Multi-Rate Unscented Kalman Filtering for Pose Estimation

Using a car-like vehicle-platform

M. de Vries

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Multi-Rate Unscented Kalman Filtering for Pose Estimation

Using a car-like vehicle-platform

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering, Biomechanical Design, Bio-Robotics at Delft University of Technology

M. de Vries

February 22, 2019

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

The work in this thesis was supported by Accenda. Their cooperation is hereby gratefully acknowledged.

Delft University of Technology
Department of

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Multi-Rate Unscented Kalman Filtering for Pose Estimation

by

M. de Vries

in partial fulfillment of the requirements for the degree of

Master of Science Mechanical Engineering, Biomechanical Design, Bio-Robotics

Dated: February 22, 2019

Supervisor(s):

Dr.ir Manuel Mazo Espinosa

Ing. John Seiffers

Reader(s):

Dr. Manon Kok

MSc. Cees Verdier

Abstract

Pose estimation through fusion of Global Navigation Satellite System (GNSS) with secondary sensors has long been an established field. With the developments surrounding autonomous navigation over the past decade this topic has gained extra importance.

In the current literature GNSS based pose and localisation is often improved through fusion with either a Inertial Measurement Unit (IMU) or Vehicle Sensors (VS) with the goal of improving on stand-alone GNSS localisation results as well as dealing with GNSS outages. In this thesis however, all three of these sensors will be fused together using a cascade of a IMU orientation filter and a Multi-Rate Unscented Kalman Filter (UKF). This filter structure is evaluated using simulations and real-world data obtained using a created vehicle-platform.

The simulated results indicate that using a Multi-Rate Unscented Kalman Filter for pose estimation is promising as the filter, when configured properly, outperforms stand-alone GNSS receivers for pose estimation. However, the real-world experiments show that the used sensors lack accuracy and precision to obtain satisfactory results.

Table of Contents

List of Figures

List of Tables

Preface & Acknowledgements

While I was working at NEC in Japan, during the summer and fall of 2017 I started looking for a thesis subject. Thankfully, due to my previous employment as Mechatronics SA I came into contact with both Dr. ir Manuel Mazo Espinosa and Ing. John Seiffers who offered me a thesis subject in the field of autonomous driving and sensor fusion when my own department was too overcrowded to provide support.

During the course of my thesis I also met Dr. ir Manon Kok who helped me with the technical aspect of the Multi-Rate UKF and provided me with useful insights where pose estimation and Inertial Measurements were concerned, especially where the dubious implementation of Mahony's filter algorithm is concerned. Thanks a lot Manon!
I would also like to thank my other committee member Cees Verdier MSc. for taking the time to check my report.

During my research I spent most of my time in the DCSC Lab where I worked together with Thomas, Boaz, Bram and Robert who, although their subjects differed greatly from my own, were always in for a cup of coffee. I really enjoyed our talks together. Last but not least I want to thank my parents and sister for their continued support during my long-lasting student career.
Dear reader, I hope the topics discussed in this report will be of use to you for your own research.

Kind Regards,

Maarten de Vries

Chapter 1

Introduction

In the introduction the motivation and goals for this research will be given. Firstly, **??** will provide the reader with the motivation behind this research whereafter section 1-2 will lay out the goals and proposed solution contained in the thesis. Lastly the structure of the thesis is given in section 1-3.

1-1 Motivation

In the last couple of years the development of level 5, or fully autonomous vehicles, has been a hot-topic(Forbes 2018, [1]). Well-known companies involved in the race to level-5 autonomy are Google, Tesla and Uber together with the classical car manufacturers such as General-Motors and BMW.
On a smaller scale, Accenda, a tech company situated in Delft, wants to do research in the field of (semi)autonomous vehicles and they require a hardware and software solution to do so.

For the development of fully or semi autonomous vehicles a number of hurdles need to be overcome. One of these hurdles is accurate pose (location and orientation) estimation.

Apart from navigation in the classical sense, (semi-) autonomous vehicles need accurate localisation for the implementation of their higher level control architecture (i.e., path planning, and decision making [2])
The default system used for absolute localisation is a Global Navigation Satellite System (GNSS) such as Global Positioning System (GPS). However, a stand-alone GNSS does not offer the precision and accuracy required for autonomous navigation [3]. A commonly used approach to improve GNSS accuracy is to is to employ additional sensors to improve the localisation by means of sensor-fusion. Sensors often used for this purpose are Vehicle Sensors (VS) that measure things as angular velocity of individual wheels and motor RPMs or/and an Inertial Measurement Unit (IMU)

This fusion of different sensors is performed using a wide range of possible solutions of which the Kalman filter developed by Kalman et al. [4] is probably the most well known and employed [5–10]. Over the years multiple versions of Kalman filters have been proposed to deal with non-linear systems such as the Extended Kalman Filter (EKF) which is often attributed to S.F. Schmidt [11] and later the Unscented Kalman Filter (UKF)(Julier et al [12–14]).

Pose estimation filters using GNSS/VS/IMU as measurments often are combined with a bicycle model. Due to the non-linear nature of this bicycle model often a non-linear Kalman variant is employed. For example Melendez-Pastor [15] propose a single update-rate EKF based fusion of GNSS with variety of vehicle sensors through use of simple kinematic bicycle model. A similar approach has been undertaken by (Wang et al [16]) with the extension of using bezier spline interpolation to deal with the slower update rate of GNSS receivers. The paper furthermore warns of using Multi-Rate EKF's as they generally lead to a downgrade of performance and may cause instability.
Multi-Rate non-linear Kalman for navigation has been investigated Armesto et al [17]. They use an output hold method to deal with unused sensor inputs and conclude that Multi-Rate UKF is in their application superior to Multi-Rate EKF.

Apart from the fusion of different sensors to improve the GNSS localisation, a way to deal with GNSS outages, which can occur due obstructions, a lack of available satellites or other reasons is important. Literature concerning Kalman filtering with intermittent observations [18–20] use a Bernoulli based distribution to model measurement arrival to show that a model dependant critical bound exist, that when crossed, results in state estimate divergence. Practical approaches to GNSS intermittency often resort to process model only updates when GNSS data is unavailable.

The IMU often used in these fusion schemes is itself capable of providing a measurement for pose estimates as described by Kok et al. [21] among others and should therefore be considered an important part of the overall fusion solution.

1-2 Thesis Goals and Description

Accenda, a tech-company situated in Delft, the Netherlands seeks to broaden its scope and wants to do research into the field of autonomous driving. For this purpose Accenda wants a Hard & Software platform as well as an integrated pose estimation and localisation solution. These two goals will be discussed in depth in the following two subsections

1-2-1 Goal I: The Creation of A Hard & Software platform

This platform will consist of a vehicle with integrated sensors for localisation, such as a GNSS receiver, IMU as well as VS. Part of the development will be the realisation of a Python and C++ code-library to operate and log the output of the sensors. Furthermore, the created hard and software platform must be modular, low-cost and well documented.
To achieve this goal hardware must be selected, a proper code libraries must be set-up as

well as an analysis of the maximum achievable data-rate with the chosen sensors must be conducted.

The description of the work related to the achievement of this goal laid out will be described in chapter 2.

1-2-2 Goal II: Development of a pose estimation and localisation solution

After the first goal is achieved and a working Hard& Software platform is created a filter that converts the raw observations into a pose estimate must be realised. The filter in question should satisfy the following requirements:

1. Modular by design
   A modular filter will a allow future improvements as well as the inclusion of more sensors.

2. Be capable of handling variable data-rates
   Due to differing data-rates of the to be used sensors it is important that the filter can deal with differing data-rates.

3. Handle sensor intermittency
   Sensor intermittency is an important fact-of-life for GNSS based navigations systems. When driving in urban areas or through tunnels GNSS data may not/ or is not available. In this case the system should be able to rely on its 'dead-reckoning' capabilities. Therefore it is important that the state estimates the filter produces are usable in these situations.

Apart from this, the IMU itself can be used to obtain an orientation estimate. For the purpose of creating a pose estimation and localisation solution it is beneficial to investigate the optimal way to filter IMU data.

Based on the literature the development of a Multi-Rate Unscented Kalman filter using a non-linear kinematic bicycle model should allow for the required modularity as well as the ability to deal with GNSS outages.

1-3 Structure of the Thesis

The structure of the thesis is as follows: In chapter 2 the hard and software platform will be discussed in more detail to give the reader background information on the underlying hard and software for the following chapters. This chapter will also detail the way the first goal of the thesis is achieved. The mathematical definitions used in the thesis will be briefly described in chapter 3. This chapter will also give a description of the used 'frames' and references of the vehicle-platform.
In chapter 4 the UKF structure used in the following chapters will be given as well as a set of experiments to determine the used IMU orientation filter.
The Multi-Rate UKF used for localisation and pose estimation will be described and evaluated using simulations in chapter 5. The following chapter, chapter 6 contains the real-world

experiments obtained using the vehicle-platform. Hereafter, chapter 7 will sum up the results and the added value of the created solution to the available literature. Furthermore, this chapter will present possible improvements as well as provide avenues for future research. Lastly, chapter 8 will restate the goals of the research and summarise the achieved results.

Chapter 2

Hardware and Computer Architecture

As one of the goals of Accenda is the development of a positioning system, the choice and design of hardware and computer architecture is an important step in achieving this goal. Therefore this chapter will give an overview of the used hardware and software. For more details about the software itself it is recommended to read the documentation of the software. The first section, section 2-1, will explain the choices for the hardware used in the thesis. After which section 2-2 in broad strokes describes the software. Lastly, section 2-3 and section 2-4 describe the used methods for communicating between the hardware as well as give an evaluation on the performance of said methods.

2-1 Hardware

For the localisation solution a multitude of sensors is required. These sensors include an Inertial Measurement Unit (IMU), a Global Navigation Satellite System (GNSS) receiver and a set of vehicle sensors for the determination of the steering angle and the vehicle speed. In this section the selection of the specific sensors will be explained.

2-1-1 IMU Selection

The choice for an IMU was based mainly based on cost and the requirements of being 9-Degrees of Freedom (DOF) (for Gyroscope drift corrections through use of a magnetometer) and ease of programming. Therefore the decision was taken to use the Invensense MPU9250 [22]. This IMU is available on different breakout boards, is cheap[1] and is widely used and well documented. The MPU9250 consists of an accelerometer, gyroscope and a magnetometer

[1]The IMU is available at TinyTronics for roughly €9,-: https://www.tinytronics.nl/shop/nl/sensoren/magnetisch-veld/mpu-9250-accelerometer-gyroscope-magnetometer-9dof-module-3.3v-5v?search=9250, 11-1-2019

(AK8963). Furthermore, the IMU has the ability to use a Digital Motion Processor (DMP)[2] which can be used to offload calculations to the IMU's onboard processor.

2-1-2 GNSS Receiver Selection

The second sensor required for the localisation solution is the GNSS sensor plus antenna. The U-Blox NEO M8P GNSS receiver was chosen because Accenda already had two of these receivers in stock. Furthermore, this sensor is one of the cheaper receivers capable of receiving raw GNSS data. This ability is important for future implementations of more advanced GNSS algorithms such as Single Frequency Precise Point Positioning (SF-PPP)[3].
The specific variant of the board used is made by the French company Drotek [23]. The combined cost of this board together with the Tallysman TW2412 [24] costs €264.82 which, for a raw capable GNSS module and antenna is acceptable.

2-1-3 Vehicle-Platform

Accenda provided the development platform used for this research. This vehicle-platform consists of a converted mobility scooter outfitted with an Arduino and a CAN-BUS shield developed by Accenda. The steering angle is measured using a potentiometer with a voltage sense line. The motor RPM is measured using an phototransistor(an Omron EE-SX1161-W11 transmissive photomicrosensor) and an encoder disc (see appendix A-2). The phototransistor measures the angular velocity of the motor shaft based on the duration at which light is or isn't allowed to pass through. Based on the transmission-ratio the average rear wheel velocity can thus be determined[4].

2-1-4 Micro-controllers and Processors

On the hardware side two different micro-controllers are used. The IMU is connected through an Inter-Integrated Circuit (I2C) bus with an ESP32 micro-controller. Because the required computational power was uncertain the ESP32 was chosen because of its high clockspeed dual-core processor (240 Mhz).
The vehicle-platform uses an Arduino-Uno [25] with a CAN-BUS shield. For the tasks this Micro-controller has to perform: Send/Receive CAN-BUS messages, read remote controller inputs, measure both the motor angular velocity as well as the steering angle and actuate the vehicle the Arduino is more than sufficient.

To collect and record all data and, in the future, fuse the data, a Raspberry PI 3B+ [26] was chosen. This choice was motivated by the experience of Accenda with the Raspberry, their development of a CAN-BUS shield for this platform and the low cost. Since the Raspberry

[2]Sadly, the DMP lacks documentation and only allows for the fusion of the Accelerometer and the Gyroscope making and therefore lacking the drift correction provided by the magnetometer

[3]A method that allows for high accuracy localisation through filtering out the most common GNSS disturbances such as: Multi-Path, Ionospheric & Tropospheric delays among others.

[4]A caveat of the employed system is that the software has to wait for a full section (either air or metal) to pass before a measurement update can be completed. This means that the minimum measurable velocity is roughly 600[RPM]

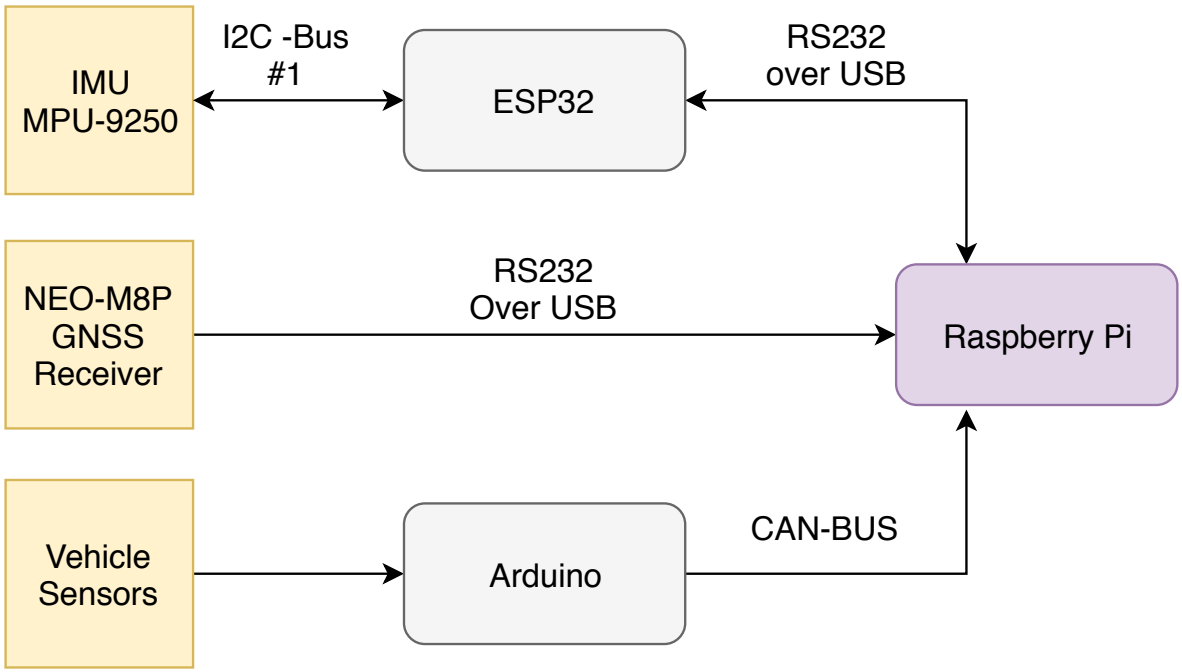


Figure 2-1: A schematic overview of the hardware system

PI 3B+ has four cores operating at 700 Mhz it should have enough computational power to run real-time sensor fusion as well as the ability to allow for multi-processing which is advantageous for parallelisation of computationally intensive tasks.

2-2 Software

In this section an overview of the software architecture is given. For the thesis software was developed for Python 3, Arduino (C++) as well as Matlab. Since the Matlab code is not strictly part of the vehicle-platform this software is not described here.

2-2-1 Raspberry PI

The code running on the Raspberry PI is based on Python 3.6. This code is Multi-Threaded and Multi processed. Each process has its own clearly separate function. The reason the code has been split into different processes is for performance purposes.

Processes and Threads

In this section the processes and threads that are spawned will be listed and briefly explained;

- **Process: Logger**
  This process is handles status messages it receives from the other processes and threads. These messages allow the user to identify the operating parameters of the software. These logs are also stored in a separate *.log* file.

- **Process: GNSS_Handlers**
  Receives GNSS messages from the receiver and packs & transmits them to the Data-Logger thread.
- **Process: CAN_Handler**
  This process sends/receives messages from the CAN-BUS and packs & transmits them to the Data-Logger.
- **Thread: Data-Logger**
  This thread receives all information from the sensors via their corresponding process or thread and stores them in *.csv* files per sensor. Therefore, when the system is fully operational three different *.csv* files are filled.
- **Thread: IMU_Handler**
  This thread decodes the information send over Serial-USB using the protocol described in section 2-3-2. This information is then packed and sent to the Data-Logger.

A visualisation of the process and threads structure can be seen in figure 2-2.

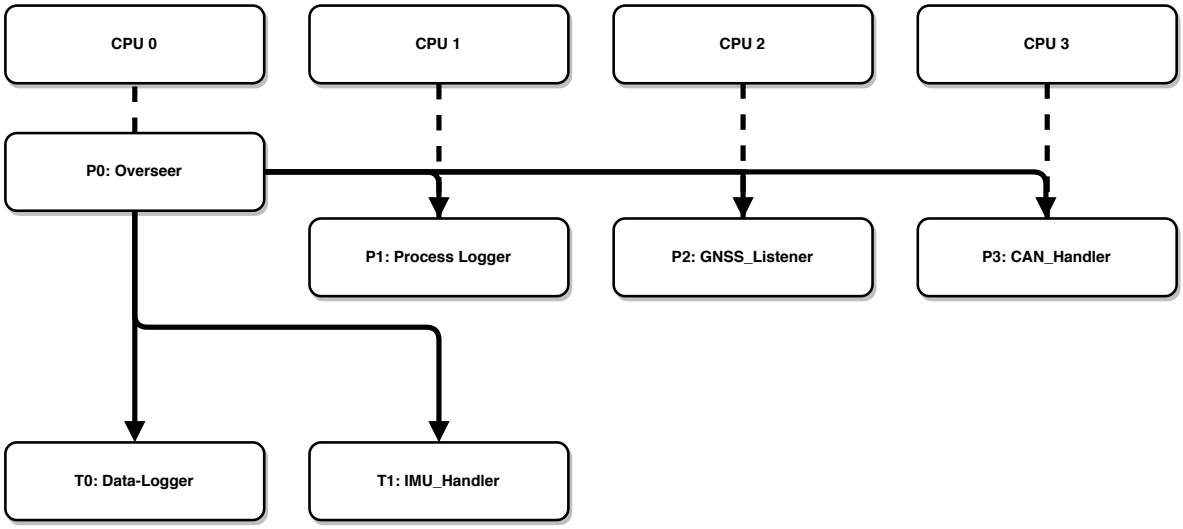


Figure 2-2: This figure displays the Process and Threads structure in the Python code. As can be seen the Overseer process (P0) initiates all processes and threads (denoted by T).

2-2-2 Arduino and ESP32

On the ESP a modified version of the SparkFun library for the MPU9250 is implemented [27], these modifications consists of, among other things, the addition of a magnetometer calibration loop (see appendix A-1) as well as the inclusion of a custom protocol for communication. On the vehicle-platform a modified version of Accenda's software is ran. Notable additions include CAN-BUS communication, battery reading as well as the inclusion of a state machine and the optimization of the sensor loop. For example, the new code only reads remote control inputs 10 times per second whereas the old code read the remote state every iteration loop wasting valuable computation time.

2-3 Communication

In this section the three different communication channels will be described. These are: a self-developed protocol for RS232(serial) communication over USB between the ESP32 and the Raspberry PI, CAN-BUS communication between the vehicle-platform and the Raspberry PI and RS232 communication over USB between the GNSS receiver and the Raspberry PI. The data-rates chosen for the CAN-BUS and IMU logging is roughly 20[Hz]. Since the platform is reaching speeds that are up to a max of $\approx 12Km/h$ a higher data-rate is not necessary. The GNSS update rate is only 1[Hz], this limitation is imposed by the receiver as it is incapable of updating at a higher rate.

2-3-1 CANBUS Messages

The vehicle-platform described in **??** uses CAN-BUS communication to send information to the Raspberry PI logger. As CAN-BUS is a well documented communication bus no in depth description will be presented here. Instead, the interested reader is referred to the following sources: [28, 29]. Basically, the CAN-BUS packets consist of a header which is used to distinguish different payloads from one another and a 8-byte payload. For communication two different packages are implemented with the following headers:

- **Telemetry-Packet, 0x16**
- **Control-Packet, 0x15**

The control-packet (table 2-1) requires some explanation as to the chosen data-sizes. On the vehicle-platform itself, the received control-packet is read and both the steering value and the throttle value are then converted to a Pulse Width Modulation (PWM) value. On the vehicle-platform side the single byte value $dim(0, 255)$is mapped into the range of $dim(-255, 255)$. This mapping is necessary since the H-bridges that control both the steering servo-motor as well as the drive-motor use a separate PWM signal for left/right forward/reverse.

Table 2-1: These two tables represent the data being communicated through the CAN-BUS.

Telemetry-Packet	Type	Bytes
Steering Angle	Float32_t	4
Motor RPM	uInt16	2
State	byte	1
Battery Voltage	byte	1

Control-Packet	Type	Bytes
Set_State	byte	1
Steering Value	byte	1
Throttle	byte	1

2-3-2 RS232 Protocol

At first an attempt was made to directly connect the IMU to the Raspberry PI. However, since there was no publicly available library which would allow the use of the IMU's internal DMP it was decided to integrate the ESP32 into the communication line. However, as the ESP32 is not by default programmable using the Arduino Integrated Development Environment (IDE) an external library was required [30]. Unbeknownst to the author, this library was at the start

of the software implementation unable to sustain a stable I2C connection with the IMU. This resulted in some frustration during the development process. Furthermore, the Invensense's DMP implementation was poorly documented.
After studying the implementation of orientation filters on hardware vastly inferior to the ESP32 by [31] it was expected that the ESP32 would be able to run filters at update rates far exceeding system requirements [5]. It was therefore decided to implement an orientation filter on te ESP32 itself.

This automatically leads to the question on how the filtered data needs to be transferred between the ESP32 and the Raspberry PI and how to optimize the use of the RS232 communication bus. For this purpose a custom protocol was developed. This protocol is significantly more efficient than 'text' based data transfer. For example, encoding the variable '100' in pure text is equivalent to using a message of three bytes representing the characters 1, 0, 0 in 'UTF-8' or equivalent encoding. Whereas using a single byte to represent the number 100 directly saves 2 bytes of bandwidth. For floating point numbers this difference is even more pronounced as a 32-bit number requires only 4 bytes. With these four bytes a value between $-3.4E+38, +3.4E+38$ can be encoded with a decimal accuracy of about 7. See [32] which describes the standards for floating-Point arithmetic.

Message encapsulation

Firstly a packet is defined as a message encapsulated within a protocol, the message itself consists of sensor data and other data, see Figure 2-3. In the developed protocol, the protocol layer which encapsulates the message consists of three reserved hex characters. The encapsulation itself works as follows: When a new message is ready a byte array is iteratively filled with the data in this message. The first byte in the array is filled with in 'SOT' or Start Of Transmission byte, after which follows the packet type byte. For example 0x0C describes the M_Packet which contains all IMU data. This reserved byte was created to differentiate packet types from one another[6]. Table 2-3.

