Wilhelm Schickard-Institut für Informatik

Diploma Thesis

Fusion of Sensory and Motor Egomotion-Information for UAVs

by

Tobias Beck

Advisors:

Prof. Dr. rer. nat. Hanspeter A. Mallot

Chair in Cognitive Neurosciences

Prof. Dr. rer. nat. Andreas Zell

Chair in Computer Architecture

Registration Date:	February	4th 2009
Due Date:	November	18th 2009

Eberhard Karls Universität Tübingen

This thesis is a presentation of original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

I hereby declare and confirm that this thesis is entirely the result of my own work except where otherwise indicated.

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Tobias Beck
Tübingen, November 17, 2009

Thesis advisors
Prof. Dr. rer. nat. Hanspeter A. Mallot
Prof. Dr. rer. nat. Andreas Zell

Author
Tobias Beck

Fusion of Sensory and Motor Egomotion-Information for UAVs

Abstract

Robots permeate ever more areas of our lives. From entertainment uses to military applications, more people previously not familiar with these complex machines come into contact with robots or other autonomous vehicles. To simplify their control, additional navigational and other auxiliary systems are developed. In order to facilitate navigational support for the controller by the robot, the robot needs to be aware of its own motion and the consequences of its actions in its environment. This thesis focuses on the possibility of achieving an improved estimation of a robot's egomotion by employing several sensors, especially inertial, gyroscopic and optical sensors, in combination with a recursive state estimator.

A recursive nonlinear continuous-discrete-time formulation of such an estimator is derived step by step, presenting existing solutions from the original Kalman filter to the eventual continuous-discrete square root Unscented Kalman filter with sequential updates. The necessary process and observation models are derived. They represent dynamic nonlinear functions describing the evolution of the system state over time as well as expected sensory observations with respect to a given state estimate. The model parameters are acquired empirically.

In state estimation experiments, it is shown that the estimation algorithm can approximate the attitude and angular velocity of the UAV successfully. However, the choice of sensors denies the correct estimation of the linear velocity and location, because no sensor delivers absolute measurements on their effects.

Acknowledgments

This work was created as part of the graduation requirements for the diploma degree in the course of studies of Bioinformatics at the Eberhard Karls Universität Tübingen. The diploma thesis was developed at the robotics lab of the cognitive neurosciences chair under Prof. Dr. Hanspeter Mallot. My thanks go to him and Prof. Dr. Andreas Zell at the chair of computer architecture at the Wilhelm-Schickard-Insitut für Informatik at the Universität Tübingen, for offering and supervising this interesting and challenging task in a combined effort.

In this context I would like to thank the following people who have all contributed in some form to this endeavor:
My colleagues at the robotics lab for their intellectual support, and for offering a gratifying and stimulating research and working environment. In particular Fabian Recktenwald for his supportive advice and insightful discussions. Special thanks go out to Dr. Hansjürgen Dahmen for his time, ideas and support, as well as for sharing his mathematical expertise and wisdom.

I offer my thanks to my proof readers for lending their time to my work and offering constructive criticism. Your help is greatly appreciated.

Finally, my family for their love and support. Bianca, Matt, Otto, Mami, Omi & Opa: Thank you for always believing in me.

Contents

List of Algorithms

Chapter 1

Introduction

1.1 Overview

Robots are permeating ever more areas of our lives. In our homes, domestic appliances clean our floors autonomously[1] while children play with robots dancing a programmed routine and older generations design complex self-built and -programmed LEGO robots[2]. In branches of our governments such as police and firefighters, robots are used to scout dangerous territory, while in the military mobile attack drones reduce the risk of human casualties[3]. Figure 1.1 shows some examples for drones. Established medical applications of robots are currently non-autonomous tools for improved surgical intervention[4, 5, 6] or artificial limbs[7], but the vision of nanobots circulating through our bodies looms closer as these robots become smaller[8].

