonov, A. Fehn, and A. Kugi. Unscented Kalman filter for vehicle state estimation. *International Journal of Vehicle Mechanics and Mobility* **49**.5 (2011), 1497–1520.

[3] A. R. A. Bacha, D. Gruyer, and S. Mammar. Neural Network Aided Adaptive Kalman Filtering for GPS Applications. *Intelligent Vehicles Symposium* **4**.1 (2013), 195–200.

[4] J. Bechtloff. Fast identification of a detailed two-track model with onboard sensors and GPS. *International Munich Chassis Symposium* **5**.1 (2014), 303–325.

[5] Y.-J. Cheon. Unscented Filtering in a Unit Quaternion Space For Spacecraft Attitude Estimation. *International Symposium on Industrial Electronics* **-**.- (2007), 66–71.

[6] J. Farell and M. Barth. *The global positioning system and inertial navigation*. 1st ed. McGraw-Hill, 1999. ISBN: 9780070220454.

[7] W. R. Hamilton. Letter from Sir William R. Hamilton to John T. Graves, Esq. on Quaternions. *Philosophical Magazine* **3**.25 (1844), 489–495.

[8] Q. Honghui and J. Moore. Direct Kalman filtering approach for GPS/INS integration. *Transactions on Aerospace and Electronic Systems* **38**.1 (2002), 687–693.

[9] M.-D. Hua. Attitude estimation for accelerated vehicles using GPS/INS measurements. *Control Engineering Practice1* **18**.7 (2010), 723–732.

[10] S. Julier and H. Durrant-Whyte. Process Mode s For The High-speed Navigation of Road Vehicles. *International Conference on Robotics and Automation* **1**.1 (1995), 101–105.

[11] S. Julier and J. K. Uhlmann. "New extension of the Kalman filter to nonlinear systems". Vol. 3068. -, pp. 182–193.

[12] D.-J. Jwo, C.-S. Chang, and C.-H. Lin. Neural Network Aided Adaptive Kalman Filtering for GPS Applications. *International Conference on Systems, Man and Cybernetics* **4**.1 (2004), 3686–3691.

[13] D.-J. Jwo and S.-C. Chang. Particle swarm optimization for GPS navigation Kalman filter adaptation. *Aircraft Engineering and Aerospace Technology: An International Journal* **81**.4 (2009), 343–352.

[14] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering* **82**.D (1960), 35–45.

[15] J. Kennedy and R. Eberhart. Particle Swarm Optimization. *International Conference on Neural Networks IV* **4**.1 (1995), 1942–1948.

[16] E. Kraft. A Quaternion-based Unscented Kalman Filter for Orientation Tracking. *Proceedings of the Sixth International Conference of Information Fusion* **1**.6 (2003), 47–54.

[17] P. Lamon and R. Siegwart. Inertial and 3D-odometry fusion in rough terrain - Towards real 3D navigation. *International Conference on Intelligent Robots and Systems* **2**.1 (2004), 1716–1721.

[18] J. S. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association* **443**.93 (1998), 1032–1044.

[19] S. O. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Technical Report* **1**.1 (2010), 1–31.

[20] J. L. Marins et al. An Extended Kalman Filter for Quaternion-Based Orientation Estimation Using MARG Sensors. *International Conference on Intelligent Robots and Systems* (2001).

[21] W. McCulloch and W. Pitts. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics* **5**.4 (1843), 115–133.

[22] R. van der Merwe and E. A. Wan. Sigma-Point Kalman Filters for Integrated Navigation. *Proceedings of the 60th Annual Meeting of the Institute of Navigation (ION)* (2004), 641–654.

[23] R. Mukundan. *Advanced methods in computer graphics: with examples in OpenGL*. SpringerLink, 2012.

[24] H. B. Pacejka. *Tyre and vehicle dynamics*. Butterworth-Heinemann, 2006. ISBN: 0080543332, 9780080543338.
[25] H. Ren et al. Vehicle State Information Estimation with the Unscented Kalman Filter. - **1**.1 (2014), 1–11.
[26] Y. Ren and X. Ke. Particle Filter Data Fusion Enhancements for MEMS-IMU/GPS. *Intelligent Information Management* **2**.7 (2010), 417–421.
[27] I. Skoog and P. Händel. In-car positioning and navigation technologies: a survey. *Transactions on Intelligent Transportation Systems* **10**.1 (2009), 4–21.
[28] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. 1st ed. MIT Press, 2000. ISBN: 978-0262201629.
[29] M. Wahde. *Biologically inspired optimization methods: an introduction*. WIT Press, 2008. ISBN: 1845643445.
[30] Y. Wang et al. Vehicle state estimation using GPS/IMU integration. *Sensors, IEEE 2011* **1**.1 (2011), 1815–1818.
[31] G. Welch and G. Bishop. An introduction to the Kalman filter (2006).
[32] Y. Yang and J. Farrell. Megnetometer and differential carrier phase GPS-aided INS for advanced vehicle control. *Robotics and Automation* **19**.2 (2003).
[33] G. Zunini. "Simultaneous Localization and Mapping for Navigation in Realistic Environments". PhD thesis. Royal Institute of Technology, 2002.