After the message type, the sensor data and the rest of the message is checked against the reserved hex values (SOT, EOT or DLE). If one of the bytes encodes one of the reserved hex values, a 'DLE' or Data Link Escape hex is inserted before that byte. This DLE lets the receiver ignore the first byte after the DLE, so if a reserved hex value is present it is ignored. If the complete message is processed an EOT hex is appended to the packet to denote the end of the packet and the packet is send to the serial out-buffer. The performance of the protocol on the ESP32's side indicate that sending messages costs less than 1 [ms] per message.

[5] This turned out to be roughly 200[Hz] and is directly tied to the maximum output rate of the magnetometer.
[6] Initially two different packets were sent using this protocol. The reserved byte allocation was kept for the modularity of the code.

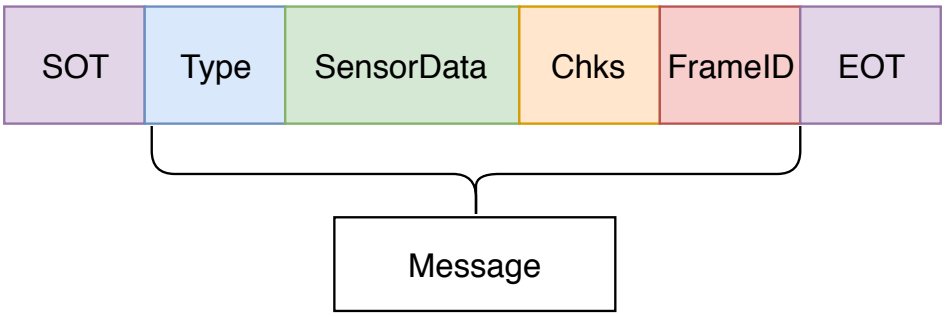


Figure 2-3: A visualisation of the created RS232 protocol.

Table 2-2: Reserved protocol HEX values

Packet	Type	Value
SOT	byte	0x01
EOT	byte	0x04
DLE	byte	0x10

Table 2-3: Message Structure

MAN_Packet	Type	Bytes	Value
Type	byte	1	0x12
AX	Float32_t	4	-
AY	Float32_t	4	-
AZ	Float32_t	4	-
GX	Float32_t	4	-
GY	Float32_t	4	-
GZ	Float32_t	4	-
MX	Float32_t	4	-
MY	Float32_t	4	-
MZ	Float32_t	4	-
Q0	Float32_t	4	-
Q1	Float32_t	4	-
Q2	Float32_t	4	-
Q3	Float32_t	4	-
Posix_t	Float32_t	4	-
Chks	byte	1	0-255
FrameID	byte	1	0-255
Total Size		59	

2-4 Communication-rate Evaluation for IMU and CAN-BUS

After programming the communication for all three sensor systems it is important to check if all rates and assumptions are as expected. That is, an IMU and CAN-BUS update rate of roughly $20[Hz]$.
The GNSS data-rate has not been taken into account since its data-rate is fixed.

As can be seen from figure 2-4 the data-rate of the IMU relatively stable, however there exist a notable amount of peaks. These can be attributed to the fact that the python-code running on the Raspberry PI periodically enters sleep mode. This depends on the requirements of other pieces of code needing to be executed on the same physical CPU-core. When this happens the Serial buffer will continue to fill with packages. However, when the IMU_Handler awakes it will timestamp these packages in buffer with near-identical time-stamps.

To mediate this, the ESP32's internal clock is synchronised with the Raspberry PI's internal

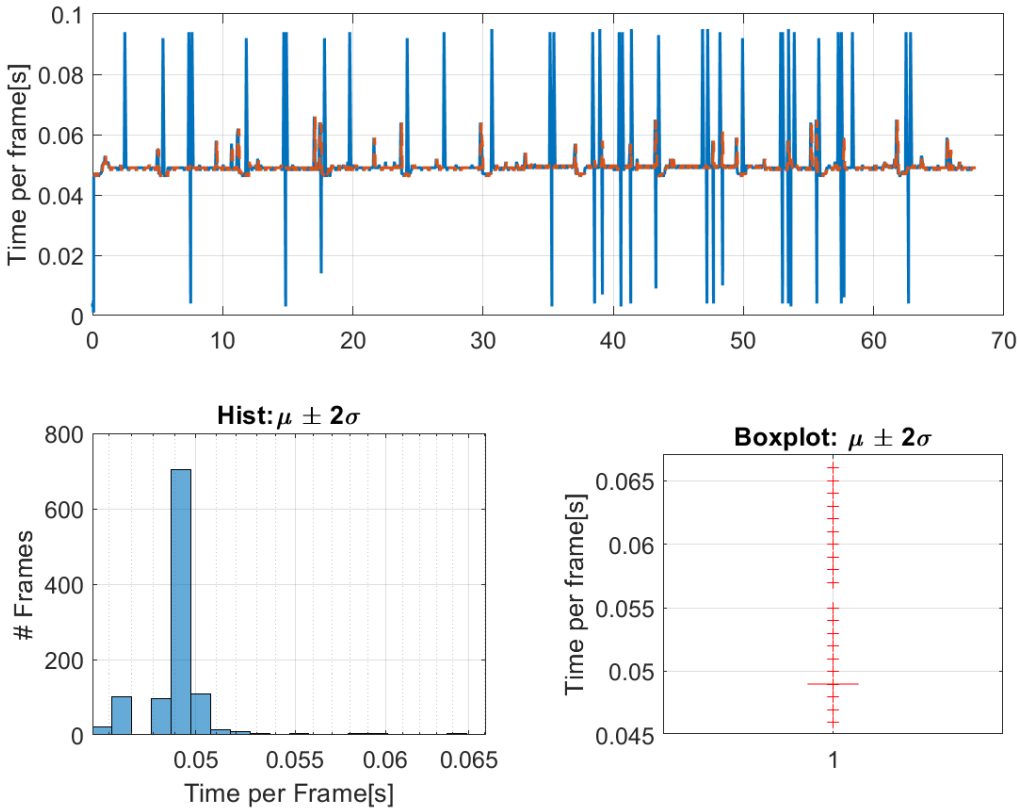


Figure 2-4: This figure displays the inter-frame time of the received IMU packages. Note that this inter-frame time is based on the Raspberry PI timestamping **Top**: Displays the time per frame per measurement instance in seconds. **Bottom-left**: Shows the histogram of the inter-frame times. **Bottom-right**:Displays the box-plot of inter-frame times.

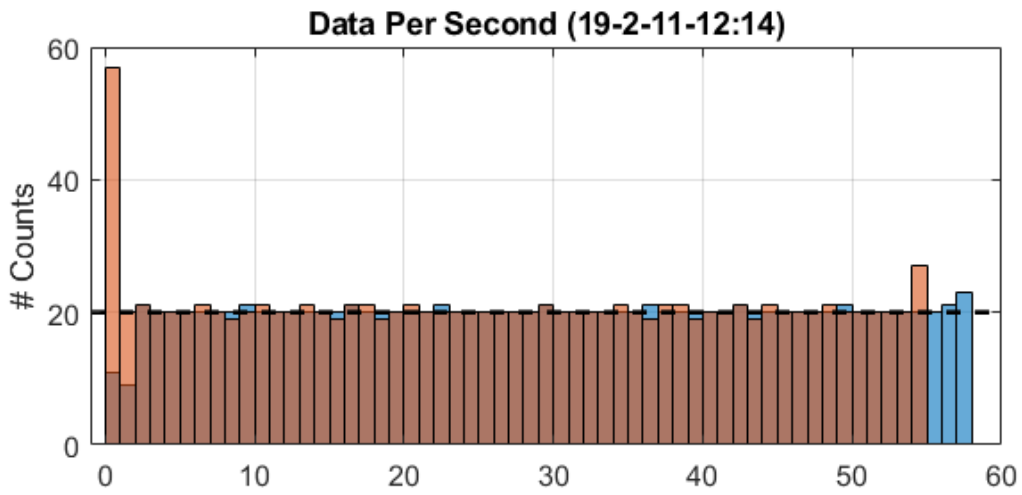


Figure 2-5: This figure displays the frames per second for both the ESP32-timestamp (blue) and the Python-timestamp(red), the x-axis displays the time in seconds.

clock. This allows the ESP32 to timestamp its IMU messages with its own clock.
In figure 2-5, which displays the amount of frames per second, it can clearly be seen that using the synchronised time has its advantages. Still, it is not possible to determine in [ms] which frame is created when.
Using the synchronised time however, it is possible to assume equally spaced samples over each measurement second allowing for millisecond time-stamping by assuming both a constant 20[Hz] data-rate as well as equally spaced samples[7].

The CAN-BUS data-rate, as can be seen in figure 2-6 is variable. This has to do with the hardware implementation of the velocity measurement. This measurement depend on measuring the time it takes a blade of an optical encoder to pass. This measurement is thus directly dependant on the motor [RPM] (which the encode encodes) and thus the velocity of the vehicle.
Furthermore, the implemented CAN-BUS chip (Microchip's MCP2515) does not support hardware-based frame-timing. This results in time-stamping being dependant on the 'enthusiasm' of the Process. If delays occur somewhere these can place a measurement at the wrong time in the chain which can make data-fusion difficult later-on.

[7]This approach is taken for the Pose-Estimation filter implemented in Matlab.

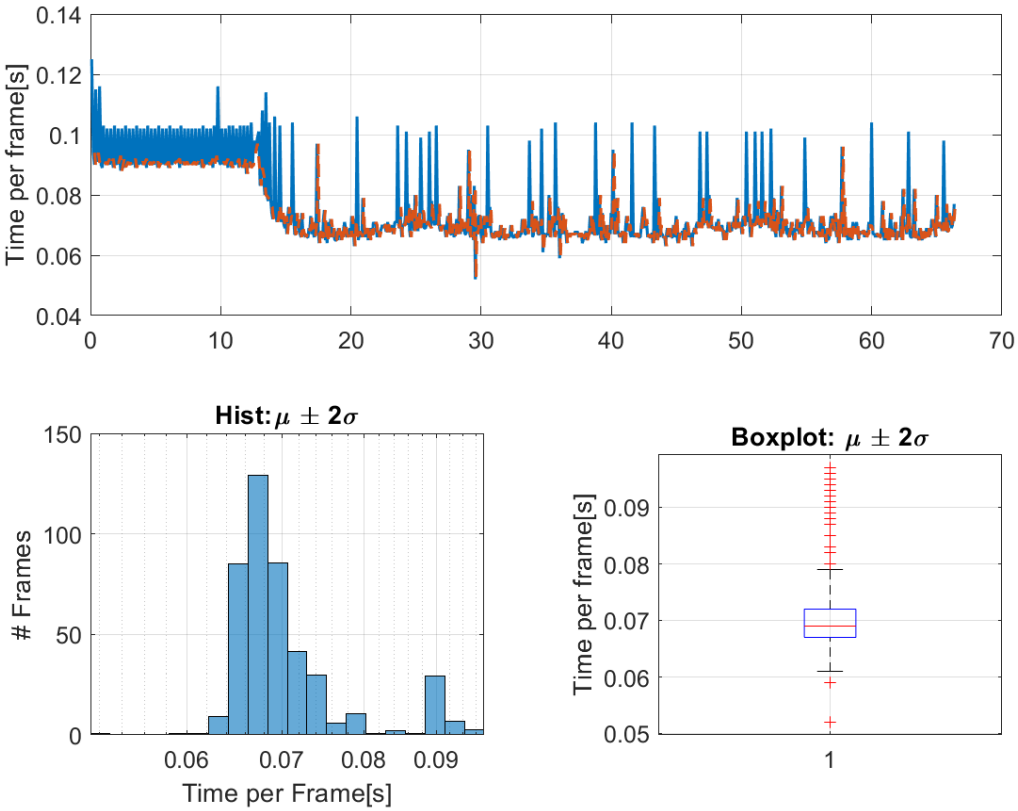


Figure 2-6: This figure displays the inter-frame time of the received Vehicle Sensors (VS) packages. Note that this inter-frame time is based on the Raspberry PI timestamping **Top**: Displays the time per frame per measurement instance in seconds. **Bottom-Left**: Shows the histogram of the inter-frame times. **bottom-right**:Displays the boxplot of inter-frame times

2-5 Boundary Conditions

The hardware chosen and the software developed in this chapter satisfies the conditions as set described by the first goal of the thesis section 1-2-1. The achieved data-rates for the IMU and the CAN-BUS turned out to be a stable 20[Hz] for the IMU and and an approximate $14[Hz]$ for the CAN-BUS.
Furthermore, the choice to implement the IMU pose estimation filter on the ESP32 has been justified by the fact that besides offloading computation tasks from the Raspbery PI to the ESP32 it moreover limits the required communication over the RS232 bus.

Chapter 3

Definitions

In this chapter concepts and mathematical definitions needed for the following chapters will be given. The first section, section 3-1, will concern the used mathematical concepts after which section 3-2 will describe the choice of mathematical conventions and parameter definitions used in the following chapters. Lastly, section 3-3 will give the vehicle-platform frame definitions and parameters used in later chapters.

3-1 Mathematical Concepts

To prepare the reader for the content of this chapter, this section is set-up to provide the reader with information concerning the different terms and definitions present in the following chapters.

Frame

A frame defines what, with respect to a certain 'defined' origin, can be expressed as up/down left/right front/back.

A frame for all intents and purposes has an orthonormal-basis in $\mathbb{R}^3$, also called Euclidean space.

$$\mathcal{F}_{x,y,z} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3] \qquad \in \mathbb{R}^{3\times 3} \tag{3-1}$$

$$\mathbf{v}_1 = (1,0,0)\,, \mathbf{v}_2 = (0,1,0)\,, \mathbf{v}_3 = (0,0,1) \tag{3-2}$$

For navigation purposes North, East, Down (NED) is often used.

Euler Angles

Euler angles are the most often used representation of frame rotation. They in essence represent an order of single-axis rotation sequences to describe the orientation of a frame

$\mathcal{F}_a$ with respect to frame $\mathcal{F}_b$. To perform rotations based on Euler angles one needs the three angles of rotation and a rotation order, for example using (ψ, φ, θ) (Yaw, Pitch, Roll) and the following rotation rotation order zyx[1]. These single-axes rotations are described by the following rotation matrices:

$$\mathbf{R}_z^{(\psi)} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}_y^{(\varphi)} = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix} \quad \mathbf{R}_x^{(\theta)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & \sin\theta \\ 0 & -sin\theta & \cos\theta \end{bmatrix}$$

A notable downside of using Euler Angles it the possibility of 'Gimabl-Lock' this occurs when a rotation of 90[°] occurs around one axis. This results in the loss of a degree of freedom.

DCM

A Direction Cosine Matrix (DCM) is a $\mathbb{R}^{3\times3}$ rotation matrix that is similar to the Euler rotation matrices. However, where Euler rotation matrices always denote rotations around one of the three principle axes of the inhabited frame, a DCM is more general. This means that a DCM can be used to describe a rotation around an arbitrary vector thus changing the complete basis of a frame using a single rotation matrix:

$$\mathcal{F}_2 = \mathbf{R}_\psi \mathcal{F}_1 \tag{3-3}$$

Quaternion

Quaternions, invented by Hamilton [33], are like Euler Angles and DCM's as that they can be used for frame rotations. A quaternion is a 4-tuple consisting of a scalar and of a vector part:

$$\mathbf{q} = (q_0, q_1, q_2, q_3) \qquad \in \mathbb{R}^4 \tag{3-4}$$
$$\mathbf{q} = (q_0, \vec{\mathbf{q}}) \tag{3-5}$$
$$\mathbf{q} = (q_0, \mathbf{i}q_1, \mathbf{j}q_2, \mathbf{k}q_3) \tag{3-6}$$

The vector part denoted here by $\vec{\mathbf{q}}$ is expressed in the standard orthonormal basis for $\mathbb{R}^3$. Since the reader may be unfamiliar with the concept of quaterions and they form an integral part of chapter 4 the basic rules for addition/substraction and multiplication will be laid out here.

Basic mathematical operations on quaternions such as equality and addition/subtraction that is for equality all individual components of the quaterions need to be equal:

$$\mathbf{q} = \mathbf{p} \implies (q_0, q_1, q_2, q_3) = (p_0, p_1, p_2, p_3) \tag{3-7}$$
$$\mathbf{q} + \mathbf{p} = (q_0 + p_0, q_1 + p_1, q_2 + p_2, q_3 + p_3) \tag{3-8}$$
$$\tag{3-9}$$

Multiplication of quaternions will be denoted by $\otimes$:

$$\mathbf{p} \otimes \mathbf{q} = p_0 q_0 - \vec{\mathbf{p}}\vec{\mathbf{q}} + p_0\vec{\mathbf{q}} + q_0\vec{\mathbf{p}} + \vec{\mathbf{p}} \times \vec{\mathbf{q}} \tag{3-10}$$

[1] This is referred to as the 'Aerospace-Sequence'

It is important to note that all quaternions used in this thesis will be unit quaternions:

$$|\mathbf{q}| = 1 \tag{3-11}$$

For more information about quaternions the book by Kuipers [34] is wholeheartedly recommended.

Pose

A pose ($\mathbf{p}$) is defined as a combination between an orientation $\mathbf{o}$ and position $\mathbf{x}$.

$$\begin{aligned} \mathbf{o} &= (\psi, \varphi, \theta) && \in\mathbb{R}^3 \\ \mathbf{x} &= (x, y, z) && \in\mathbb{R}^3 \\ \mathbf{p} &= \begin{bmatrix} \mathbf{x} & \mathbf{o} \end{bmatrix} && \in\mathbb{R}^6 \end{aligned}$$

Or in words, a pose defines the orientation and position of an object **a** with respect to an Orthonormal-frame **b** Note that in this example Euler angles are used to express orientation, this could of-course alternatively be a quaternion, DCM or other orientation representation.

3-2 Mathematical Conventions and Parameter Definitions

As most of the thesis deals with transformations and rotations between different frames it is important to know in which frame certain action or measurement takes place. For this purpose the notation in Madgwick [35] is followed. Thus ${}^{E}q$ denotes a quaternion reference expressed in frame E which is the earth reference frame in NED. The notation ${}^{E}_{S}q$ describes an orientation of frame E with respect to frame S. For quaternions the following holds:

$${}^{E}_{S}q^{*} = {}^{S}_{E}\,q \tag{3-12}$$

Matrices in the text will be expressed as bold capital letter $\mathbf{A}$, whereas vectors will be expressed as bold lower-case letters $\mathbf{v}$. Scalars will generally be expressed using non-bold, lower-case letters c. For an overview of the used symbols please look at the nomenclature in appendix C.

3-3 Vehicle-Platform Dimensions and Frames

The vehicle-platform itself as displayed in figure 3-1 uses three frame definitions:

$$\mathcal{F}_e = (North, \quad East, \quad Down) \tag{3-13}$$

$$\mathcal{F}_b = \mathbf{R}_z^{\beta}\mathcal{F}_e \tag{3-14}$$

$$\mathcal{F}_s = \mathbf{R}_z^{(\beta+\frac{\pi}{2})}\mathcal{F}_e \tag{3-15}$$

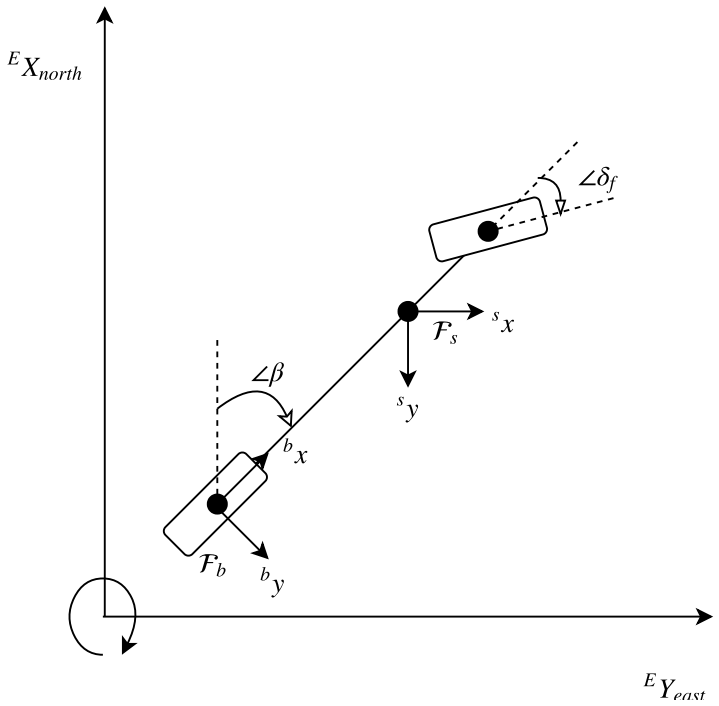


Figure 3-1: This figure displays the vehicle-platform's frames of reference. Note that F_b Denotes the body frame, F_s the sensor frame and the global frame is aligned with the NED frame. Note that the frame's orientation definition is left-hand positive.

Note that the body frame $\mathcal{F}_b$ is situated in the middle of the rear axle furthermore, the reason there is a $\frac{\pi}{2}$ rotation from $\mathcal{F}_b$ to $\mathcal{F}_s$ is due to the physical rotation of the sensor when mounted in the actual vehicle-platform.
For the filter as discussed in chapter 5 the following parameters need to be defined: $L = 0.9[m]$ as the distance between the front and rear axle. The distance $L_s = 0.64[m]$ defines the distance between the Inertial Measurement Unit (IMU) and the and the rear axle. The wheel radius $W_r = 0.13[m]$ denotes the radius of the wheels of the vehicle-platform. Lastly, $R_t = \frac{1}{20}$ denotes the transmission ratio between the motor and the rear wheels. The conversion from motor RPM to velocity[m/s] can thus be calculated using:

$$^{B}u_x = RPM \cdot R_t \cdot 2\pi W_r \tag{3-16}$$

Chapter 4

IMU Orientation Estimation

This chapter starts with a brief description of the chosen filter structure in section 4-1. After which the first part of this filter, the IMU orientation filter will be discussed in section 4-2. The chosen filter will then be used for experiments (see section 4-3). The chapter will end with a section 5-3 wherein the results will be discussed as well as conclusions will be drawn.

4-1 Filter Architecture

As a direct result of the boundary conditions laid out in the previous chapter (see section 2-5), a cascading filter structure where the raw Inertial Measurement Unit (IMU) data is pre-filtered and then communicated with the rest of the system is chosen. This structure has the added benefit that it is modular as well as capable of handling variable data-rates. This is inline with the requirements for the second thesis goal as described in section 1-2-2.

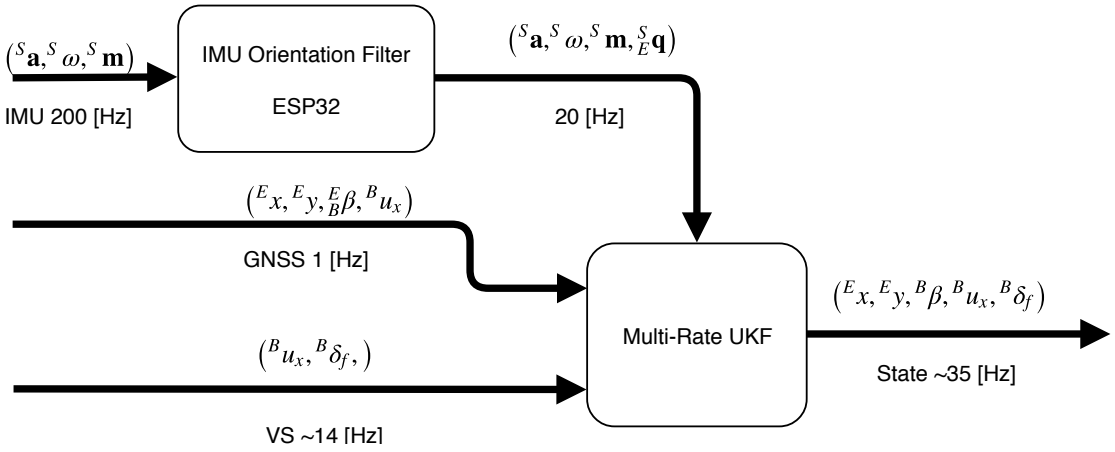


Figure 4-1: This figure displays the chosen filter structure with the parameter propagation through the filter as well as the expected update-rates.