Robots' areas of application are as numerous as the problems they are supposed to solve. The examples mentioned above only represent a small part of the exciting role robots are playing and are going to play in our world. Current leading scientific research focuses on mobile robots able to execute missions in unknown environments with as little human control as necessary or possible, as can be seen in the current and future Mars missions[9, 10]. Developing autonomous mobile robots includes challenges in various disciplines, from navigation and route planning over obstacle avoidance to mission control and the choice of sensors and actuators needed for each task. Challenges such as the DARPA Urban Challenge[11] reflect the scientific community's current state of the art while pushing for new advancements in the field.

Mobile robots can be distinguished into three types based on their locomotion: sea-, air- and land-based, with land-based robots having been researched the most, since sea and air navigating robots face more restrictions and higher requirements, e.g. a land-based robot can carry heavier equipment and stay stationary more easily than a flying robot. The lower requirements allowed a larger research community an easier and earlier start to studying land-based mobile robots. As challenges of land-based mobile robots are resolved to a certain extent and required technologies become affordable, sea- and air-based robots can become the focus of wide-spread research.

(a) LEGO Mindstorms NXT

(b) Hermes 450 Civil Registration UAV

(c) MQ-8B Fire Scout

(d) ScanEagle with Catapult Launcher

Figure 1.1: UAV Examples.

This thesis treats air-based navigation by concentrating on a certain type of flying mobile robot, or Unmanned Aerial Vehicle: a small, low-altitude quadrocopter with a maximum payload of 0.3kg. This section will present an overview of this work's main topics along with a short introduction to each chapter.

1.1.1 Notation Conventions

A common understanding of the symbols used in this work is helpful to better comprehend the descriptions and formulae given. Throughout the thesis, mainly continuous dynamic systems with multi-dimensional system states are dealt with. In a few cases, to lay the groundwork for continuous systems, discrete and linear systems are regarded. Here time variables such as t, t_0, t_k are integers representing multiples of the unity time interval set as discrete time step. Vectors will be denoted as small bold-face letters such as $\mathbf{x}$, $\mathbf{z}$ or $\mathbf{x_k}$, a vector at a certain point in time t_k. Matrices will be denoted by capital bold-face letters $\mathbf{Q}$, $\mathbf{R}$ and $\mathbf{V}_k$ indexed analogously to vectors depicting a matrix $\mathbf{V}$ at time t_k.

t, t_k	Time in general; present time
t_0	Time at which observations start
$\mathbf{x}$,$\mathbf{x}(t)$,$\mathbf{x}_k$	Basic random variable, actual system state
$\mathbf{z}$,$\mathbf{z}(t)$,$\mathbf{z}_k$	Observed random variable, measurement
$\hat{\mathbf{z}}_k$	Expected measurement
$\tilde{\mathbf{z}}_k$	Innovation, difference of real and expected measurement
$\hat{\mathbf{x}}_k^-$	Optimal estimate of $\mathbf{x}_k$ given $\mathbf{z}_{0:k-1} = \{\mathbf{z}_0, \ldots, \mathbf{z}_{k-1}\}$
$\hat{\mathbf{x}}_k$	Optimal estimate of $\mathbf{x}_k$ given $\mathbf{z}_{0:k} = \{\mathbf{z}_0, \ldots, \mathbf{z}_k\}$
$p(\mathbf{x})$	Probability distribution of $\mathbf{x}$
$p(\mathbf{x}\|\mathbf{z})$	Conditional probability of $\mathbf{x}$ being true if $\mathbf{z}$ was seen
$p(\mathbf{x}_k\|\mathbf{z}_k, \mathbf{x}_{k-1})$	Probability of $\mathbf{x}_k$ being true if $\mathbf{z}_k$ was seen and $\mathbf{x}_{k-1}$ were true

Table 1.1: Optimal Estimates.