4-2 Orientation filter for Motion Processing Unit (MPU)9250

In this section the way the MPU9250's accelerometer, gyroscope and magnetometer data is fused will be explained. Three different filters will be investigated and compared to a high samplerate filter as described by Madgwick [35].

The structure of this section is as follows: First in section 4-2-1 the to be fused frames will be defined, in section 4-2-2 the inner workings of the Madgwick complementary filter are given. In section 4-2-3 the Unscented Kalman Filter (UKF) will be explained toghether with the two resulting filter variants in consideration. After which section 4-3 will detail the experiment performed to evaluate the performance of the selected filters. Lastly, section 4-4 will give the result of the evaluation of the filters together with a conclusion.

4-2-1 Definition of Frames

For the fusion of, in essence, three different sensors on a single IMU, it is important to define a way to fuse the sensor data. When performing sensor fusion the most often held definition is that there exist a process and measurement model:

$$\hat{\mathbf{x}} = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u} + \mathbf{w}_k \tag{4-1}$$

$$\mathbf{y}_k = \mathbf{H}(x_k) + \mathbf{v}_k \tag{4-2}$$

The chosen IMU, the MPU9250 provides raw accelerometer ${}^{S}\mathbf{a} = (a_x, a_y, a_z)$, gyroscope ${}^{S}\omega = (\omega_x, \omega_y, \omega_z)$ and magnetometer ${}^{S}\mathbf{m} = (m_x, m_y, m_z)$ measurements.
For IMU orientation estimation the sensor is often modelled as a point in space with an orientation with respect to a defined global frame. Often two sensor frames, based on the grouping of the different sensors in the MPU are defined [36–42]. In the following sections these frames will be defined and explained in detail.

Frame I: Relative Gyroscope Frame

The first frame is based solely on the previous state and the measured gyroscope angular velocity. This is then simply integrated over time

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + \dot{\mathbf{x}} \cdot \Delta t \tag{4-3}$$

This integration thus expresses a changing orientation using for example quaternions or Euler angles.

Frame II: Absolute Frame Based on Accelerometer and Magnetometer

The second frame, which measures the orientation in an absolute sense. Absolute here means that the orientation at each time k is solely dependent on the measured values of that time step.
The sensors used to define this frame are the accelerometer and the magnetometer. Ideally both are used since using only the accelerometer or the compass result in having free parameters in the second frame definition, making the second frame no longer constrained.
To generate this second frame two assumptions are made:

Assumption 1. *The accelerometer predominantly measures the gravity. That is:*

$$\mathbf{g} \approx \mathbf{a} \tag{4-4}$$

Assumption 2. *The magnetometer predominantly measures the direction of the earth's magnetic field.*

As long as both assumptions hold it is possible to create a fully defined orientation with respect to the earth frame $\mathcal{F}_e$. Thus the problem becomes how to fuse the measurements of these two sensors together. This problem, often referred to as the *Least Squares Estimate of Satellite Attitude* or *Wahba* problem [43] pertains to the difficulty of fusing two or more sets observation vectors to obtain a Direction Cosine Matrix (DCM), quaternion or Euler angles representation describing the orientation of the object, to which the observation vectors pertain, with respect to a global frame. From the literature three different algorithms have been evaluated that solve this problem:

- **Quaternion Estimator (QUEST)** [40, 44–46]
  This algorithm requires three or more vector measurements to determine the orientation of the body. QUEST is computationally fast however, it cannot perform $\geq 180\,\text{deg}$ rotations. Another challenge is that, contrary to Three Dimensional Attitude Estimation (TRIAD) at least three different vector groups are required therefore making it more difficult to implement[1] and is therefore omitted in the evaluation.

- **TRIAD** [42, 47]
  This algorithm requires at least two vectors to determine body orientation. This algorithm has the advantage that it does not require matrix inverse calculations and is thus relatively fast and stable. For the implementation used see [42]

- **Gradient descent** [35]
  Gradient descent is an effective method for determining the optimal orientation based on two vector measurements. Though computationally complex it provides suitable answers. For the implementation chosen the stepsize update is determined by the Barzilai-Borwein algorithm [48]. If faster computation is required, the iteration count can be set to 1 (as is default by most Madgwick filter implementations). This has the disadvantage that the initial stepsize has to be tuned. This initial step-size is dependent on both the update rate of the algorithm and sensor as well as the movement of the body.

4-2-2 Filter 1: Madgwick

For the first filter we follow Madgwick's filter description. For the readers understanding, the math of the filter is included in the text as it has been reformulated from the source material.

[1] Using only two vector groups effectively reduces the QUEST algorithm into a TRIAD algorithm [46]

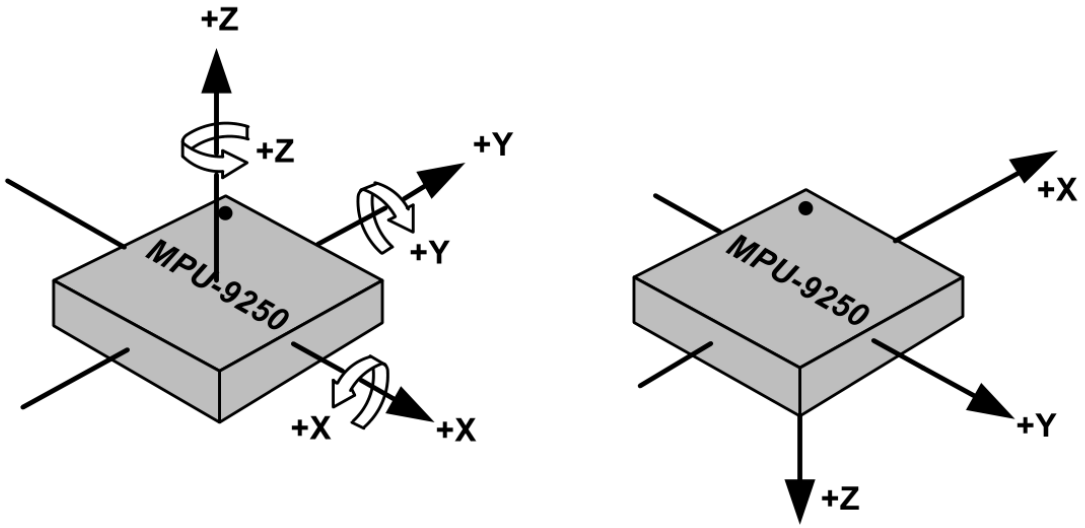


Figure 4-2: Sensor orientation axes. Left, Accelerometer and Gyroscope. Right, Magnetometer Source: [22]

Frame 1: Gyroscope

As the gyroscope measures an angular rate at a sample instance, it cannot be used for an absolute determination of the sensors orientation. It is however possible to determine the change of frame orientation based on the gyroscope's measurement. This can be done using the following equations (adapted from Madgwick):

$$^{S}\boldsymbol{\omega} = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix} \tag{4-5}$$

$$^{S}_{E}\dot{\mathbf{q}}_{\omega,t} = \frac{1}{2}\,^{S}_{E}\mathbf{q}_{est,t-1} \otimes {}^{S}\boldsymbol{\omega} \tag{4-6}$$

$$^{S}_{E}\mathbf{q}_{\omega,t} = {}^{S}_{E}\mathbf{q}_{est,t-1} + {}^{S}_{E}\dot{\mathbf{q}}_{\omega,t}\Delta t \tag{4-7}$$

Frame 2: Accelerometer + magnetometer

Accelerometer

$$^{E}\mathbf{g} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{T} \tag{4-8}$$

$$^{S}\mathbf{a} = \begin{bmatrix} 0 & a_x & a_y & a_z \end{bmatrix}^{T} \tag{4-9}$$

$$\mathbf{F}_g(^{S}_{E}\mathbf{q}, {}^{S}\mathbf{a}) = {}^{S}_{E}Q^{T} \cdot {}^{E}\mathbf{g} - {}^{S}\mathbf{a} = \begin{bmatrix} 0 \\ \boldsymbol{f}_g \end{bmatrix} \tag{4-10}$$

$$\mathbf{J}_g(^{S}_{E}\mathbf{q}) = \frac{\partial \boldsymbol{f}_g}{\partial\, ^{S}_{E}\mathbf{q}} \qquad \mathbf{J}_g \in \mathbb{R}^{3\times 4} \tag{4-11}$$

The notation of equation (4-10) is chosen to clearly define $\mathbf{f}_g$.

Magnetometer
The measurements from the magnetometer are corrected using an intermediate step:

$$^{S}\mathbf{m} = \begin{bmatrix} 0 & m_x & m_y & m_z \end{bmatrix} \tag{4-12}$$

$$^{E}\mathbf{h} = \begin{bmatrix} 0 & h_x & h_y & h_z \end{bmatrix} = {}^{S}_{E}\hat{\mathbf{q}} \otimes {}^{S}\mathbf{m_t} \otimes {}^{S}_{E}\hat{\mathbf{q}}^{*} \tag{4-13}$$

$$^{E}\mathbf{b}_t = \begin{bmatrix} 0 & \sqrt{h_x^2 + h_y^2} & h_z \end{bmatrix} \tag{4-14}$$

$$\mathbf{F}_b({}^{S}_{E}\mathbf{q}, {}^{S}\mathbf{b}_t) = {}^{S}_{E}Q^T \cdot {}^{E}\mathbf{b}_t - {}^{S}\mathbf{m} = \begin{bmatrix} 0 \\ \boldsymbol{f}_b \end{bmatrix} \tag{4-15}$$

$$\mathbf{J}_b({}^{S}_{E}\mathbf{q}, {}^{E}_{\mathbf{b}}) = \frac{\partial \boldsymbol{f}_b}{\partial {}^{S}_{E}\mathbf{q}} \qquad \mathbf{J}_b \in \mathbb{R}^{3\times 4} \tag{4-16}$$

With the accelerometer providing the vector to the earth's gravitational center and the magnetometer providing the direction to the earth's magnetic north, Frame 1 can be fully defined based on sensor measurements. To obtain the correct quaternion representation for this frame based on the sensor data, gradient descent is employed:

$${}^{S}_{E}\mathbf{q}_{\nabla,t} = {}^{S}_{E}\hat{\mathbf{q}}_{est,t-1} - \mu_t \frac{\nabla \boldsymbol{f}}{\|\nabla \boldsymbol{f}\|} \tag{4-17}$$

For the gradient descent the gradient function $\nabla \boldsymbol{f}$ is required. This function is defined as:

$$\nabla \boldsymbol{f}_{g,b} = \boldsymbol{J}^T_{g,b}({}^{S}_{E}\mathbf{q}_{est,t-1}, {}^{E}\mathbf{b}) \boldsymbol{f}_{g,b}({}^{S}_{E}\mathbf{q}_{est,t-1}, {}^{S}\mathbf{a}_t) \qquad \nabla \boldsymbol{f}_{g,b} \in \mathbb{R}^{4\times 1} \tag{4-18}$$

$$\boldsymbol{f}_{g,b}({}^{S}_{E}\mathbf{q}, {}^{S}\mathbf{a}, {}^{E}\mathbf{b}, {}^{S}\mathbf{m}) = \begin{bmatrix} \boldsymbol{f}_g({}^{S}_{E}\mathbf{q}, {}^{S}\mathbf{a}) \\ \boldsymbol{f}_b({}^{S}_{E}\mathbf{q}, {}^{S}\mathbf{b}_t, {}^{S}\mathbf{m}) \end{bmatrix} \qquad \boldsymbol{f}_{g,b} \in \mathbb{R}^{6\times 1} \tag{4-19}$$

$$\mathbf{J}_{g,b} = \begin{bmatrix} \mathbf{J}_g({}^{S}_{E}\mathbf{q}) \\ \mathbf{J}_b({}^{S}_{E}\mathbf{q}, {}^{E}\mathbf{b}) \end{bmatrix} \qquad \mathbf{J}_{g,b} \in \mathbb{R}^{6\times 4} \tag{4-20}$$

4-2-3 Filter 2: Unscented Kalman

As the representation of orientation is non-linear a 'Classic' Kalman Filter (CKF) [4] isn't suited for state estimation. For non-linear systems often the Extended Kalman Filter (EKF) is employed. Alternatively, the UKF proposed by Julier and Uhlman [12, 14] can be used for the estimation of non-linear systems.

There are some notable differences between EKF and UKF. Where the EKF uses linearisation to deal with non-linearity, the UKF generates a set of state vectors (sigma-points) based on variance matrix which is then propagated through the model. This procedure prevents local sampling and allows the UKF to perform at least equal and in most cases better than the EKF algorithm.

As mentioned in section 3-1 this research uses unit quaternions for relative orientation representation. This choice has some downsides where Kalman filters are concerned.

Remark. *As pointed out by [49], a normal quaternion consists of four independent variables, a unit quaternion however reduces the degrees of freedom by one due to its normality constraint.*

$$\mathbf{q} = \begin{bmatrix} q_1 & q_2 i & q_3 j & q_4 k \end{bmatrix}$$
$$|\mathbf{q}| = 1$$

This results in all quaternion parameters being loosely connected and therefore affecting the variance-covariance matrices. This in turn makes the noises of the individual quaternion vector and scalar non-normal which is in violation of the basic Kalman filter assumption of normally distributed noise over the measured states.[2] *Therefore, the quality of the fusion result is impacted.*

Applied Unscented Kalman Filter

For the implementation of the UKF the approach in table 7.3 of Haykin [50] is followed (this approach is similar to the one described by Wan et al. [51]). Since this report contains an implementation based on quaternions. The full algorithm will be included in this section.

Initial state

$$\hat{\mathbf{x}}_0 = \mathbf{q}_0 = \begin{bmatrix} 1, 0, 0, 0 \end{bmatrix}^T \qquad \in \mathbb{R}^{4\times 1} \tag{4-21}$$
$$\mathbf{P}_0 = \mathbf{0}_{4\times 4} \qquad \in \mathbb{R}^{4\times 4} \tag{4-22}$$

According to UKF theory the amount of sigma points generated is equal to the size of the state vector times two plus one thus;

$$L = dim(x) \tag{4-23}$$
$$\mathcal{X}_{k-1} = \begin{bmatrix} \hat{\mathbf{x}}_{k-1} & \hat{\mathbf{x}}_{k-1} \pm \gamma\sqrt{\mathbf{P}_{k-1}} \end{bmatrix} \qquad \in \mathbb{R}^{4\times(2L+1)} \tag{4-24}$$

[2] Quantifying the effect of this coupling when using quaternion based sensor-fusion compared to for example is a active field of research.

Note that in equation (4-24) the square root of the Variance Covariance Matrix (VCM) $\mathbf{P}$ is calculated using the Cholesky-Decomposition [52]. Furthermore, to generate the sigma-points the columns of the VCM are added to the state estimate $\hat{\mathbf{x}}_{k-1}$

Update through 'system' model using the gyroscope data:

$$\hat{\mathbf{x}}_{i|k} = \left(\mathbf{I}_{4\times4} + \frac{1}{2}\Omega_k \Delta t\right)\hat{\mathbf{x}}_{k-1} \tag{4-25}$$

$$\Omega_k = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega)z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{4-26}$$

$$\mathcal{X}^*_{k|k-1} = \mathbf{F}\left(\mathcal{X}_{k-1}, \mathbf{u}_{k-1}\right) \tag{4-27}$$

$$\hat{\mathbf{x}}^-_k = \sum_{i=0}^{2L} W_i^m \mathcal{X}^*_{i,k|k-1} \tag{4-28}$$

$$\mathbf{P}^-_k = \sum_{i=0}^{2L} W_i^c \left(\mathcal{X}^*_{k|k-1} - \hat{\mathbf{x}}^-_k\right)\left(\mathcal{X}^*_{k|k-1} - \hat{\mathbf{x}}^-_k\right)^T + \mathbf{R}^v \tag{4-29}$$

$$\mathbf{R}^v = \hat{\mathbf{x}}^-_k \otimes \begin{bmatrix} 0 & \epsilon_g \end{bmatrix} \tag{4-30}$$

Now the literature suggests one of the following two approaches to account for the process noise $\mathbf{R}^n$

- Augment the sigma-points
  This means append the sigma-points from equation (4-24) with extra sigma-points that take the added noise matrix into account. This would mean that the size of the sigma-points would become:

$$\mathcal{X}_{k|k-1} = \begin{bmatrix} \mathcal{X}^*_{k|k-1} & \mathcal{X}^*_{0|k-1} \pm \gamma\sqrt{\mathbf{R}^v} \end{bmatrix} \qquad \in \mathbb{R}^{4\times4L+1} \tag{4-31}$$

- Redraw the sigma-points
  Redrawing the sigma-points would include the effect of the added VCM in the new sigma-points:

$$\mathcal{X}_{k|k-1} = \begin{bmatrix} \hat{\mathbf{x}}^-_k & \hat{\mathbf{x}}^-_k \pm \gamma\sqrt{\mathbf{P}^-_k} \end{bmatrix} \qquad \in \mathbb{R}^{4\times2L+1} \tag{4-32}$$

The second option of redrawing the sigmapoints will be chosen in the implementation as that will reduce the computational load significantly because less states are used in the filter (With the caveat that this "discards any odd-moments information captured by the original propagated sigma points"(Haykin)).

Next the measurement update has to be computed:

$$\mathcal{Y}_{k|k-1} = \mathbf{H}\left(\mathcal{X}_{k|k-1}\right) \qquad \in \mathbb{R}^{4\times2L+1} \tag{4-33}$$

$$\tag{4-34}$$

where we use the measurement update as described in equation (4-17).

$$\mathcal{Y}_{k|k-1} = \mathcal{X}_{k|k-1} - \mu \frac{\nabla \boldsymbol{f}}{\|\boldsymbol{f}\|} \tag{4-35}$$

Then the measurement estimate is calculated using:

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^m \mathcal{Y}_{i,k|k-1} \tag{4-36}$$

$$\mathbf{P}_{\hat{\mathbf{y}}_\mathbf{k}\hat{\mathbf{y}}_\mathbf{k}} = \sum_{i=0}^{2L} W_i^c \left(\mathcal{Y}_{i|k-1} - \hat{\mathbf{y}}_k^-\right)\left(\mathcal{Y}_{i|k-1} - \hat{\mathbf{y}}_k^-\right)^T + \mathbf{R}^n \qquad \in \mathbb{R}^{4\times 4} \tag{4-37}$$

$$\mathbf{P}_{\hat{\mathbf{x}}_\mathbf{k}\hat{\mathbf{y}}_\mathbf{k}} = \sum_{i=0}^{2L} W_i^c \left(\mathcal{X}_{i|k-1} - \hat{\mathbf{x}}_k^-\right)\left(\mathcal{Y}_{i|k-1} - \hat{\mathbf{y}}_k^-\right)^T \qquad \in \mathbb{R}^{4\times 4} \tag{4-38}$$

$$\mathcal{K}_k = \mathbf{P}_{\hat{\mathbf{x}}_\mathbf{k}\hat{\mathbf{y}}_\mathbf{k}} {\mathbf{P}_{\hat{\mathbf{y}}_\mathbf{k}\hat{\mathbf{y}}_\mathbf{k}}}^{-1} \qquad \in \mathbb{R}^{4\times 4} \tag{4-39}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}_k\left(\mathbf{y}_k - \hat{\mathbf{y}}_k^-\right) \tag{4-40}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathcal{K}_k \mathbf{P}_{\hat{\mathbf{y}}_\mathbf{k}\hat{\mathbf{y}}_\mathbf{k}} \mathcal{K}_k^T \tag{4-41}$$

The weight factors denoted by W can be calculated as follows:

$$W_0^m = \frac{\lambda}{L+\lambda} \tag{4-42}$$

$$W_0^c = \frac{\lambda}{L+\lambda} + 1 - \alpha^2 + \beta \tag{4-43}$$

$$W_i^m = W_i^c = \frac{1}{2\left(L+\lambda\right)} \qquad i = 1, ..., 2L \tag{4-44}$$

$$\lambda = \alpha^2\left(L+\kappa\right) - L \tag{4-45}$$

$$\gamma = \sqrt{L+\lambda} \tag{4-46}$$

In general the values assigned to α, β and κ are chosen dependent on the prior knowledge of the distribution. This means that for a Gaussian distribution the following values are chosen:

$$1 \geq \alpha \geq 10^{-4} \tag{4-47}$$

$$\beta = 2 \qquad \textit{If Gaussian} \tag{4-48}$$

$$\kappa = 3 - L \tag{4-49}$$

According to [50] there exists no procedure for determining these parameters, Haykin et al. furthermore states that in general the choice of parameters doesn't matter for the stability of the solution, although the speed of convergence may suffer if they are chosen incorrectly. Julier, one of the principle inventors of UKF states that individual weighing factors do not necessarilly have to sum to unity [13] a belief that is held strongly by some online resources.

4-2-4 Filter 3: TRIAD

Filter 3 is the same as filter two, only the measurement model is changed. The measurement model now uses TRIAD for the fusion of the accelerometer and the magnetometer.

The TRIAD algorithm, developed by [45] uses two observation vectors ($\mathbf{w}_1, \mathbf{w}_2$) to determine the orientation of an object in space. For this to work two reference vectors have to be provided ($\mathbf{v}_1, \mathbf{v}_2$). These vectors define the world orientation frame to which the algorithm provides the objects relative frame. This is relative orientation is represented by a DCM, which will have to be converted to a quaternion.

The TRIAD algorithm as described in [42]:

$$\mathbf{A} = \mathbf{M}_o \mathbf{M}_r^T \qquad \in SO(3) \tag{4-50}$$

$$\mathbf{M}_o = \begin{bmatrix} \mathbf{o}_1 & \mathbf{o}_2 & \mathbf{o}_3 \end{bmatrix} \tag{4-51}$$

$$\mathbf{M}_r = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix} \tag{4-52}$$

$$\mathbf{o}_1 = \mathbf{w_1} \tag{4-53}$$

$$\mathbf{o}_2 = \frac{\mathbf{w_1} \times \mathbf{w_2}}{|\mathbf{w_1} \times \mathbf{w_2}|} \tag{4-54}$$

$$\mathbf{o}_3 = \frac{\mathbf{w_1} \times (\mathbf{w_1} \times \mathbf{w_2})}{|\mathbf{w_1} \times (\mathbf{w_1} \times \mathbf{w_2})\,|} \tag{4-55}$$

$$\mathbf{r}_1 = \mathbf{v_1} \tag{4-56}$$

$$\mathbf{r}_2 = \frac{\mathbf{v_1} \times \mathbf{v_2}}{|\mathbf{v_1} \times \mathbf{v_2}|} \tag{4-57}$$

$$\mathbf{r}_3 = \frac{\mathbf{v_1} \times (\mathbf{v_1} \times \mathbf{v_2})}{|\mathbf{v_1} \times (\mathbf{v_1} \times \mathbf{v_2})\,|} \tag{4-58}$$

4-3 Experiment

For the experiment the THORLABS PRO01/M z-axis rotation platform was used.
For driving it is most important to have a stable yaw angle rotation since for the most part the vehicles plane of motion will be perpendicular to the global z-axis. Which means that its orientation accuracy is directly dependant on the accuracy of the yaw angle.
The full set-up can be seen in figure section 4-3.

Using this setup, it is possible to make a coarse quantitative analysis of the selected filters for some basic scenarios. As the goal of this research is to see which of the selected filters performs 'best' running on a lower update frequency of 20[Hz] with respect to the 'Gold Standard' Madgwick filter running at 200[Hz]. For the experiment three different datasets have been collected:

- Test 1: YAW01
  This dataset has been created by rotating the sensor $\pm 90°$.
- Test 2: YAW02
  For this dataset z axis rotations of $\pm 20°$ have been used.
- Test 3: YAW03 For this dataset z axis rotations of $\pm 5°$ have been used.

Of the collected datasets, dateset 2: YAW02 will be discussed in more detail, the rest of the results can be found in appendix B-2.

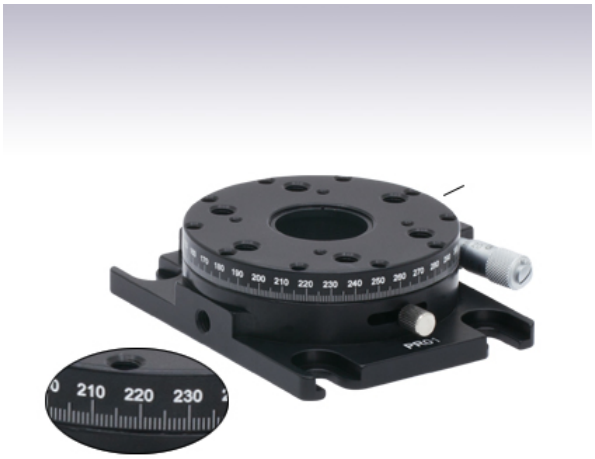


Figure 4-3: The THORLABS PRO01/M z-axis rotation platform.
source: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=2464

4-4 Results & Conclusion

When looking at figure 4-4 and figure 4-5 the raw-data shows the periodicity one can expect from repeated rotations around the z-axis. As expected a periodic saw-tooth like wave can be observed. The varying amplitude of this wave in the gyroscope subplot is due tot the manual nature of the excitation. It is interesting to note that the roll and pitch angle, as can be seen in the filtered data are, contrary to expectation, not zero. This can be explained by looking at the accelerometer values. Since the $a_{x,y}$ measurements are non-zero the roll and pitch angle are also non-zero. This can be caused by the sensor's x and y axis not being fully perpendicular to the direction of the earths gravity.
Nevertheless, as the vehicle in question is assumed to be moving on a 2d-plane, minor errors in roll and pitch should not result in problems.
One final note, the reason the 'Gold-Standard' Madgwick filter starts at a different point has to do with the implementation on the ESP32. During these tests the quaternion was initiated as: $\mathbf{q}_0 = (1, 0, 0, 0)$. Whereas the Matlab implementation of the filters start with $\mathbf{q}_0 = (0, 0, 0, 1)$.

Of the implemented filters the Madgwick filter, together with the TRIAD based Unscented Kalman filter perform the best. The other results as shown in appendix B-2 confirm that, of the UKF candidates the TRIAD implementation is the most promising one.
In conclusion, it has become clear that using the Madgwick filter for orientation estimation is reasonable. The only suitable tested alternative would be the the UKF using TRIAD as a measurement model. However, since UKF and TRIAD require more extensive and computationally heavy mathematical operations such as matrix inversion, Cholesky decomposition and DCM to quaternion conversions, the filter update rate of 200[Hz] is likely to suffer[3].

[3]Note that this is expected, not proven, due to time constraints and the relative difficulty to program matrix operations using the Arduino the choice has been made to continue the project without implementing the UKF on the ESP32.

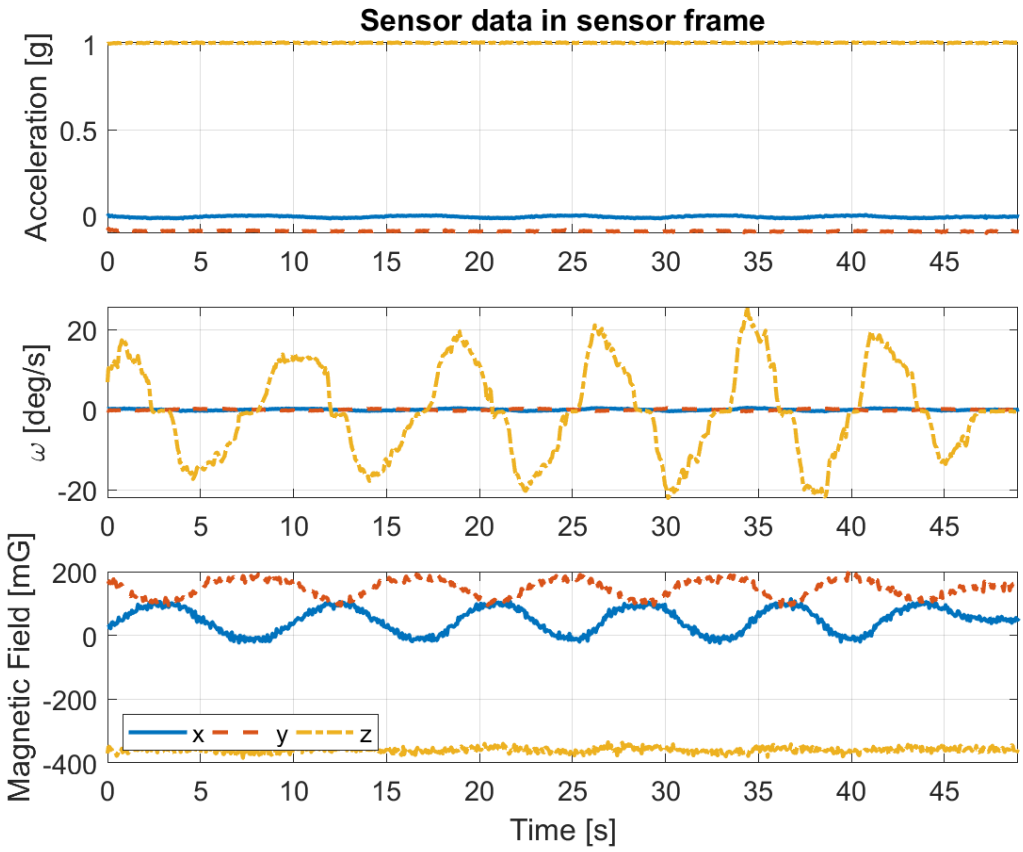


Figure 4-4: This figure displays the raw data obtained from the accelerometer, gyroscope and magnetometer.

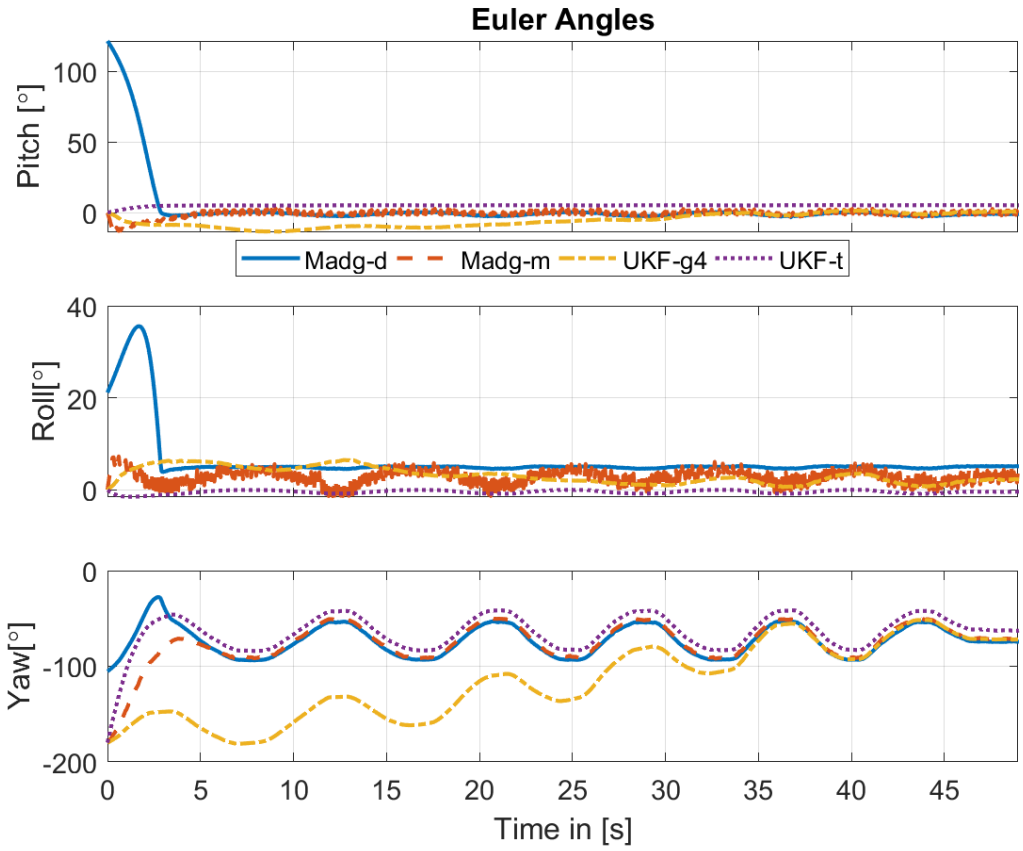


Figure 4-5: This figure displays the filtered data after using the 'gold standard' high-rate embedded Madgwick filter, the low rate Madgwick filter as well as an Unscented Kalman filter using either a gradient descent based measurement model or a TRIAD based measurement model.

Chapter 5

Multi-Rate UKF Pose estimation filter

In this chapter the Multi-Rate pose filter is discussed. In section 5-1 the structure of the Unscented Kalman Filter (UKF) will be given. After which the second section, section 5-2 will evaluate the chosen filter structure by means of simulations. Lastly, section 5-3 will contain conclusions drawn based on the simulations.

5-1 UKF structure

To account for the different data-rates of the sensors, the data-rate of the Inertial Measurement Unit (IMU), Vehicle Sensors (VS) and Global Navigation Satellite System (GNSS) are, $20[Hz]$, $\approx 14[Hz]$[1], $1[Hz]$ respectively, a Multi-Rate UKF implementation is chosen to fuse the sensor measurements together. The amount of literature where Multi-rate Unscented Kalman filtering is concerned is somewhat limited. However, a working example can be found in Lapouge et al. and Armesto et al. [17, 53] (Although it must be noted that Lapouge's implementation is directly based on Armesto's).

The chosen process model is a non-linear kinematic bicycle model operating in $\mathbb{R}^2$. This means that zero lateral or longitudinal wheel slip is assumed. Furthermore, the process model assumes constant velocity ${}^{B}u_k$ in the positive body x direction and and front wheel steering angle ${}^{B}\delta$.
A justification for the simplicity of the model used is the lack of sensors on the vehicle platform. For proper wheel slip estimates it is useful to have angular velocity measurements

[1] This rate is strongly dependent on the speed the vehicle travels.

of, at least, the two front wheels.

$$\mathbf{x}_k = \left({}^{E}x_k, {}^{E}y_k, {}^{E}_{B}\beta_k, {}^{B}u_k, {}^{B}\delta_f\right)^T \tag{5-1}$$

$$\mathbf{x_k} = \mathbf{f}\left(\mathbf{x}_{k-1}\right) + \mathbf{w}_{k-1} \tag{5-2}$$

$$= \begin{bmatrix} {}^{E}x_{k-1} + \Delta t \cdot {}^{B}u_{k-1} \cdot \cos({}^{E}_{B}\beta_{k-1}) \\ {}^{E}y_{k-1} + \Delta t \cdot {}^{B}u_{k-1} \cdot \sin({}^{E}_{B}\beta_{k-1}) \\ {}^{E}_{B}\beta_{k-1} + \Delta t \cdot \frac{{}^{B}u_{k-1}}{L} \cdot \tan({}^{B}\delta_{k-1}) \\ {}^{B}u_{x,k-1} \\ {}^{B}\delta_{f,k-1} \end{bmatrix} + \mathbf{w}_{k-1} \tag{5-3}$$

The measurement vectors of the different sensors are expressed as:

$$\mathbf{y}_{IMU} = \left({}^{E}_{S}\beta\right) \tag{5-4}$$

$$\mathbf{y}_{VS} = \left({}^{B}u, {}^{B}\delta_f\right) \tag{5-5}$$

$$\mathbf{y}_{GNSS} = \left({}^{E}x, {}^{E}y, {}^{E}_{B_*}\beta, {}^{B_*}u\right) \tag{5-6}$$

Now two assumptions are made concerning different body angle measurements and the GNSS velocity measurement:

Assumption 3 (Body angles)**.** *The angles obtained from both the IMU (${}^{E}_{S}\beta$) and the GNSS ${}^{E}_{B_*}\beta$ represent the same rotation as the body angle:*

$$\mathcal{F}_b \approx \mathbf{R}_z^{\left({}^{E}_{S}\beta\right)}\mathcal{F}_e \approx \mathbf{R}_z^{\left({}^{E}_{B_*}\beta\right)}\mathcal{F}_e \tag{5-7}$$

Henceforth both the representations of the body angle will be written as ${}^{E}_{B}\beta$.

Assumption 4 (Velocity representation)**.** *The body velocity as measured by the GNSS receiver is the same as the velocity measured by the vehicle sensors. That is both represent a velocity in the x direction of the body frame.*

The measurement models for the IMU, VS and GNSS are denoted as $\hat{\mathbf{y}}_1^-$, $\hat{\mathbf{y}}_2^-$, $\hat{\mathbf{y}}_3^-$:

$$\hat{\mathbf{y}}_{n,k}^- = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_{n,k} \tag{5-8}$$

$$\hat{\mathbf{y}}_{1,k}^- = \left({}^{E}_{B}\beta_k\right) + \mathbf{v}_{1,k} \tag{5-9}$$

$$\hat{\mathbf{y}}_{2,k}^- = \left({}^{E}u_k, {}^{E}\delta_k\right) + \mathbf{v}_{2,k} \tag{5-10}$$

$$\hat{\mathbf{y}}_{3,k}^- = \left({}^{E}x_k, {}^{E}y_k, {}^{E}_{B}\beta_k, {}^{B}u_k\right) + \mathbf{v}_{3,k} \tag{5-11}$$

Since both the state β and the IMU β reference the magnetic north it can be safely assumed that, the measured directions are parallel as long as a sufficient distance from the earth's magnetic north location is maintained. Since the measurement vectors are of different size the measurement update equations and their matrices are affected. For the sake of clarity the formulas for this part of the filter are included here:

Measurement update

$$\mathbf{P}_{\mathbf{\Delta},\hat{\mathbf{y}}_\mathbf{k}\hat{\mathbf{y}}_\mathbf{k}} = \sum_{i=0}^{2L} W_i^c \left(\mathcal{Y}_{i|k-1} - \hat{\mathbf{y}}_k^-\right)\left(\mathcal{Y}_{i|k-1} - \hat{\mathbf{y}}_k^-\right)^T + \mathbf{R}^n \qquad \in \mathbb{R}^{M\times M} \tag{5-12}$$

$$\mathbf{P}_{\mathbf{\Delta},\hat{\mathbf{x}}_\mathbf{k}\hat{\mathbf{y}}_\mathbf{k}} = \sum_{i=0}^{2L} W_i^c \left(\mathcal{X}_{i|k-1} - \hat{\mathbf{x}}_k^-\right)\left(\mathcal{Y}_{i|k-1} - \hat{\mathbf{y}}_k^-\right)^T \qquad \in \mathbb{R}^{L\times M} \tag{5-13}$$

$$\mathcal{K}_{\Delta,k} = \mathbf{P}_{\mathbf{\Delta},\hat{\mathbf{x}}_\mathbf{k}\hat{\mathbf{y}}_\mathbf{k}}\mathbf{P}_{\mathbf{\Delta},\hat{\mathbf{y}}_\mathbf{k}\hat{\mathbf{y}}_\mathbf{k}}{}^{-1} \qquad \in \mathbb{R}^{L\times M} \tag{5-14}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}_k\left(\mathbf{y}_\mathbf{k} - \hat{\mathbf{y}}_\mathbf{k}^-\right) \qquad \in \mathbb{R}^{L} \tag{5-15}$$

$$\mathbf{P}_k = \mathbf{P}_\mathbf{k}^- - \mathcal{K}_{\Delta,k}\mathbf{P}_{\mathbf{\Delta},\hat{\mathbf{y}}_\mathbf{k}\hat{\mathbf{y}}_\mathbf{k}}\mathcal{K}_{\Delta,k}^T \qquad \in \mathbb{R}^{L\times L} \tag{5-16}$$

Where L denotes the size of the state-vector and M the size of the measurement vector.
Note that the rest of the UKF implementation is the same the one described in section 4-2-3

5-2 Filter Verification Simulations

In this subsection the filter proposed in the previous section will be evaluated using simulated data. The goal of these simulations is to prove that the filter works and evaluate it's inner workings.

5-2-1 Generation of Simulated data

The data used for the simulation was generated using the process model described in equation equation (5-3). The fourth and fifth state were used as input and the propagated output was used as the reference for analysis.
The obtained time series was then subsampled at different rates (20[Hz], 14[Hz], 1[Hz]) to represent virtual IMU, VS and GNSS data.
For the ease of implementation it was assumed that the IMU only provides a direct measurement of the yaw angle.

The obtained data was then contaminated with noise based on the requirements of the trial. The kalman filter scaling parameters and weighing functions are set to the following:

$$L = 5 \tag{5-17}$$

$$\alpha = 10^{-2} \tag{5-18}$$

$$\kappa = 3 - L \tag{5-19}$$

$$\lambda = L - \alpha^2\left(L + \kappa\right) \tag{5-20}$$

Note that the choice for λ differs from the 'suggestion' by switching the sign of the L and α. Without this change the first weighing factor, calculated using $W_0^{(m)} = \frac{\lambda}{2(L+\lambda)}$ would have a highly negative value. This would in turn result in grievous state errors. As mentioned in section 4-2-3, no clear or agreed upon way to choose these parameters is available[2] therefore changing the definition of λ does not violate UKF principles.

[2]Discovering a way to tune the UKF's parameters λ, β, κ may be a worthy subject of study.

5-2-2 Simulations and Results

To evaluate the filter four different test were performed:

1. Only VS no noise
   The goal of this test is to verify the integrity of the programming of the filter solution itself.

2. Full fusion using known noise matrices
   The goal of this test is to see if there are any issues with multi-sensor fusion in the filter.

3. Full fusion using unknown noise matrices and erroneous initial conditions
   This test is a good indication on how the filter responds to guessed noise matrices. This is important since for the real-world data determining accurate noise matrices is difficult. Therefore it is useful to know how 'sensitive' the filter is in this respect.

4. Full fusion with GNSS outage using known noise matrices
   This test aligns with the requirements set out in section 1-2-2 concerning the intermittent nature of the GNSS localisation.

Simulation 1: Only VS

When looking at figure 5-1 it is clear that the noiseless data does not reproduce the generated data completely. This may seem to suggest an erroneous filter. However, after further studying the filter in-depth it was determined that the choice for scaling factor λ which in turn influences γ and the weighing factors has a strong influence the weight of the estimate with respect to the 'off-centre' sigma-points (equation (4-24)). Therefore, by favouring the estimate more strongly.
For example, figure 5-1 was generated using a $\frac{W_0^m}{\sum_{i=1}^{2L+1} W_i^m} = 1$. If a different scaling is chosen for example $\frac{W_0^m}{\sum_{i=1}^{2L+1} W_i^m} = 4$, the actual result become much closer to the expected values (see figure 5-2).

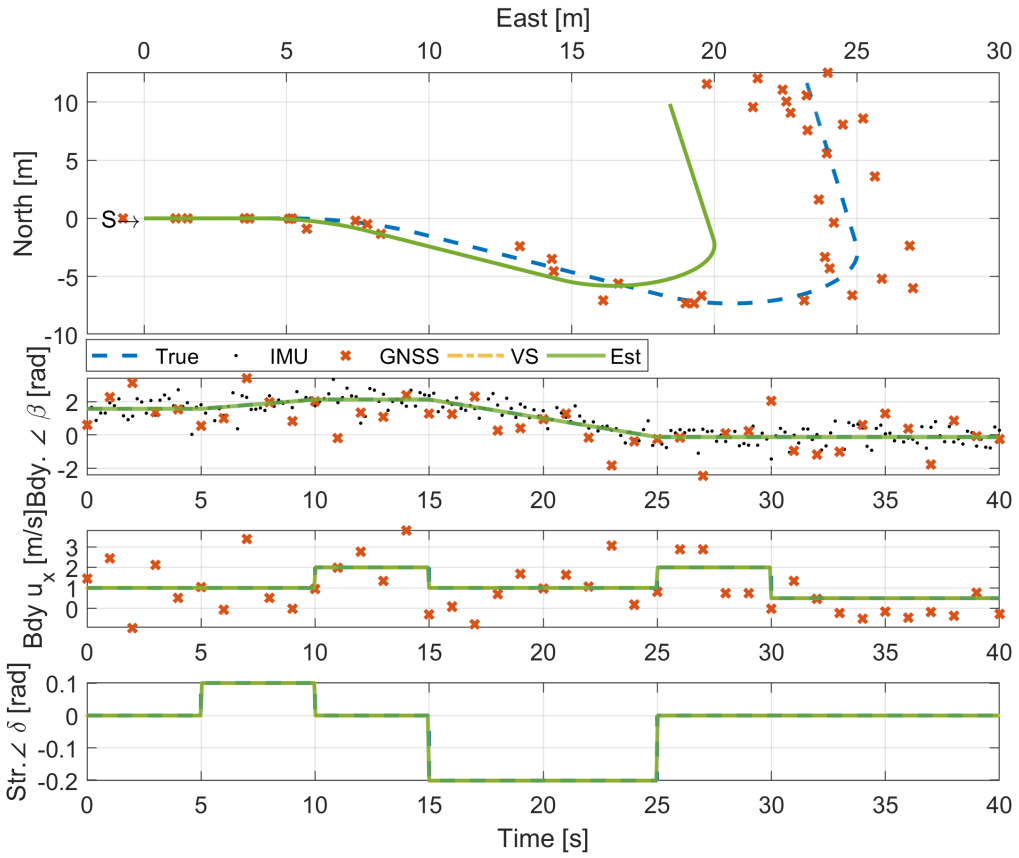


Figure 5-1: Simulation 1 State Plot 1: For this figure it is important to note that only the VS data is relevant. For this figure Sigma-point weighing is set to $W_0^m \left(\sum_{i=1}^{2L+1} W_i^m\right)^{-1} = 1$

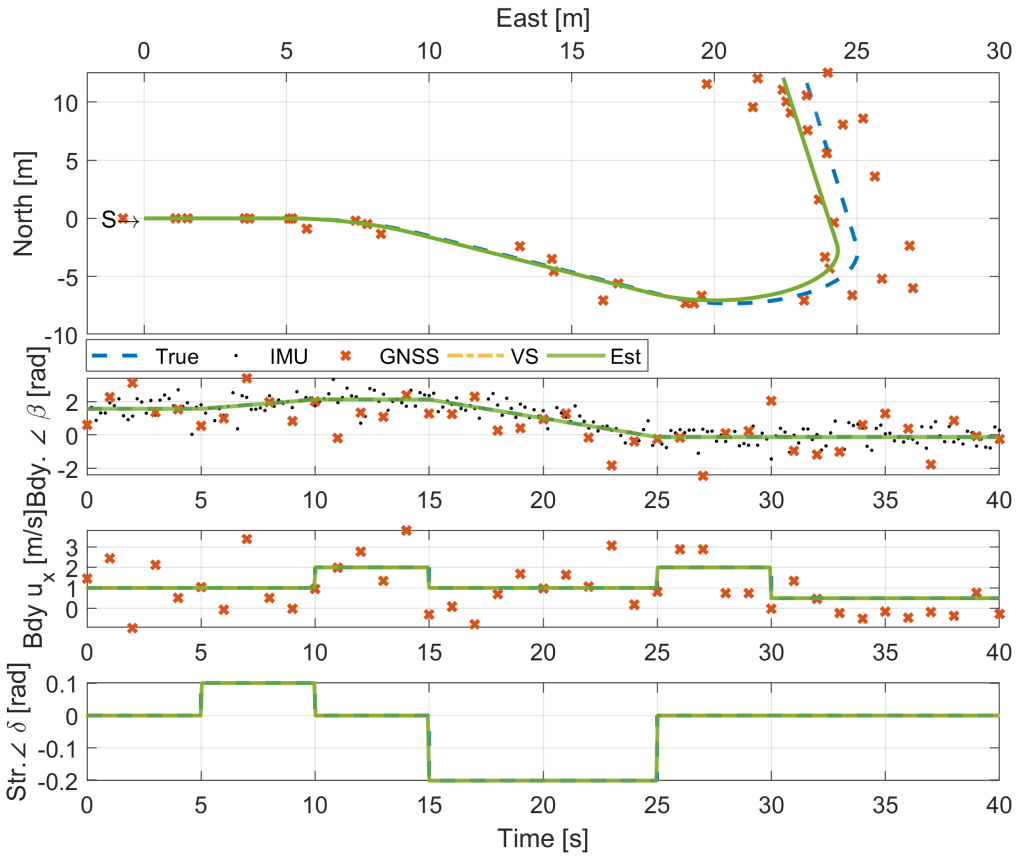


Figure 5-2: Simulation 1 State Plot 2: For this figure it is important to note that only the VS data is relevant. Note that for this figure the Sigma-point weighing is set to $W_0^m \left(\sum_{i=1}^{2L+1} W_i^m\right)^{-1} = 4$.