1.1.2 Unmanned Aerial Vehicles

A remotely controlled aircraft is called an Unmanned Aerial Vehicle, or UAV. Note that UAVs need not be autonomous, only the absence of a human controller or pilot within the vehicle determines this definition. Minimization of the payload required by air-based mobile robots results in diverse solutions, e.g. gas-filled blimps, rocket-propelled planes, or helicopters, each with its own advantages and disadvantages. Blimps do not have good mobility, as compared to helicopters, but their advantage is their independence from having to run rotors to stay in the air. Rocket-propelled planes need high speeds to avoid crashing down, making them appropriate for high altitude missions and certain low altitude missions spanning large coverage areas. Helicopters embody a highly mobile vehicle for low altitude missions in cluttered terrains. The slower velocity allows for more mobility, with an emphasis on the ability to stay stationary and hover, but limits the area one can cover before energy runs out. As with all air-based mobile robots, the craft's size restrains its maximum payload and a compromise between air time and features on board needs to be struck.

As noted above, this work focuses on quadrocopters. A quadrocopter is a type of helicopter with four horizontal and no vertical rotors. It is a small drone for low altitude missions in crowded terrain, the specific model this thesis covers is the AirRobot AR-100 (Figure 1.2). The AR-100 hardware and features are described in more detail in section 3.3.

The advantages of quadrocopters, stable and stationary flight, good maneuverability and their small size allow them to be used in cluttered terrain, without fear of unintentionally hurting casual bystanders. Unfortunately, successfully piloting a quadrocopter still demands special training, above all if the machine is equipped with expensive hardware and used in

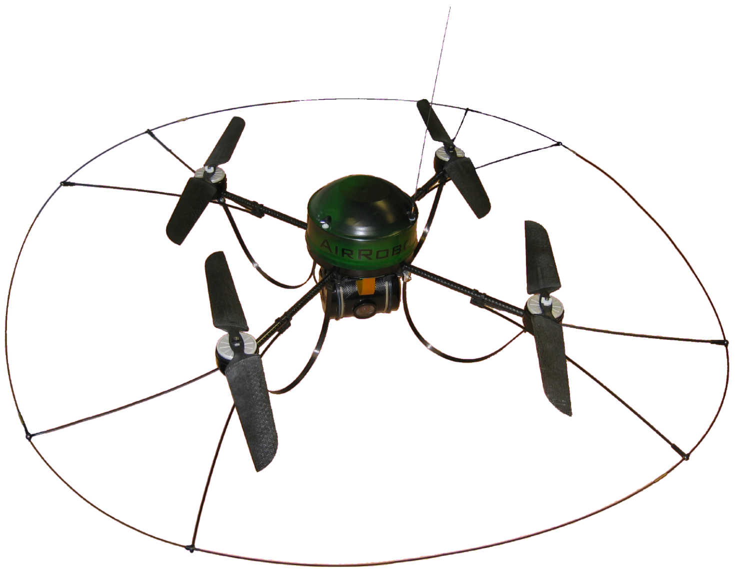


Figure 1.2: AirRobot AR-100.

difficult mission scenarios. In current real life scenarios these machines are usually controlled by trained personnel, but industry and military, turning to science, want to remove the difficult piloting from the controller's worries and allow him to concentrate on his actual assignment, letting the UAV assume control to a certain extent concerning locomotion and navigation.

1.1.3 Egomotion

For navigational purposes, mobile robots need to be aware of where they are and how fast they are going. An easy solution to a part of this problem can be achieved by including GPS support (*Global Positioning System*) and using absolute GPS coordinates to navigate. While this is comparatively easy to implement, it offers only rudimentary navigation throughout the world. GPS only works where its signals can be received and used effectively, negating its use for navigating indoors or in complex crowded locations where too many reflections and interferences degrade the signal. Complex navigational objectives such as obstacle avoidance or object tracking need additional or more sophisticated methods.

Another solution is Dead Reckoning. Dead Reckoning is the method of computing one's location by the change of position from a given initial location. In other words navigation by integrating information about one's own movement. Egomotion is the motion of the robot within its environment, as seen by the robot itself. This poses the question of how to calculate one's own egomotion. Land-based robots can compute their egomotion e.g. by monitoring the turning of their wheels. But flying robots cannot determine their total velocity via their rotors' speeds as easily as that, because they move within a highly dynamic carrier system (air). Wind is a strong external force acting on these vehicles, therefore other sensors must be considered to calculate the robot's egomotion.