5-2-3 Simulation 2: Full Fusion, Known Noise

For this experiment, the noise matrices $\mathbf{R}^n_{1,2,3}$ for the IMU, VS and GNSS respectively are fed into the Multi-Rate UKF. The used noise matrices for this experiment are:

$$\mathbf{R}^v = diag\left(10^{-8}[m^2], \quad 10^{-8}[m^2], \quad 10^{-12}[rad^2], \quad 10^{-3}[m^2], \quad 10^{-3}[rad^2]\right) \tag{5-21}$$

$$\mathbf{R}^n_1 = 0.500[rad^2] \tag{5-22}$$

$$\mathbf{R}^n_2 = diag\left(0.0256[m^2/s^2], \quad 0.010[rad^2]\right) \tag{5-23}$$

$$\mathbf{R}^n_3 = diag\left(2.250[m^2], \quad 2.250[m^2], \quad 0.360[m^2/s^2], \quad 0.160[rad^2]\right) \tag{5-24}$$

The results are displayed in figure 5-3 and figure 5-4. As can be seen from the results, the filter is successfully able to estimate the body angle and velocity. It has more difficulty estimating the actual position of the vehicle. The reason behind this is that, apart from the relatively higher noise of the steering angle δ_f the continuous curve generates more points laying on the inside of the corner. This in turn results in the position estimate laying in the inside of the corner.

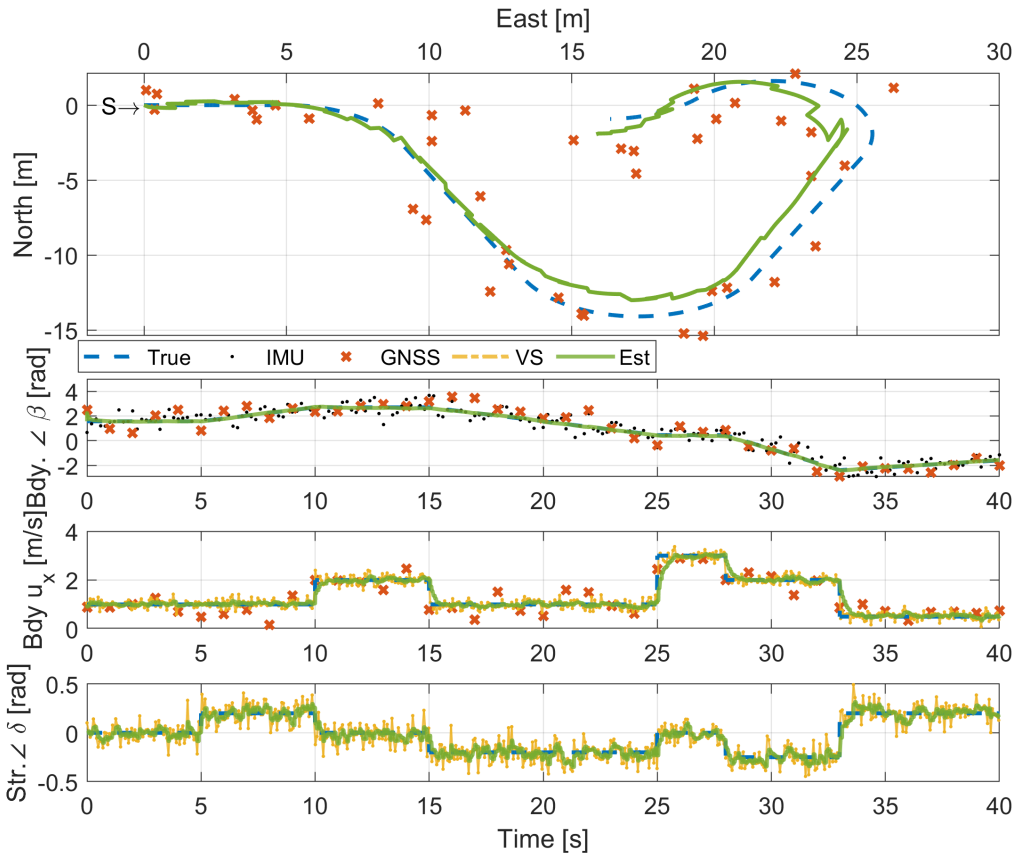


Figure 5-3: Simulation 2 State Plot: The first sub-plot displays the position of the vehicle, the second its body angle ${}^{E}_{B}\beta$ the third the body velocity ${}^{B}u$ and the fourth the steering angle ${}^{B}\delta_f$

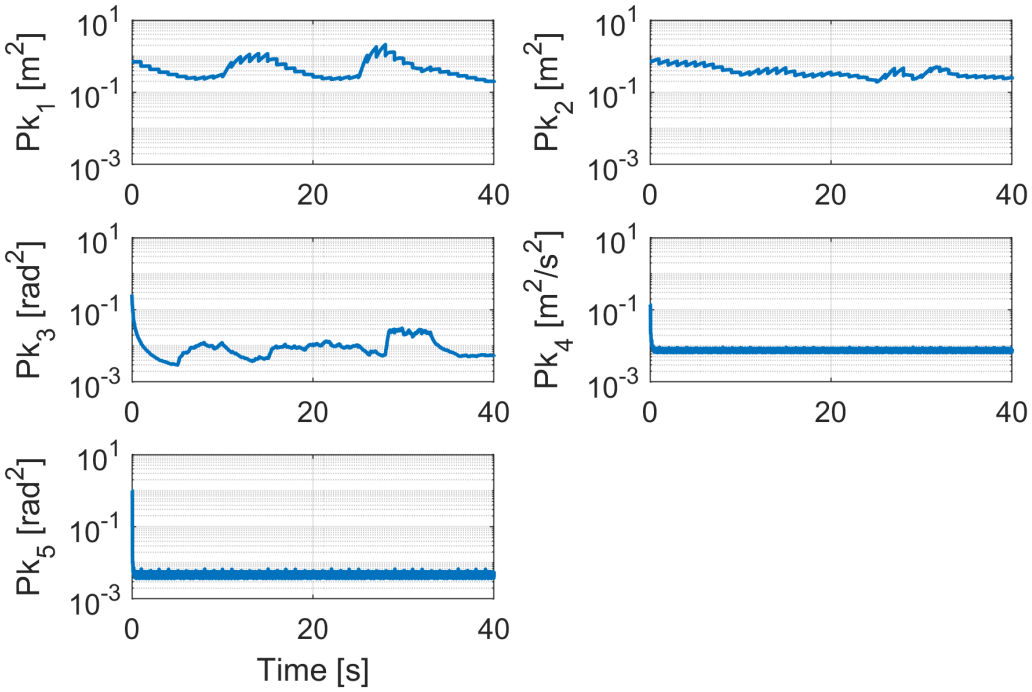


Figure 5-4: Simulation 2 $trace\,(\mathbf{P_k})$

5-2-4 Simulation 3: Full Fusion, Unknown Noise and Erroneous initial conditions

For this experiment the following noise matrices and initial conditions are implemented:

$$\hat{\mathbf{x}}_0 = (-2[m], \quad -2[m], \quad \pi[rad], \quad 0[m/s], \quad 0[rad]) \tag{5-25}$$

$$\mathbf{R}_1^n = 5 \cdot 0.500[rad^2] \tag{5-26}$$

$$\mathbf{R}_2^n = 5 \cdot diag\left(0.0256[m^2/s^2], \quad 0.010[rad^2]\right) \tag{5-27}$$

$$\mathbf{R}_3^n = 5 \cdot diag\left(2.250[m^2], \quad 2.250[m^2], \quad 0.360[m^2/s^2], \quad 0.160[rad^2]\right) \tag{5-28}$$

Compared to simulation 2 the noises that have been entered in the UKF have been multiplied by five with respect to their actual values. When looking at the results as displayed in figure 5-5 and figure 5-6 one can see that the filter responds adequatly to the erroneous starting condition. Furthermore, as expected the filter performs worse without the actual noise characteristics. This is clearly visible when looking at the overal position and the response to changes in velocity.
It is interesting to note that the filter furthermore doesn't strongly responds to the GNSS position. The cause of this behaviour lies at the trust that is given to the process model. The factor five increase in estimated measurement noise does not convince the filter it is at the wrong position.

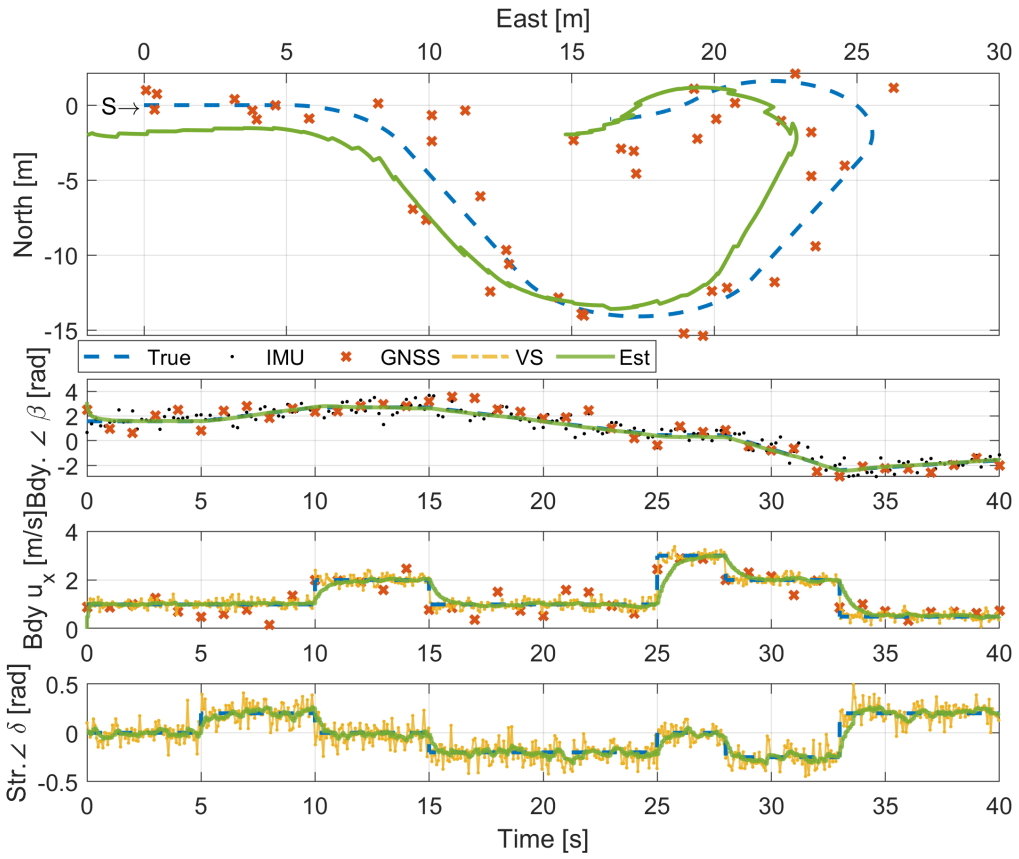


Figure 5-5: Simulation 3 State Plot: The first sub-plot displays the position of the vehicle, the second its body angle ${}^{E}_{B}\beta$ the third the body velocity ${}^{B}u$ and the fourth the steering angle ${}^{B}\delta_f$

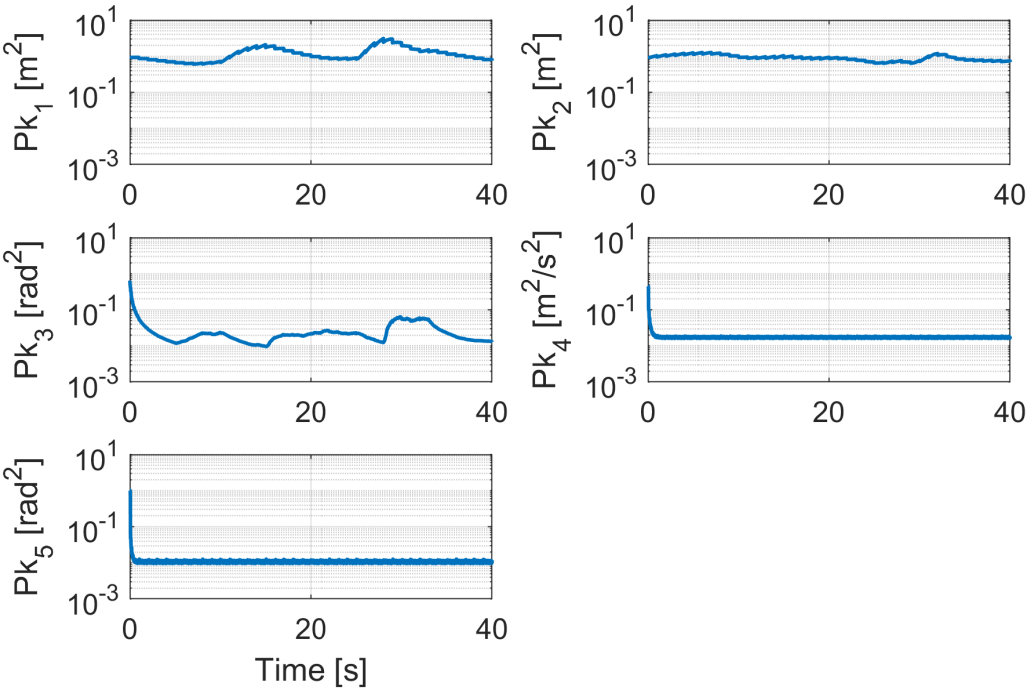


Figure 5-6: Simulation 3 $trace\,(\mathbf{P_k})$

5-2-5 Simulation 4: GNSS Outage Simulation

This simulation has been performed using the noise matrices from simulation 2 as well as its initial conditions . The GNSS outage was simulated by removing the GNSS data in the interval between 20-30s.

When looking at the state results expressed in figure 5-7 it is clear that when compared to figure 5-3 the quality of the localisation solution has decreased. The last corner in the top right of the first subplot clearly shows that, when GNSS data becomes available again a large corrective jump is performed. This results in an overshoot of the corner in the current simulation with respect to simulation 2.
Interestingly, only a minor difference exists between the $trace\,(\mathbf{P}_k)$ of simulation 2 figure 5-4 and simulation 4 figure 5-8. The largest difference is visible in the $\mathbf{P}_k1$ and $\mathbf{P}_k2$ subplots. In these figures the time that the GNSS localisation is switched off the σ_x^2 and σ_y^2 becomes more smooth. This indicates that the GNSS updates when they are available, are conflicting with the position estimate of the previous sensor updates. Hence the 'sawtooth' pattern[3].

[3]This can be seen as an omen for what is to come during the real-world experimental results

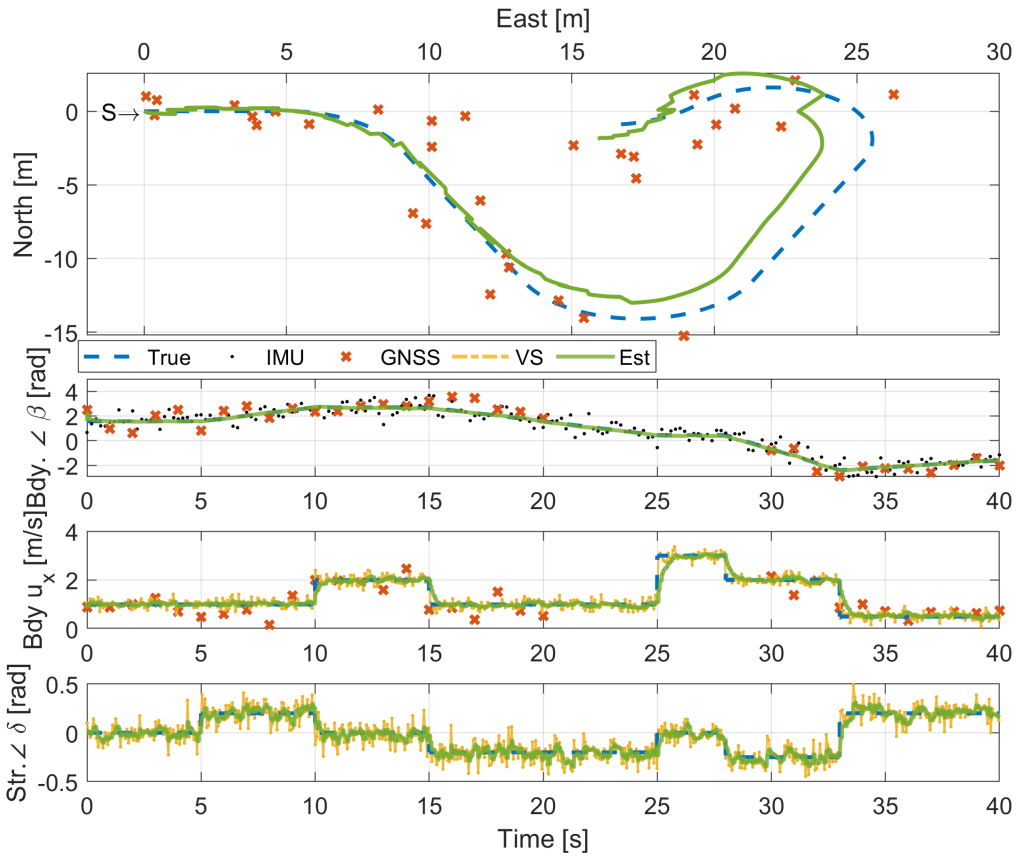


Figure 5-7: Simulation 4 State Plot: The first sub-plot displays the position of the vehicle, the second its body angle ${}^{E}_{B}\beta$ the third the body velocity ${}^{B}u$ and the fourth the steering angle ${}^{B}\delta_f$

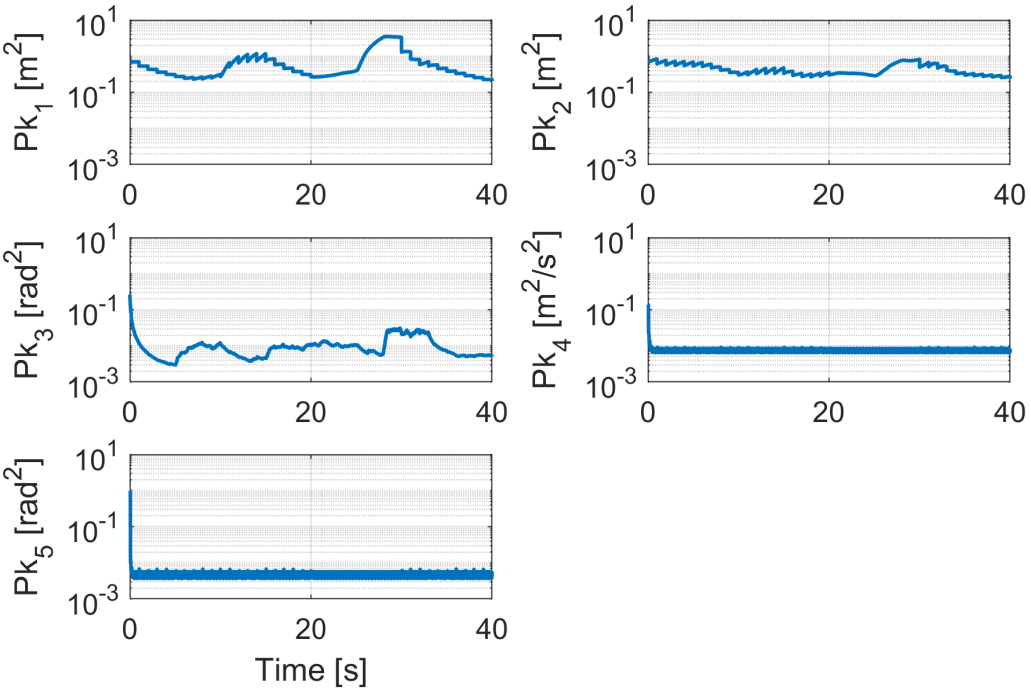


Figure 5-8: Simulation 4 $trace\,(\mathbf{P_k})$

5-3 Simulation Conclusions

From the results it can be seen that the chosen filter structure is capable of dealing with varying data-rates. Furthermore, it is clear to see that, with the exception of experiment 3 the filtered data provides an improved localisation solution with respect to the raw GNSS data.
The results from simulation 3 indicate that the determination of proper noise matrices that have their basis in actual sensor characteristics is of great importance. Without proper scaling of these parameters the fusion result can deteriorate the results. Lastly, the results from simulation 4 show that the filter is capable of dealing with intermittent GNSS measurements. In conclusion, based on the simulation results, the filter proves to be capable of handling variable data-rates and GNSS intermittency. Thus the filter in simulation satisfies the second and third requirement for the achievement of the second thesis goal section 1-2-2.

Chapter 6

Localisation Trials and Results

In this chapter the filter developed in the previous chapter (chapter 5) is used for the evaluation of real-world data collected using the vehicle-platform. The first section will explains the real-world time-synchronisation between different data-sources. The second section, section 6-2 will give the reader the chosen UKF parameter values as well as the noise matrices. This section is followed by the real world experiments in section 6-3.

6-1 Time Synchronisation

As each filter update is performed when a measurement from one of the three sensors is available, the Δt, required for the process update (equation (5-3)), is calculated based on the last measurement update. As the update rates of the sensors, with the exception of the Global Navigation Satellite System (GNSS) receiver, are not strictly regulated the inter frame time $f_{\Delta t}$ per sensor may vary and can be expressed as:

$$f_{\Delta t,IMU} = t_{s1} + \Delta t_{p1} + \Delta t_{s1} \tag{6-1}$$

$$f_{\Delta t,VS} = t_{s2} + \Delta t_{p2} + \Delta t_{r} \tag{6-2}$$

$$f_{\Delta t,GNSS} = t_{s3} + \Delta t_{s2} \tag{6-3}$$

Where t_s is the sample time (which vary per sensor and may not be consistent), the Δt_p expresses the time it takes to pack the measured data into a packet, Δt_s denotes the synchronisation error between the Rapsberry PI time and the timebase used on the sensor. Lastly, Δt_r denotes the time difference between the availability of the packet in the CAN-BUS buffer and the actual time the package gets time-stamped.
The following assumptions are made when processing measurements:

Assumption 5 (Equal Time source)**.** *It is assumed that the measurement data, which is timestamped based on UTC is the same, which is not guaranteed since the IMU receives it's UTC from the Raspberry PI without Receive/Transmission delays being considered and*

the GNSS based UTC of the GNSS receiver is dependent on the atomic clocks on-board the satellites and is thus per definition another timesource than the UTC of the Raspberry PI[1]

Assumption 6 (No delay between measurement and frame construction)**.** *This especially holds true for the IMU as the packet creation time is not accounted for. The error due to this is expected to be small however* $\approx 1[ms]$.
The Vehicle Sensors (VS) suffers from from inaccuracies due to time-stamping as well. The used MCP2515 doesn't support hardware based timestamping through the IEEE 1588 (Precision Time Protocol). Therefore the messages are timestamped using software which is less accurate.

6-2 Multi-Rate UKF Configuration

In this section an overview of the used parameters for the experiments will be given. These include the noise matrices $\mathbf{R}^v$ and $\mathbf{R}^n_{1,2,3}$ as well as the UKF scaling parameters.

The noise matrices for the measurement data are partially based on actual measurements as well as 'educated guesses'. The noise analysis itself can be found in appendix A.

$$\mathbf{R}^v = diag\left(10^{-8}[m^2], \quad 10^{-8}[m^2], \quad 10^{-12}[rad^2], \quad 10^{-3}[m^2], \quad 10^{-3}[rad^2]\right) \tag{6-4}$$

$$\mathbf{R}^n_1 = 0.5[rad^2] \tag{6-5}$$

$$\mathbf{R}^n_2 = diag\left(0.0256[m^2/s^2], \quad 0.01[rad^2]\right) \tag{6-6}$$

$$\mathbf{R}^n_3 = diag\left((HDOP)^2\,[m^2], \quad (HDOP)^2\,[m^2], \quad 0.36[rad^2], \quad 0.16[m^2/s^2]\right) \tag{6-7}$$

Where Horizontal Dilation of Precision (HDOP) is used to dynamically change the noise estimate for the GNSS receiver.

6-3 Experiments

In this section two real-world experimental results will be presented.

6-3-1 Experiment 1 GNSS & VS Fusion

In this experiment the vehicle-platform was driven along the Bieslandsepad in Delft. As can be seen from the fusion results the GNSS sensor has a significant error with respect to the actually travelled path (see figure 6-3). The 'sawtooth' pattern alluded to in the previous chapter makes its appearance again in the $\mathbf{P}_k1$ and $\mathbf{P}_k2$ plot of figure 6-2.
It is clear that the fusion result based on only the GNSS and VS is lacking an additional direction reference. Now only the GNSS heading is available, resulting in a clear deviation from the actual path. It is interesting to note that the filter solution does ignore the GNSS position when the corner is taken to pass under the overpass. This indicates that the filter

[1] Although, timestamping is performed based on the Raspberry PI's time as well.

trusts the propagated vehicle sensor results more than the GNSS position result. This is possibly reinforced by the HDOP VCM update rule (appendix A).

The a postiori VCM state descriptions in figure 6-2 clearly show that the the body angle ${}^{B}\beta$ is initialised incorrectly. This covariance shows a sharp decrease, starting roughly from the moment when the first GNSS based heading estimates are available (visible in figure 6-1). As expected, the variance of the body velocity ${}^{B}u$ and ${}^{B}\delta_f$ decreases immediately after the experiment starts. This is expected because these parameters are directly measured by the VS and thus are updated at least 14 times per second.
With respect to the other variances, the variances describing the global position of the vehicle-platform ${}^{E}x$ and ${}^{E}_{y}$ do not decrease as sharply. This suggests that the filter is less certain about the accuracy of its location than about the orientation, velocity or steering angle.

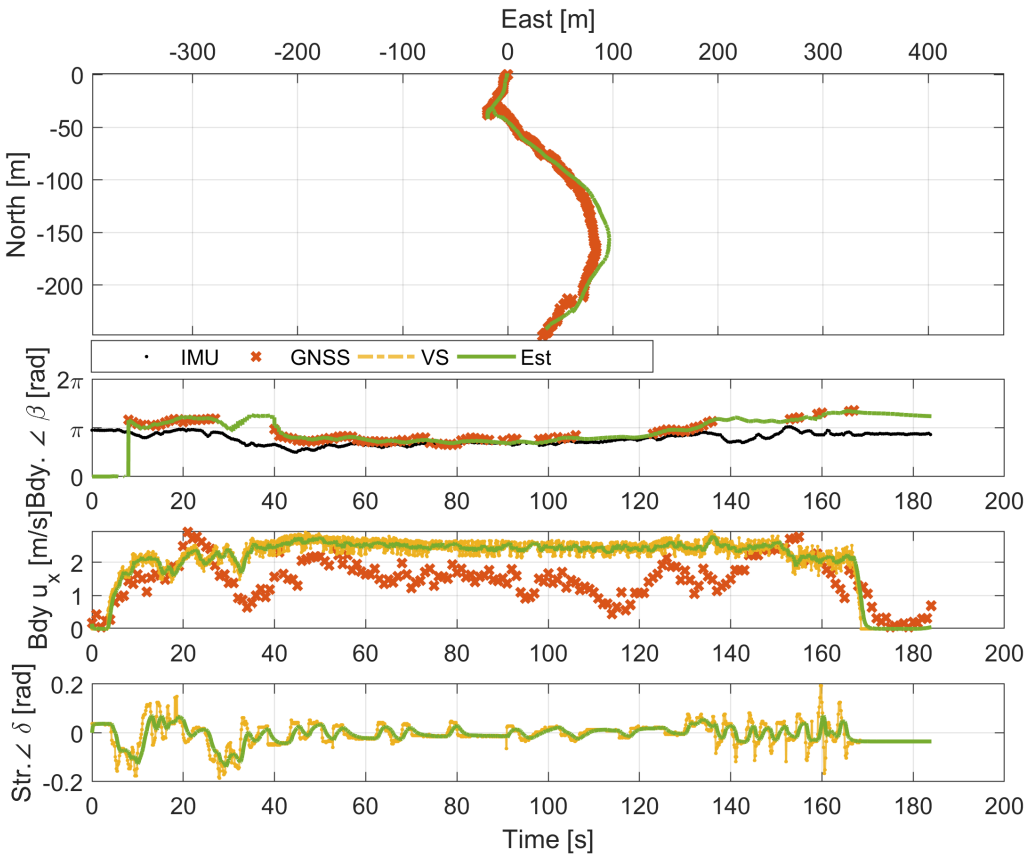


Figure 6-1: Simulation 3 State Plot: The first sub-plot displays the position of the vehicle, the second its body angle ${}^{E}_{B}\beta$ the third the body velocity ${}^{B}u$ and the fourth the steering angle ${}^{B}\delta_f$. Note that, although the IMU based body angle measurement is given it is not used in the filter.

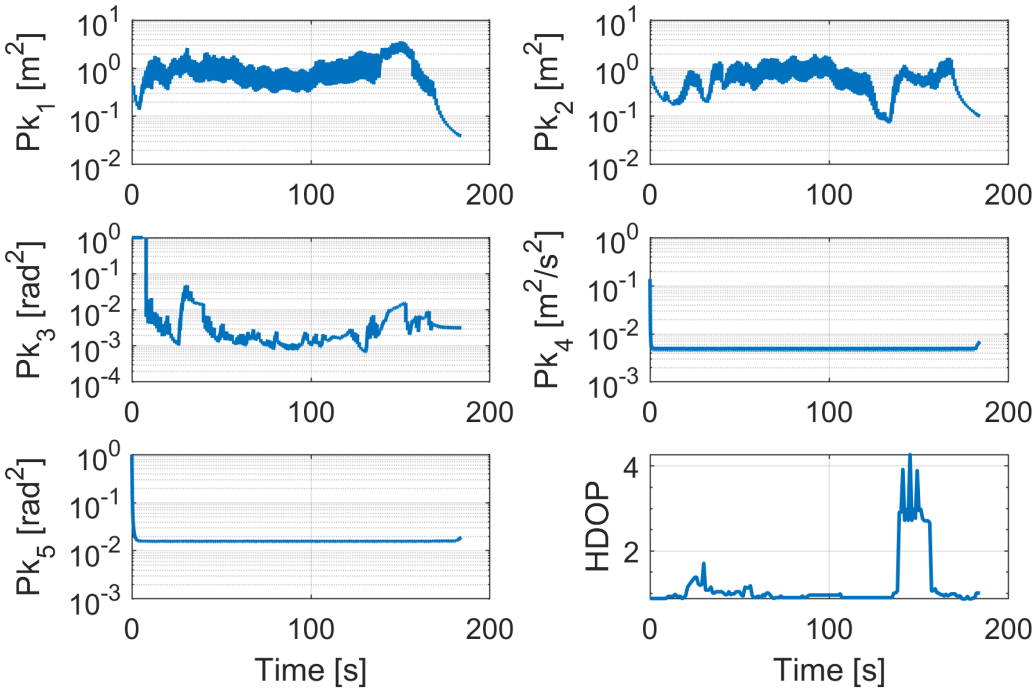


Figure 6-2: Simulation 3 $trace\left(\mathbf{P_k}\right)$

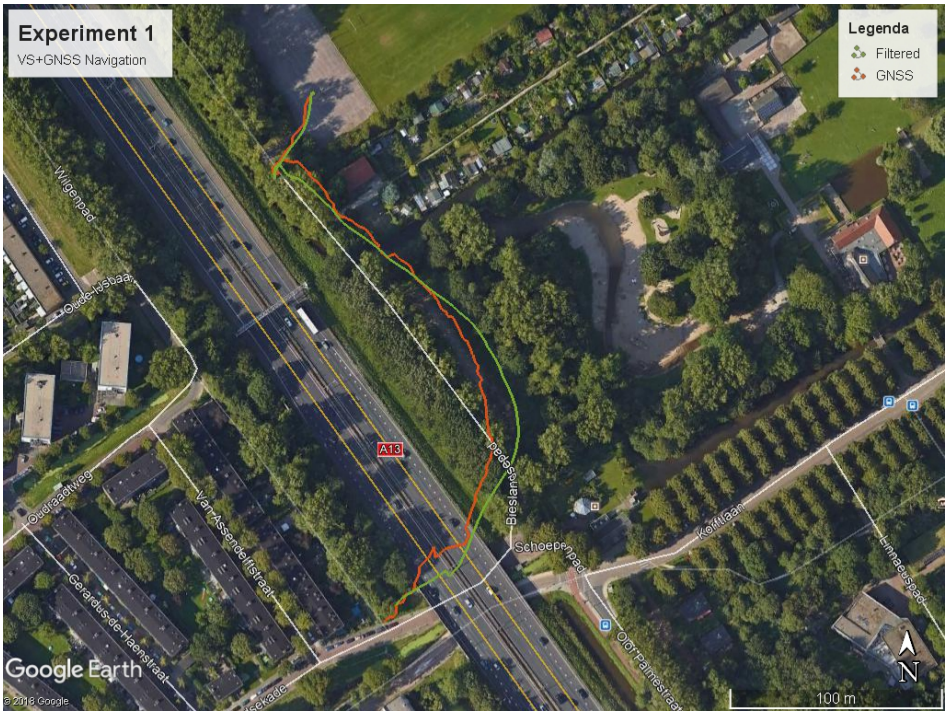


Figure 6-3: This figure shows the GNSS data as well as the filtered output. The vehicle drove along the Bieslandsepad (the path hidden behind the treeline).

Experiment 2

For this experiment the vehicle-platform was driven around on the sidewalk. Figure 6-6 displays the results of the filter in the real-world. Note that the actually travelled path's start is shared by all lines (since it is based on the initial GNSS fix). From then onwards the route travelled was northward, then turning left and taking all subsequent first available left corners.

It is clear from the results that the filter failed to reject the incorrect GNSS points. When looking at the path travelled if only the VS were to be used, it is clear to see that the VS sensor are more accurate where total distance travelled is concerned. This is clear even without changing the λ parameter as was done with the first simulation in the previous chapter.

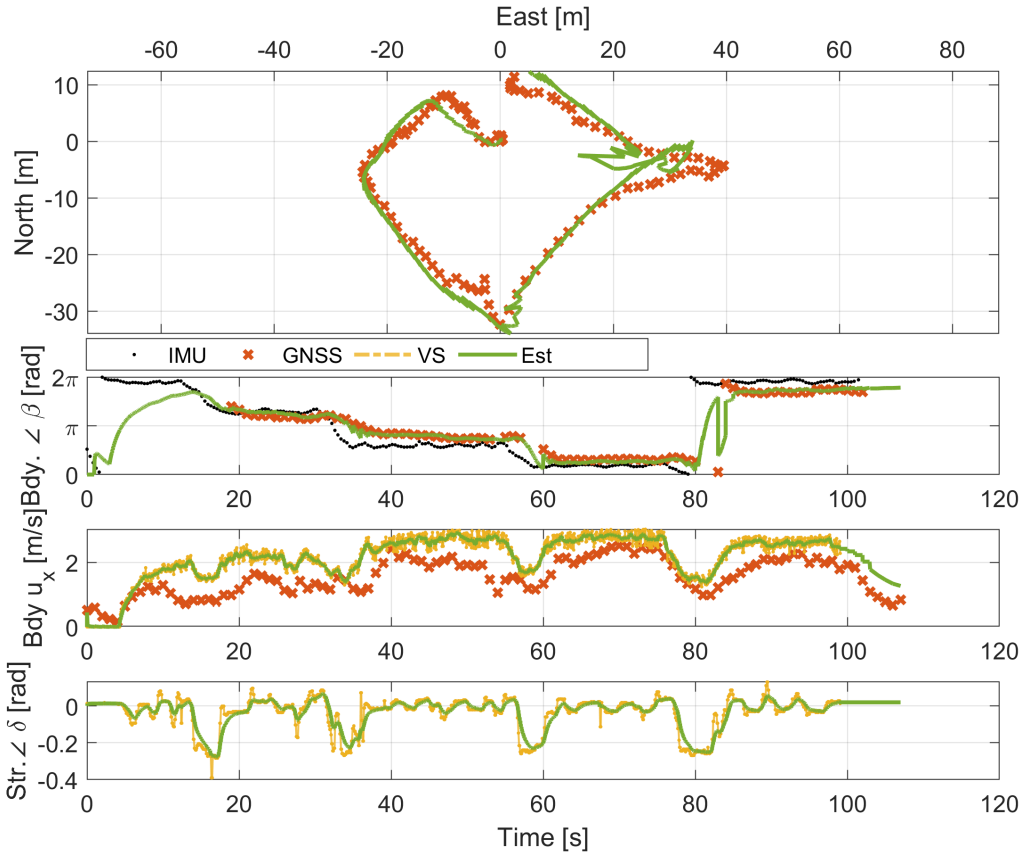


Figure 6-4: Simulation 3 State Plot: The first sub-plot displays the position of the vehicle, the second its body angle ${}^{E}_{B}\beta$ the third the body velocity ${}^{B}u$ and the fourth the steering angle ${}^{B}\delta_f$

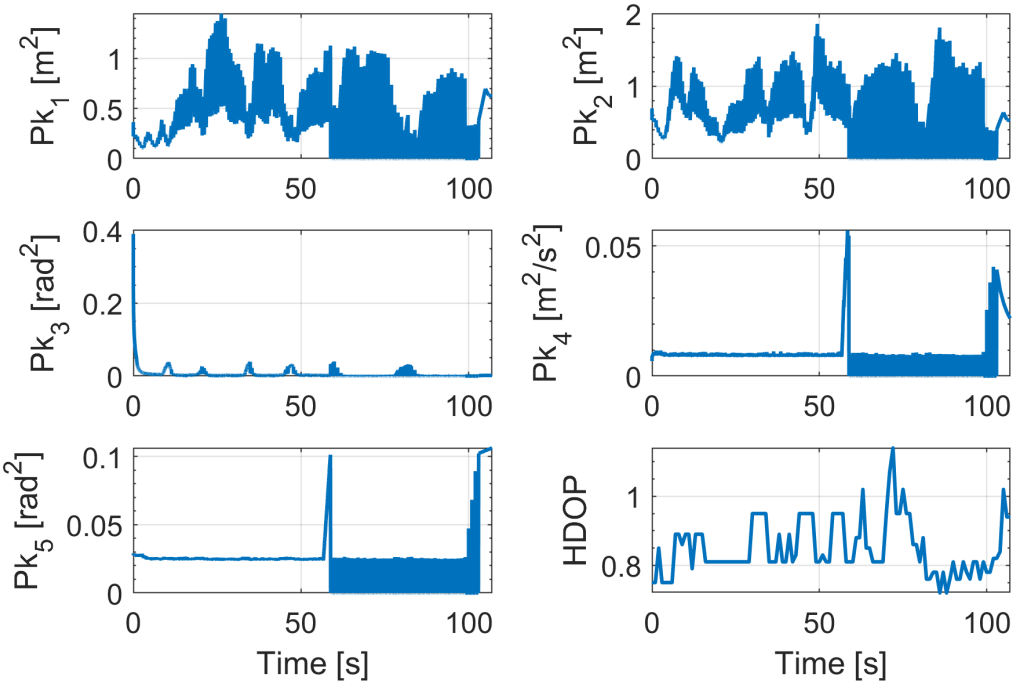


Figure 6-5: Simulation 3 $trace\left(\mathbf{P_k}\right)$

Figure 6-6: This figure shows the results of the filter in green, the GNSS data in red and the VS only result in yellow. The actual travelled path is displayed in white

Chapter 7

Discussion and Future Work

In this chapter the combined results of the thesis will be discussed and future work to improve the current results will be discussed. Firstly, section 7-1 will start with a description of the different subjects presented in the thesis, after which these subjects will be discussed in their own subsections. Secondly, section 7-2 will mention several avenues for future research.

7-1 Discussion

As the work presented in this thesis knows three distinct parts namely, the soft & hardware development, the research conducted into IMU orientation filtering and the development of a multi-rate UKF for localisation and pose estimation these parts will be discussed in their own subsection.

7-1-1 Hardware and Software

The development of the software was an integral part of the performed work in the thesis. The initial goal to develop an on-line filter using self-developed code is the only software related thing that has not been achieved at the end of this thesis. Due to delays in the delivery of the hardware platform it was not possible to spend the time required to implement a real-time filter, though the developed software and used hardware should, with minor tweaks be more than capable of running the proposed filter at high enough update rates.

Though the real-time filter may not be implemented, it is safe to say that the rest of the code developed is well engineered, documented and performs to specifications. The used process and threads structure allows for high data-rates on the Raspberry PI side. The implemented protocol running on the ESP32 gives superior performance to out of the box methods and should be good stepping stones for future research projects.

7-1-2 IMU Orientation Filtering

The results of the research performed in the area of IMU orientation filtering (chapter 4) provided results that were in line with the literature. Even though in the end the Madgwick filter [35] was chosen instead of the more complex UKF based frame fusion of gyroscope with magnetometer and accelerometer, it was worth noting that apart from De Marina et al. [42] very little literature is available on orientation filtering using UKF. This especially holds true for the UKF based filter that used Madgwicks's integration scheme. Though, through its poor performance it has demonstrated that this way of fusion is not recommended.
One of the main limitations of the research done on IMU orientation filtering is the absence of a 'ground truth' or reference signal. This leads to the results being only quantitative in nature.

7-1-3 Multi-Rate UKF

The choice for a Multi-Rate filter instead of a more conventional Kalman based filter is validated through the modularity constraint as laid out in chapter 1. Moreover, a Multi-Rate filter allows for more data-points to be used by the filter without resulting to things such as spline interpolation, zero or higher order hold functions. It was expected however that, based on equivalent research into the field of GNSS IMU and VS fusion such as the work by Melendez-Pastor et al [15], the results of a stand-alone GNSS receiver would be improved upon.
This expectation was not challenged by the simulation results, as can bee seen in chapter 5. In this chapter simulation 2 and simulation 4 clearly show superior performance when using the proposed Multi-Rate UKF. However, when the noise matrices are not properly dimensioned the results are less positive. Simulation 3, clearly shows that incorrect noise parameters reduce the filter results.

Based on the real-world experiments, it is more difficult to state that the filter's performance is an improvement to stand-alone GNSS. For experiment 1 the case can be made that this is indeed the case since the filter output arguably describes the travelled route better than the GNSS. It is likely that a large part of the poor filter results can be attributed to abysmal GNSS performance. In experiment 2 it is clearly visible that the GNSS measurements forcibly constrict the filter solution. The poor performance of the GNSS receiver can be due to a multitude of reasons. One of these reasons may be the configured elevation mask of only 20[°]. The elevation mask determines which satellites are used for the localisation solution. The signals of these lower elevation satellites are subject to larger error due to multi-path errors as well as ionospheric and tropospheric delays. However, the difficulty of magnetometer calibration as discussed in appendix A-1 may be seen as an accomplice.
Even though the real-world results presented in the thesis are not encouraging, it does not necessarily mean that the conclusion can be drawn that the filter structure itself is faulty as, especially the second experiment is by its very nature flawed. Since it does not allow the filter to stabilise by first driving in a straight line for a prolonged time[1].

[1]For the presentation of the thesis an attempt will be made to obtain better measurement data.

7-2 Future Work

In this section the avenues for future work are discussed, starting of with an extensive description of issues with the current GNSS receiver and its configuration.
To mediate the poor real-world results of the Multi-Rate UKF, the GNSS receiver's performance must be optimised. At least, measuring the velocity and heading noise in ideal conditions should be attempted and, ideally related to measurable parameters such as HDOP. Furthermore, the current GNSS receiver, the ublox M8P should be replaced with a version that supports the Galileo satellite system as well. This would be highly advantageous since it would allow the system to be more robust to GNSS outages as well as provide improved localisation information.
Another way the GNSS results can be improved is by changing the configuration of the GNSS such as increasing the elevation mask (which determines which satellites are used for localisation, lower elevation is generally less accurate) of the GNSS to a value above the default 20[°], this should reduce multipath errors from signal reflections. Apart from these improvements the GNSS localisation should be improved by adapting Single Frequency Precise Point Positioning (SF-PPP) techniques.
Another improvement that could be made is integrating maps and vision into the system. This would allow for localisation based on this map data as well as using vision techniques to see the position on the road.

The IMU is currently mounted directly on the frame of the vehicle-platform. Since the vehicle-platform's frame is undampened, the noise pollutes the accelerometer measurements. By adding physical damping high frequency noise contamination can be decreased. This would allow the integration of the accelerometer into the measurement update of the UKF as a description of the steering angle (through tangential acceleration). It would moreover be beneficial to add a second magnetometer. This magnetometer could be used to improve calibration of the IMU's magnetometer by having a second reference.

The amount of used vehicle sensors should be increased. Especially measuring the angular velocity of the front wheels of the vehicle-platform could improve the filtering results. Using these rotational velocities as a measurement would add a additional references to the body velocity as well as the steering angle.

Concerning the Multi-Rate UKF it would be beneficial to study the effect of the inherent UKF parameters α, β and κ. Currently there is little to no available literature giving an in-depth description on how these parameters should be chosen or, how these parameters effect the weighing of the sigma-points. Moreover, it might be worth investigating the effect of adapting the weighing factors per sensor.

Chapter 8

Conclusion

The goals set out by Accenda were the development of a soft & hardware platform for future research in the field of autonomous navigation as well as the development of a pose estimation and localisation solution.
As described in chapter 2 the realisation of a soft & hardware platform is achieved. The sensors required for the pose estimation and localisation are integrated into the vehicle-platform. The communication rates of the sensors are guaranteed through the analyses performed.

The second goal concerning the development of a pose and localisation solution by means of data-fusion of GNSS, IMU and VS data is in its current state partially achieved.

Based on the simulations conducted (see chapter 5) the filter structure consisting of a separate IMU orientation filter and a Multi-Rate UKF seems to provide superior pose estimates when compared to stand-alone GNSS measurements while at the same time satisfying the modularity constraint as well as the intermittency constraint laid out in section 1-2-2. However, the current real-world results indicate that the filter is not capable of improving upon a stand-alone GNSS localisation solution.

In conclusion, although a cascading Multi-Rate UKF does provide the flexibility required to fuse measurements from different sensors into a single coherent state representation and is capable of providing superior to GNSS pose estimates, as proven by the simulations, current measurements indicate that this does not translate into the real-world.

Appendix A

Hardware Calibrations and Noise Measurements

In this appendix the different noise measurements of the used sensors are given

A-1 MPU 9250 Noise Measurements and Calibrations

In this part of the appendix the noise measurements and the calibration of the Invensense MPU9250 are described.

A-1-1 Accelerometer and Gyroscope Calibration

The way the accelerometer and gyroscope are calibrated is very simple, an N number of measurements is taken after which the mean is calculated. This mean is then stored as the bias variable. For the z-axis the earth's gravitation is extracted first before a mean is calculated.

A-1-2 Magnetometer Calibration

For the calibration of the magnetometer we apply the method as discussed by Kris Winer [54]. The calibration works on the basis that, assuming a constant magnetic field the following holds:

$$\mathbf{m} = (m_x, m_y, m_z) \tag{A-1}$$

$$|\mathbf{m}| = c \qquad \forall\,(\mathbf{p}, \mathbf{o}) \in \left\{\mathbb{R}^6 : \dot{\mathbf{p}} = 0\right\} \tag{A-2}$$

$$\tag{A-3}$$

Or in words, the magnitude of the measurement vector is constant for any rotations around a constant point in a three dimensional euclidean space.

Based on this statement the axes of the sensor can be calibrated using the following pseudo code:

Algorithm 1 Magnetometer Calibration

1: **procedure** CALIBRATE MAGNETOMETER SCALING AND BIAS
2: **for** *n=1:1500* **do**
3: $m(n,:) = (mx, my, mz)$
4: $m_{bias} = mean(m)$
5: $m_{chord} = max(m) - min(m)$
6: $\bar{r} = mean(m_{chord})$
7: $m_{scale} = \bar{r}/m_{chord}$

To obtain a suitable measurement of the magnetic field, it is important that the sensor is moved in figure eight's to calibrate the magnetometer. This movement is the same movement that is used for the calibration of the compass on a mobile phone. Since this movement is difficult to perform on an integrated sensor, the measured calibration values are hard-coded in the software. This approach is suitable as long as the magnetic field in case of the calibration is the same or highly similar to the magnetic field experienced while integrated.
During testing the sensor is integrated in the vehicle-platform, this results in a constant disturbance due to the inherent magnetic field of the metal frame of said platform. This means that calibration of the magnetometer is of the utmost important in this situation. However, due to the calibrations temperature dependency as well as the weight of the frame a calibration in $\mathbb{R}^3$ is difficult resulting in the calibration being performed in $\mathbb{R}^2$ which reduces the orientation accuracy.

A-1-3 Noise and Normality

It is assumed that the type of noise contamination of the Motion Processing Unit (MPU) is normally distributed:

$$\mathbf{a}_k = \begin{bmatrix} \check{a}_x & \check{a}_y & \check{a}_z \end{bmatrix}_k + \epsilon_a \tag{A-4}$$

$$\mathbf{g}_k = \begin{bmatrix} \check{g}_x & \check{g}_y & \check{g}_z \end{bmatrix}_k + \epsilon_g \tag{A-5}$$

$$\mathbf{m}_k = \begin{bmatrix} \check{m}_x & \check{m}_y & \check{m}_z \end{bmatrix}_k + \epsilon_m \tag{A-6}$$

This assumption is confirmed by appendix A-1-3 and appendix A-1-3 as it appears that the sensors results are indeed contaminated by white-Gaussian noise. It is important to note however that the measurements gyroscope, accelerometer and magnetometer are all to a degree dependant on temperature fluctuations [21,22][1]. However since both the accelerometer and magnetometer are used as an absolute directional reference, see section 4-2 the effect of temperature is negligible. Furthermore, it is assumed that calibration for the gyroscope, which is executed on each start-up of the sensor should for the most part compensate for temperature related drift.

[1] This is due tot the accelerometer and gyroscope using temperature sensitive oscillators.

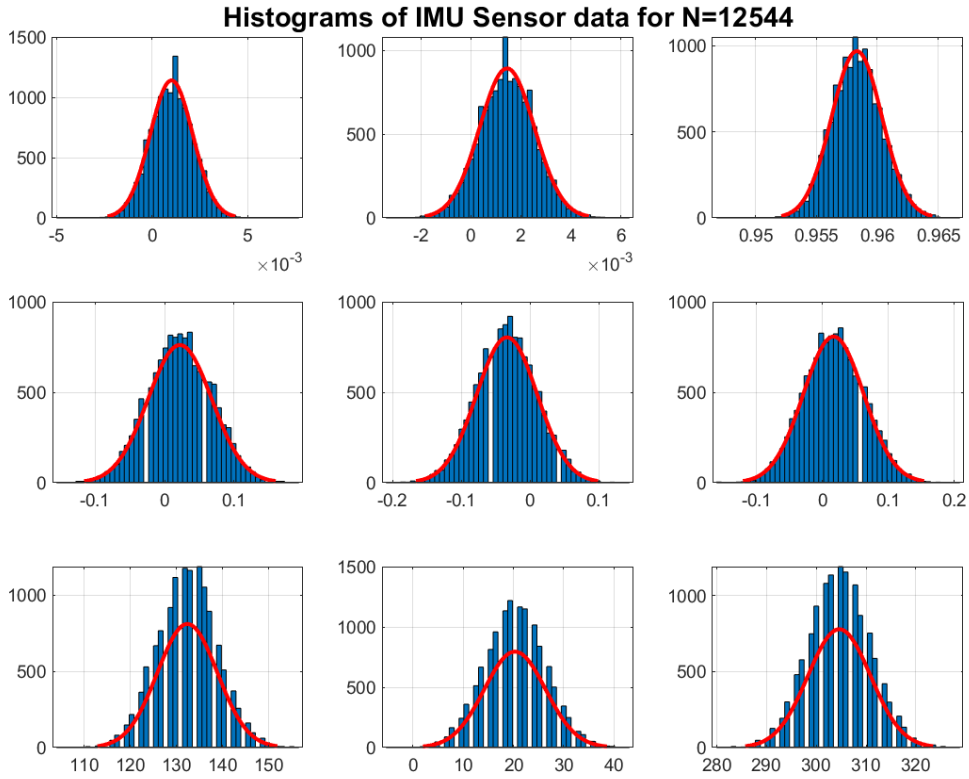


Figure A-1: The histogram of a sensor measurement performed with the sensor in a vise and update rate set to 20[Hz]. The nine plots give the sensor distributions per row. First row is the accelerometer, second the gyroscope and third the magnetometer/compass. Each column represents the sensors x, y and z axis respectively. The reason for the white lines is due to the amount of bins used(50).

Table A-1: This table gives the statistical parameters from the dataset as displayed in appendix A-1-3. Note that the unit for acceleration, gyration and magnetic fields is $[m/s^2]$, $[^\circ/s]$ and $[\mu T]$ respectively (for the μ and σ).

	Ax	Ay[]	Az	Gx	Gz	Gy	Mx	My	Mz
μ	0.001	0.001	0.958	0.022	-0.034	0.017	132.340	20.282	304.600
σ	0.001	0.001	0.002	0.046	0.044	0.046	6.486	6.105	6.298
Kurtosis	3.271	3.206	3.113	2.939	3.020	2.999	2.958	3.002	2.989
Skewness	0.033	0.003	0.100	0.006	-0.012	0.007	0.002	0.005	0.045

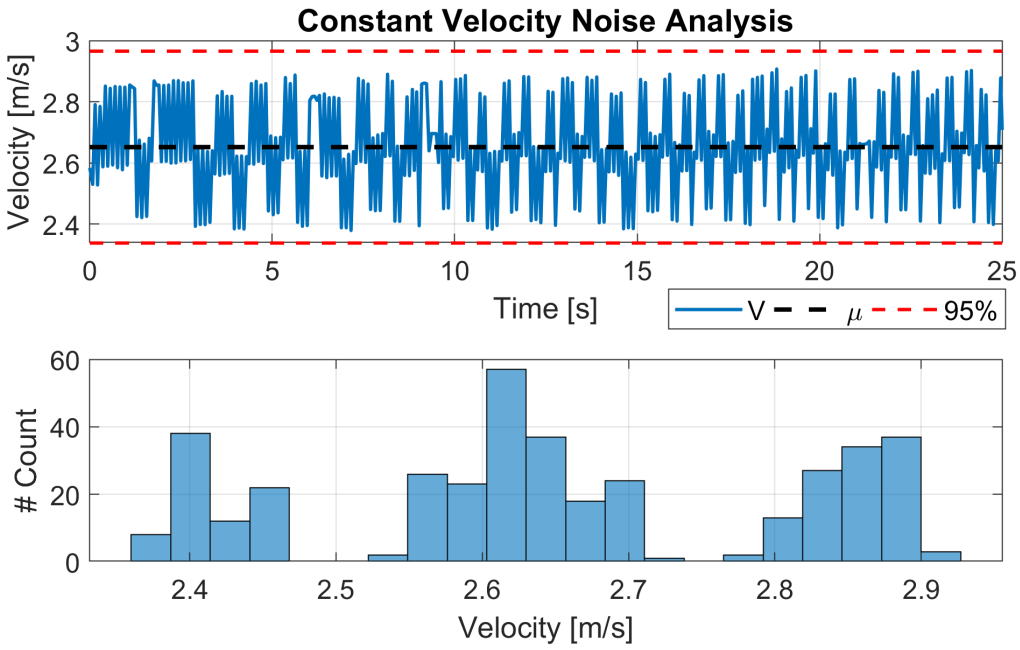


Figure A-2: This figure gives the noise analysis of the velocity measured by the rotary encoder of the vehicle-platform. The first plot shows the measured velocity together with the average velocity $\mu = 2.65[m/s]$ together with its 95% bounds ($Z = 1.96$) based on a standard deviation of $\sigma = 0.16[m/s]$

A-2 Vehicle Sensor Noise Measurements

The vehicle-platform has two different sensors. One for velocity, based on a rotary encode (see appendix A-2) and a one for the vehicle's steering angle δ_f.

The velocity measured at full speed, that is PWM=240[2] for roughly 25 seconds (384 measurements). The results can be seen in figure A-2.
When looking it this figure it is interesting to note that the noise is clearly not normally distributed. This may be problematic when filtering since Kalman based filters work best when the noise is (near-)normally distributed.
As one can see there are three distinct measurement groups. This can be directly related to the way the velocity is measured in the Arduino code and the used hardware. Namely using:

```
Speed = pulseIn(Speed_pin, LOW, 40000);
```

This code measures the time it take for an individual 'wing' of the encoder to pass by, which is then converted to velocity. As there are three different wings on the encoder it can be concluded that the curve-length of the encoder disk passing through the sensor are unequal.

The steering angle of the vehicle is encoded using a potentiometer. This potentiometer is supplied 5[v] by the Arduino situated on the vehicle-platform. One of the Arduino's analogue

[2]PWM is limited to 240/255 since the H-bridge of the motor is under-dimensioned

ports are then used to measure the resistance over the supply and the centre tab of the potentiometer. As most potentiometers can rotate up to a maximum of 360[°] and their response curve can be assumed to be linear[3] their change in resistance can be converted to a rotation in degrees.

To test the accuracy of the sensor the vehicle was driven straight, as in no steering control input, on a smooth surface. To test the repeatability at certain intervals a steering input was given to see how the system would return to normal after which the vehicle-platform was made to drive straight again. The results, as displayed in figure A-3, seem to indicate that the system does not have a single rest state. This implies that using a single bias to correct for offsets is insufficient.

A-3 GNSS noise analysis

Determining the GNSS noise is rather difficult as it is highly dependent on the amount of visible satellites, their elevation, the local surrounding area and atmospheric and tropospheric delays. However, the '$GGA' NMEA message the GNSS receiver contains a measure of 2D accuracy, namely Horizontal Dilation of Precision (HDOP). The relation between this parameter and the noise on the GNSS position is:

$$HDOP = \sqrt{\sigma_x^2 + \sigma_y^2} \tag{A-7}$$

This parameter can be used to give an estimate for the position noise by assuming that $\sigma_x \approx \sigma_y$:

$$\sigma_{xy} = HDOP \tag{A-8}$$

This allows the noise matrices of the Unscented Kalman Filter (UKF) to be updated dynamically with accurate noise estimates. When looking at figure A-5 it becomes clear however, that the assumption of equal standard-deviation for x and y direction clearly doesn't hold. Still, the chosen approach is, in the opinion of the author, the best available.

Characterising the noise on the GNSS velocity and heading measurements is more difficult and has not been attempted. For future research a relation between HDOP and the provided velocity and heading may be useful.

[3]This is in no way not guaranteed.

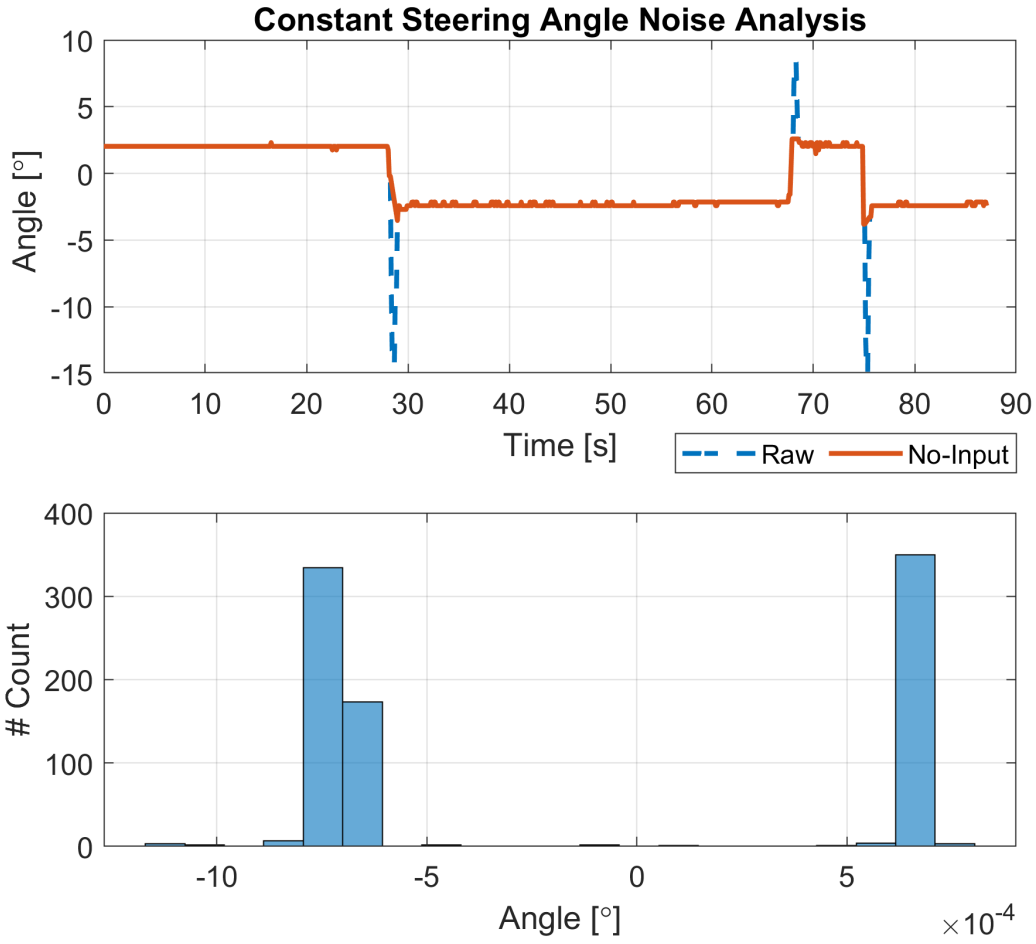


Figure A-3: This plot displays the steering angle noise measurements. The first plot displays the steering angle. The blue dashed line denotes the raw data and the red solid line denotes the data without the steering inputs. The second plot displays the histogram of the measured data without the steering inputs. The data has a mean of $\mu = 0.01[rad]$ and a standard-deviation of $\sigma = 0.038[rad]$.

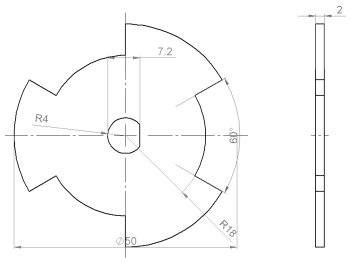


Figure A-4: A simple encoder disk designed to function as an interrupter for the phototransistor, measurements are in [mm]

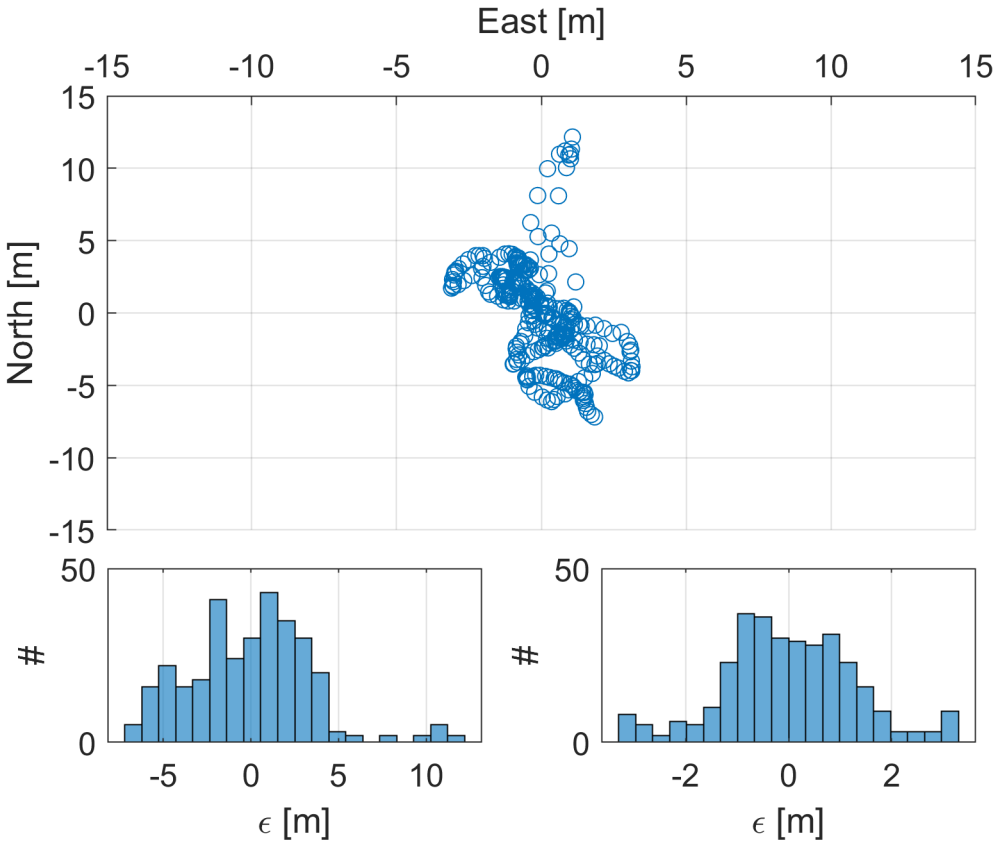


Figure A-5: This figure displays an example of GNSS position noise measured in open-field conditions. **Top**:Displays all measured GNSS points. **Bottom-left**: Displays the histogram of the GNSS noise in global North direction. **Bottom-right**: Displays the histogram of the GNSS noise in the global East direction. Note that μ_x and μ_y are both zero and $\sigma_x = 3.51[m]$, $\sigma_y = 1.28[m]$

Appendix B

Measurement Data

This appendix contains additional measurement results. The first section,appendix B-2 contains plots of additional datasets concerning the IMU orientation filter evaluated using Matlab and the second section contains RMS plot of simulations performed in chapter 5.

B-1 Sensor data-rate evaluation

This section contains result from the data-rate evaluation of both the Inertial Measurement Unit (IMU) as well as the Vehicle Sensors (VS) over CAN-BUS.

B-2 Additional measurement data from IMU9250

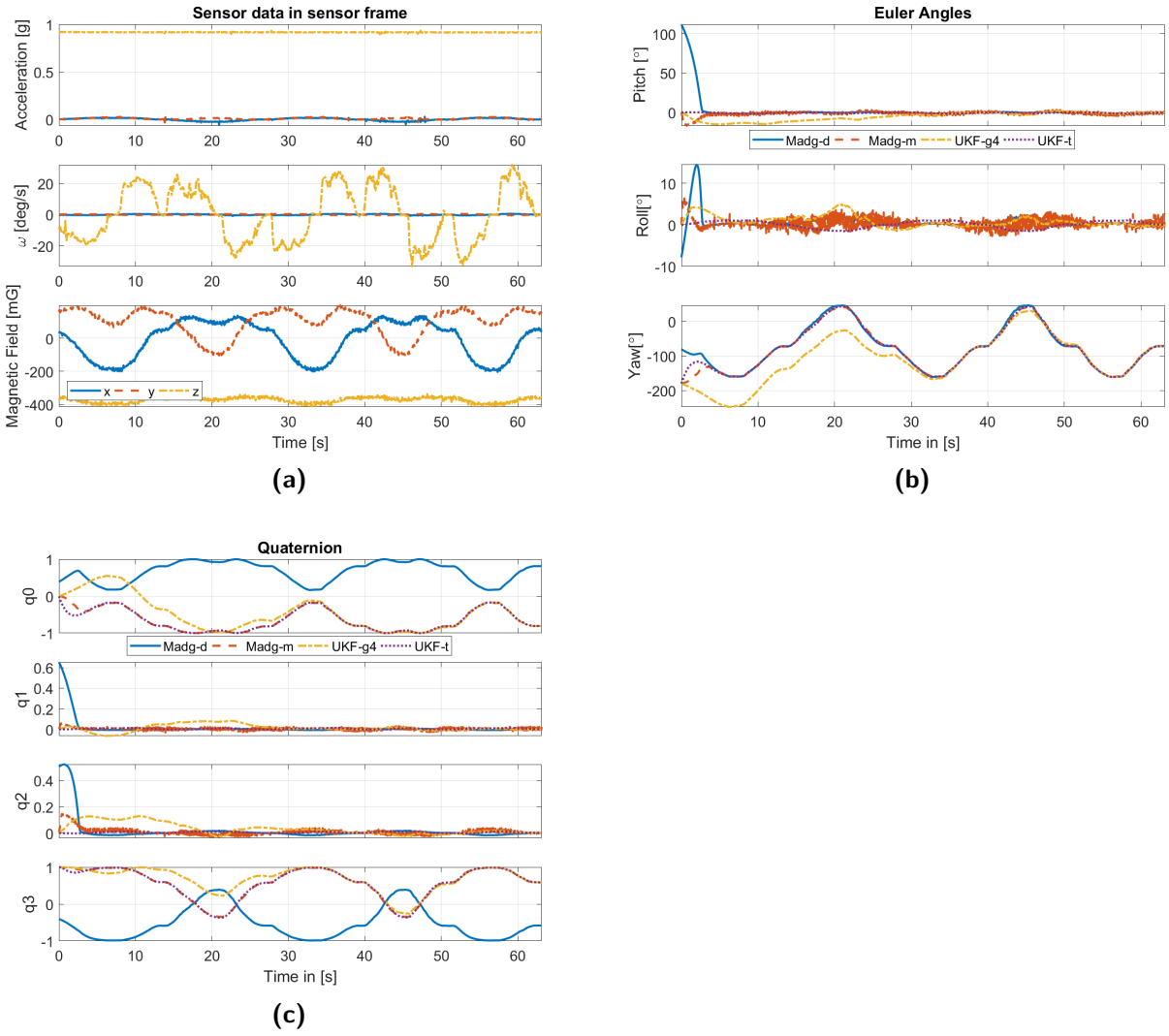


Figure B-1: The Results for the YAW01 IMU results for periodic 90° rotations around the z-axis.

B-2-1 Dataset YAW02

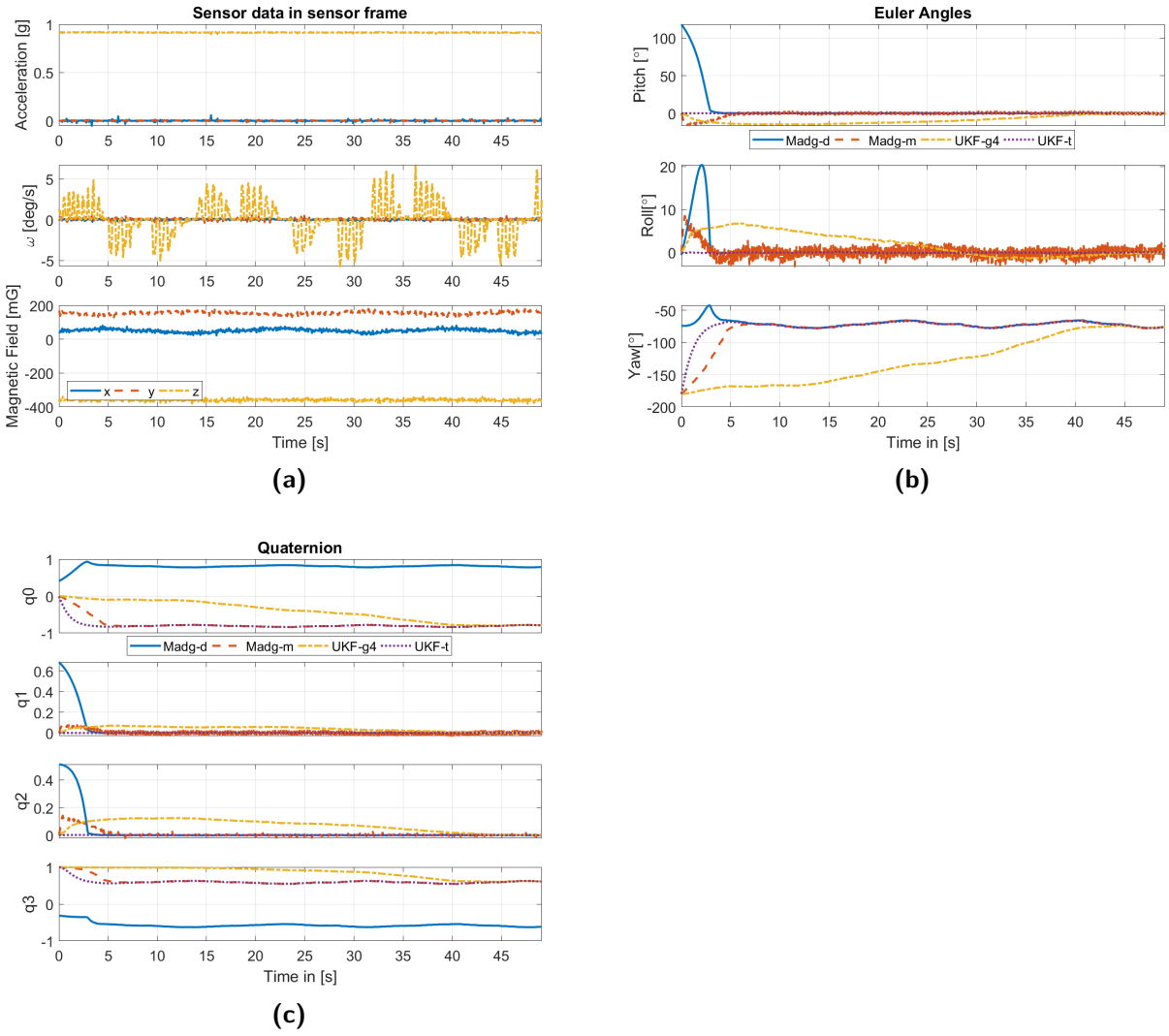
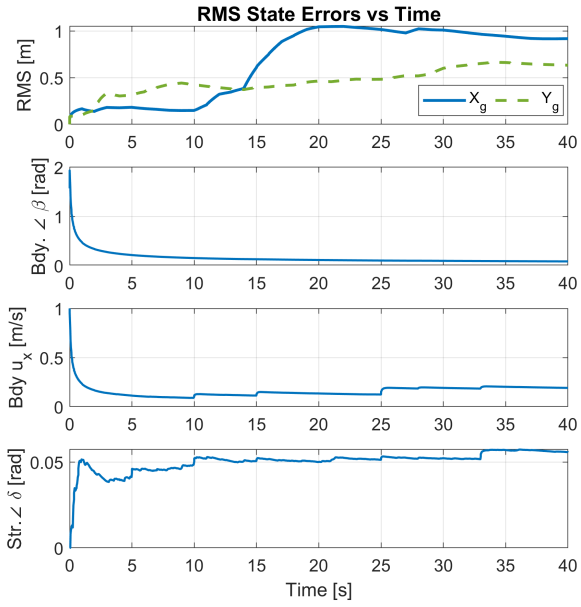
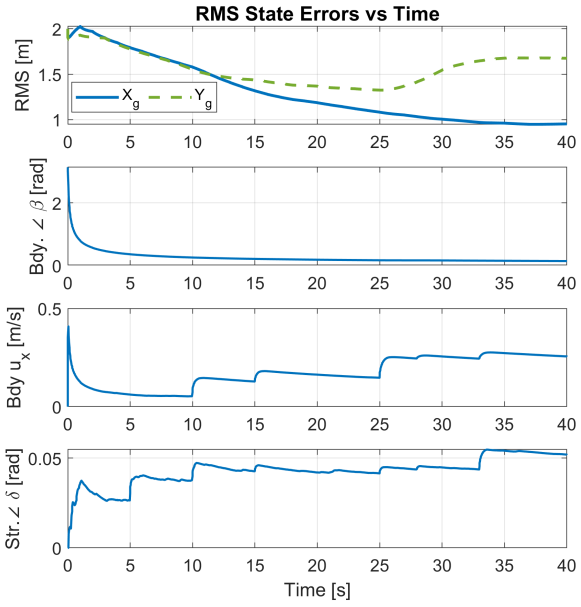


Figure B-2: The Results for the YAW03 IMU results for periodic 5° rotations around the z-axis.

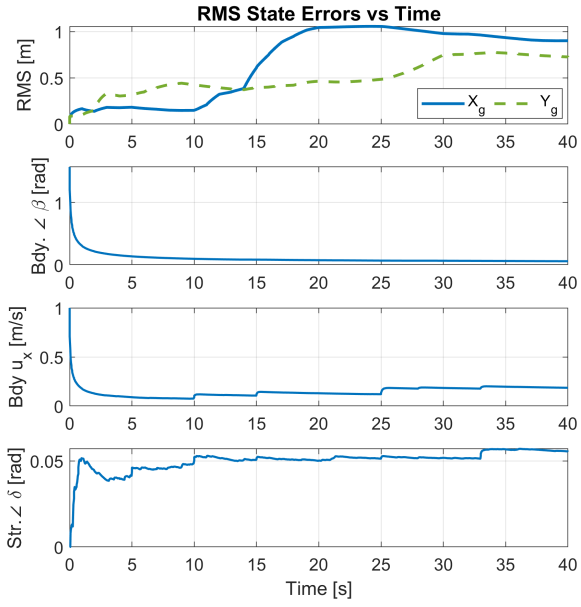
B-3 Multi-Rate Kalman Simulations



(a)



(b)



(c)

Figure B-3: These results show the state estimate RMS error for the simulations 2,3,4 (Top Left, Top Right, Bottom)

Appendix C

Hardware Recommendations

In this chapter the hardware recommendations to improve the pose estimation system will be discussed.

RTC for IMU & CANBUS

Currently the Inertial Measurement Unit (IMU) does not have a time reference. To provide optimal synchronisation of data it would be useful to add a Real-Time Clock (RTC) to the Arduino/ESP32 that would allow the RS232 packets to be sent with a UTC timestamp.

The same holds true for the CANBUS. The used CAN-BUS chip does not support external timekeeping hardware. Therefore, the time measurement of incoming data-frames are dependent on the speed of the software reading the CAN-BUS; this results in time errors.

Change of GNSS Receiver

Even though the ublox Neo m8p is a high end receiver, it does not support Galileo and has a relatively limited amount of channels (28). For future research it would be advisable to include a receiver with a higher amount of channels as well as support for Galileo.

Incorporate an Internet connection

As mentioned in chapter 7 it is advisable to implement a form of Single Frequency Precise Point Positioning (SF-PPP). For SF-PPP accurate Global Navigation Satellite System (GNSS) ephimeris and almanac data is required. This information can be obtained on-line and it is thus recommended to include an internet connection in the system.

Include Wheel Sensors

To improve the dead-reckoning aspect of the vehicle-platform it is required to allow for the measurement of rotational velocities in each wheel. These extra sensors will also allow the inclusion of wheel slip estimation as expected wheel velocities with respect to the motor RPM can be modelled and thus slip angles can be estimated (using Unscented Kalman Filter (UKF), Extended Kalman Filter (EKF) or equivalent state-estimators).

Vehicle-Platform Improvements

The vehicle-platform provided by Accenda is certainly functional, however a large amount of things can be improved:

- Mount the IMU on Dampers
  The IMU is now more or less directly connected to the frame of the platform. This results in high frequency noise being measured by the IMU. It would be beneficial if the IMU could be mounted on some dampers to cancel out high frequency noise.
- Steering Angle Measurement
  Although the measurement of the steering angle with a potentiometer is functional, it might be better to use a digital equivalent to reduce the measurement noise.
- Transfer of Servo to Wheels
  The mechanical link between the wiper motor and the steering-linkage is somewhat rickety. If the wheel is locked and the motor tries to rotate the wheels the sheet metal connection bends. It is therefore recommended to change the sheet metal link to something with a higher stiffness to remedy this.
- Mounting point for the Raspberry Pi
  It would be nice if the Raspberry PI can be mounted on the platform itself.
- Change the Rotary Encoder
  Each rotary encoder is now roughly 60° this should be reduced, therefore a design with more 'fins' should be chosen.
- Change the Remote and Remote Receiver
  Currently the signals from the Remote require 20 ms to measure per signal. This can be vastly improved by implementing say a bluetooth based controller and receiver (since the vehicle will not be operated a large distanes from the remote bluetooth's range should be fine).
- Integrate IMU and Arduino
  Currently the Arduino in the platform is responsible for the motor control and vehicle measurements. It is recommended to swap the Arduino for an ESP32 this would allow a single board to handle both the driving functions, Vehicle Sensors (VS) measurements as well as the IMU measurements.
  With this improvement the IMU data can be sent over CAN-BUS without the need for another transceiver.

- Remove End-stop switches
  Currently the PCB provided by Accenda reserves two pins for end switches of the steering mechanism. These can be removed and replaced by a software limiter. At the very least one can be removed and the switches can be put in parallel.

- Battery Measurement
  Measuring the battery is important especially for long distance travel, since the platform is rather heavy one wants to avoid carrying it back to the 'mothership'.

Bibliography

[1] A. Reichental, "Ces reflections: The future of autonomous cars," *Forbes*, 2018.

[2] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pp. 763–770, IEEE, 2013.

[3] D. G. Paul, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, 2008.

[4] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[5] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte, "A high integrity imu/gps navigation loop for autonomous land vehicle applications," *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 572–578, Jun 1999.

[6] J. Z. Sasiadek, Q. Wang, and M. B. Zeremba, "Fuzzy adaptive kalman filtering for ins/gps data fusion," in *Proceedings of the 2000 IEEE International Symposium on Intelligent Control. Held jointly with the 8th IEEE Mediterranean Conference on Control and Automation (Cat. No.00CH37147)*, pp. 181–186, 2000.

[7] H. Qi and J. B. Moore, "Direct kalman filtering approach for gps/ins integration," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, pp. 687–693, Apr 2002.

[8] J. Zhou, S. Knedlik, and O. Loffeld, "Ins/gps tightly-coupled integration using adaptive unscented particle filter," *The Journal of Navigation*, vol. 63, no. 3, pp. 491–511, 2010.

[9] A. Mohamed and K. Schwarz, "Adaptive kalman filtering for ins/gps," *Journal of geodesy*, vol. 73, no. 4, pp. 193–203, 1999.

[10] S. Rezaei and R. Sengupta, "Kalman filter-based integration of dgps and vehicle sensors for localization," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 6, pp. 1080–1088, 2007.

[11] S. F. Schmidt, "Application of state-space methods to navigation problems," in *Advances in control systems*, vol. 3, pp. 293–340, Elsevier, 1966.

[12] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, vol. 3068, pp. 182–194, International Society for Optics and Photonics, 1997.

[13] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on automatic control*, vol. 45, no. 3, pp. 477–482, 2000.

[14] S. J. Julier, "The scaled unscented transformation," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 6, pp. 4555–4559, IEEE, 2002.

[15] C. Melendez-Pastor, R. Ruiz-Gonzalez, and J. Gomez-Gil, "A data fusion system of gnss data and on-vehicle sensors data for improving car positioning precision in urban environments," *Expert Systems with Applications*, vol. 80, pp. 28 – 38, 2017.

[16] Y. Wang, J. Mangnus, D. Kostić, H. Nijmeijer, and S. T. H. Jansen, "Vehicle state estimation using gps/imu integration," in *2011 IEEE SENSORS Proceedings*, pp. 1815–1818, Oct 2011.

[17] L. Armesto, S. Chroust, M. Vincze, and J. Tornero, "Multi-rate fusion with vision and inertial sensors," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 1, pp. 193–199 Vol.1, April 2004.

[18] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1453–1464, Sept 2004.

[19] S. Kluge, K. Reif, and M. Brokate, "Stochastic stability of the extended kalman filter with intermittent observations," *IEEE Transactions on Automatic Control*, vol. 55, no. 2, pp. 514–518, 2010.

[20] L. Li and Y. Xia, "Stochastic stability of the unscented kalman filter with intermittent observations," *Automatica*, vol. 48, no. 5, pp. 978–981, 2012.

[21] M. Kok, J. D. Hol, and T. B. Schön, "Using inertial sensors for position and orientation estimation," *arXiv preprint arXiv:1704.06053*, 2017.

[22] InvenSenseInc, *MPU-9250 Product Specification Revision 1.0*.

[23] Drotek, "Tiny rtk : The world smallest precision gps u-blox neo-m8p." https://drotek.com/en/tiny-rtk-u-blox-neo-m8p/, 2016.

[24] Tallysman, "Tw2410/tw2412." website: https://www.tallysman.com/index.php/gnss/products/antennas-gpsglonass/tw2410-tw2412/, 2019.

[25] Arduino, "Arduino uno rev3." https://store.arduino.cc/arduino-uno-rev3, 2018.

[26] Raspberry PI Foundation, "Specifications of the raspberry pi 3b+." website: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, 2018.

[27] SparkFun, "Sparkfun mpu-9250 breakout arduino library." website: https://github.com/sparkfun/SparkFun_MPU-9250_Breakout_Arduino_Library, 2018.

[28] National-Instruments, "Can physical layer and termination guide." website: http://www.ni.com/white-paper/9759/en/, 2018.

[29] M. di Natale, "Understanding and using the controller area network," 2008. Part of a course on the Design of Embedded Systems at Berkeley University.

[30] EsspressifCommunity, "Esspressif arduino library." website: https://github.com/espressif/arduino-esp32, 2018.

[31] K. Winer, "Mpu9250 filters." website: https://github.com/kriswiner/MPU9250, 2014.

[32] IEEE Computer Society, "Ieee standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, Aug 2008.

[33] W. R. Hamilton, "Ii. on quaternions; or on a new system of imaginaries in algebra," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 25, no. 163, pp. 10–13, 1844.

[34] J. B. Kuipers, *Quaternions and rotation sequences*, vol. 66. Princeton university press Princeton, 1999.

[35] S. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," *Report x-io and University of Bristol (UK)*, vol. 25, pp. 113–118, 2010.

[36] F. L. Markley and D. Mortari, "Quaternion attitude estimation using vector observations.," *Journal of the Astronautical Sciences*, vol. 48, no. 2, pp. 359–380, 2000.

[37] R. Mahony, T. Hamel, and J. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on Automatic Control*, vol. 53, pp. 1203–1218, June 2008.

[38] X. Yun and E. R. Bachmann, "Design, implementation, and experimental results of a quaternion-based kalman filter for human body motion tracking," *IEEE Transactions on Robotics*, vol. 22, pp. 1216–1227, Dec 2006.

[39] X. Yun, E. R. Bachmann, and R. B. McGhee, "A simplified quaternion-based algorithm for orientation estimation from earth gravity and magnetic field measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, pp. 638–650, March 2008.

[40] R. G. Valenti, I. Dryanovski, and J. Xiao, "A linear kalman filter for marg orientation estimation using the algebraic quaternion algorithm," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 2, pp. 467–481, 2016.

[41] K. Feng, J. Li, X. Zhang, C. Shen, Y. Bi, T. Zheng, and J. Liu, "A new quaternion-based kalman filter for real-time attitude estimation using the two-step geometrically-intuitive correction algorithm," *Sensors*, vol. 17, no. 9, p. 2146, 2017.

[42] H. G. De Marina, F. J. Pereda, J. M. Giron-Sierra, and F. Espinosa, "Uav attitude estimation using unscented kalman filter and triad," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4465–4474, 2012.

[43] G. Wahba, "A least squares estimate of satellite attitude," *SIAM review*, vol. 7, no. 3, pp. 409–409, 1965.

[44] M. Shuster, "Approximate algorithms for fast optimal attitude computation," in *Guidance and Control Conference*, p. 1249, 1978.

[45] M. D. Shuster and S. D. Oh, "Three-axis attitude determination from vector observations," *Journal of Guidance, Control, and Dynamics*, vol. 4, no. 1, pp. 70–77, 1981.

[46] F. L. Markley and D. Mortari, "How to estimate attitude from vector observations," *NASA TechDoc*, 1999.

[47] H. D. Black, "A passive system for determining the attitude of a satellite," *AIAA journal*, vol. 2, no. 7, pp. 1350–1351, 1964.

[48] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA journal of numerical analysis*, vol. 8, no. 1, pp. 141–148, 1988.

[49] E. Kraft, "A quaternion-based unscented kalman filter for orientation tracking," in *Proceedings of the Sixth International Conference of Information Fusion*, vol. 1, pp. 47–54, 2003.

[50] S. Haykin, *Kalman filtering and neural networks*, vol. 47. John Wiley & Sons, 2004.

[51] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pp. 153–158, Ieee, 2000.

[52] A.-L. Cholesky, "Sur la résolution numérique des systèmes d'équations linéaires," *Bulletin de la Sabix. Société des amis de la Bibliothèque et de l'Histoire de l'École polytechnique*, no. 39, pp. 81–95, 2005. This is a republishing of the original source.

[53] G. Lapouge, J. Troccaz, and P. Poignet, "Multi-rate unscented kalman filtering for pose and curvature estimation in 3d ultrasound-guided needle steering," *Control Engineering Practice*, vol. 80, pp. 116–124, 2018.

[54] K. Winer, "Simple and effective magnetometer calibration." website: https://github.com/kriswiner/MPU6050/wiki/Simple-and-Effective-Magnetometer-Calibration, 2017.

Glossary

List of Acronyms

CKF	'Classic' Kalman Filter
DCM	Direction Cosine Matrix
DMP	Digital Motion Processor
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
GPS	Global Positioning System
GNSS	Global Navigation Satellite System
HDOP	Horizontal Dilation of Precision
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
MPU	Motion Processing Unit
NED	North, East, Down
PWM	Pulse Width Modulation
QUEST	Quaternion Estimator
UKF	Unscented Kalman Filter
RTC	Real-Time Clock
SF-PPP	Single Frequency Precise Point Positioning
TRIAD	Three Dimensional Attitude Estimation

VCM Variance Covariance Matrix

VS Vehicle Sensors

Nomenclature

Scalars		
α	UKF parameter, sets the spread of the Sigma-Points	[-]
β	Describes the vehicle-platform body angle wrt global north	[rad]
β	UKF scaling parameter used to set the expected distribution	[-]
δ_f	Steering angle	[rad]
γ	UKF sigma-point scaling factor	[-]
κ	UKF scaling parameter	[-]
λ	UKF scaling parameter	[-]
f_Δ	Inter frame time	[s]
u_x	Body velocity	[m/s]
Vectors		
$\hat{\mathbf{x}}^-$	Process State estimate	
$\hat{\mathbf{x}}_k$	State estimate	
$\hat{\mathbf{y}}_k^-$	Measurement estimate through state estimate propagation	
$\mathbf{a}$	Acceleration measurement vector	[g]
$\mathbf{m}$	Magnetism measurement vector	$[\mu T]$
$\mathbf{p}$	Pose, defined as a combination of rotation and translation	$\in \mathbb{R}^3 \cdot SO(3)$
$\mathbf{q}$	A unit quaternion $\lvert\mathbf{q}\rvert = 1$	
$\mathbf{v}_{i,k}$	Measurement noise vector	
$\mathbf{w}_k$	Process noise vector	
$\mathbf{y}_k$	Measurement	
$\mathbf{z}$	Measurement vector	
ω	Angular rates, used for gyroscope measurements $(\omega_x, \omega_y, \omega_z)$	$[rad/s]$
$W^{(c)}$	UKF VCM weights	
$W^{(m)}$	UKF state update weights	
Matrices		
$\mathbf{\Omega}$	Angular rate matrix	$[rad/s]$

$\mathbf{J}$	Jacobian derivative matrix	
$\mathbf{P}_k$	VCM for the measurement update	
$\mathbf{P}_k^-$	VCM for the process update	
$\mathbf{P}_{xy}$	Estimate vs. Measurement Cross-Covariance matrix	
$\mathbf{P}_{yy}$	Measurement Auto-Covariance matrix	
$\mathbf{Q}$	Quaternion based rotation matrix	
$\mathbf{R}^n$	Measurement noise matrix	
$\mathbf{R}^v$	Process noise matrix	
$\mathbf{R}_{x/y/z}$	Rotation matrix	
$\mathcal{K}_k$	Kalman Gain matrix	
Sets		
$\mathcal{X}$	Set of sigmapoints (Process update)	
$\mathcal{Y}$	Set of sigmapoints (Process update)	
Miscellaneous		
$\mathcal{F}_b$	Vehicle-Platform body frame	[–]
$\mathcal{F}_g$	Global, left handed frame defined as North, East, Down	[–]
$\mathcal{F}_s$	IMU sensor frame	[–]
Super and Subscripts		
$\bigcirc^-$	Process update	
$\bigcirc_k$	Discrete time instance	
${}^B\bigcirc$	Vehicle-platform body frame	
${}^E\bigcirc$	Earth frame in NED	
${}^S\bigcirc$	Sensor frame of the MPU9250	
${}^E_S\bigcirc$	Relative definition from Earth to Sensor frame	