Egomotion is necessary for many navigational tasks and can be calculated using e.g. inertial or visual sensor measurements. Having this information about the robot's motion is sufficient for navigation by Dead Reckoning, but incommensurate for more complex tasks, e.g. obstacle avoidance, where motion of other objects is needed. How egomotion is computed and estimated is covered in Chapter 3. More examples of what kinds of sensors exist and are used are covered in the next Subsection 1.1.4.

1.1.4 Perception

In order to navigate through an unknown dynamic world, a robot needs sensors to perceive information about its environment, since it cannot access the world's system state directly as in some simulations. There are different types of sensing modalities, as for example smell, touch, vision or sound. Some examples for technical sensors are listed in Table 1.2. Which sensors to use in a robot is a design choice: some might work better in

Type of Sensor	Application
Inertial and Gyroscopic	Egomotion
Camera	Egomotion
Compass	Orientation
Barometer	Height
Thermometer	Temperature
Laser	Distance To Objects
Ultrasound	Distance To Close Objects
GPS	Position

Table 1.2: Sensor Types.

certain environments, others might fit the planned missions better. Each sensor's characteristics need to be taken into account when designing a new robot and the overall choice must be tailored to its range of duty. In the case of the AR-100, the available sensors are listed below and described in more detail in section 3.3.3:

- IMU (*Inertial Measurement Unit*)
- Compass/Magnetometer
- Camera
- Barometer
- GPS

Since a variable of a physical system cannot be accessed directly, sensors providing measurements about that variable are introduced. These sensors invariably have a certain maximal precision and minimal error-margin as determined by their construction method, leading to noisy or incomplete measurements.

The errors introduced by this indirect access to variables via observations pollute the determination of the current system state, e.g. the current location. The goal is a noise free or at least reduced estimation of the system state. This leads to the next subsection, where a short overview of State Estimation methods is presented.

1.1.5 State Estimation

As mentioned above, most of the time the system's physical state cannot be observed directly, but needs to be determined by way of taking noisy measurements of indirect effects of the state(Figure 1.3), hereby introducing errors. These observations are analyzed to approximate the inaccessible system state. Errors and incomplete measurements complicate the approximation process, therefore methods for system state estimation implementing a calculation of the certainty of a particular state estimate being correct have been developed. During the years of their application, modifications and extensions to these algorithms were devised in estimation theory. Among these are recursive Bayesian estimation and Kalman Filters, which are covered in more detail in Chapter 2.

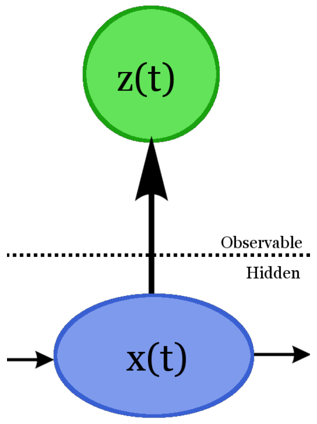


Figure 1.3: Observable Measurement and Hidden State.

Bayesian estimation methods adjust probabilities of system states given new evidence. To achieve this, the hidden variables are predicted for the next time step, when the actual observation is compared with the expected measurement. For prediction, a mathematical process model of the system must be defined, describing the evolution over time from a given state configuration, as introduced in the next section and discussed in detail in Chapter 3. Bayes' theorem describes how to adjust the probability distribution of the system variables after a measurement is received. In other words, it changes our belief in the system variables to accommodate the additional observational information, lending itself well to the predict-correct update cycle. The Bayes' rule defines the factor of the impact the evidence has on the belief in the hypothesis, it is given as

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}.$$

H is the hypothesis, that is one configuration (of many) of the hidden system variables, x in this paper's notation and E represents the evidence or measurement, here denoted as z. $P(H)$ is the *prior probability density function*, $p(x)$, of the state, which is assumed to be known beforehand and $P(E)$ the *marginal probability* known a priori of the observation E (or z) occurring, with

$$P(E) = \sum_H P(E|H)P(H),$$

where the last two terms are the *conditional probability* $P(E|H)$ of encountering observation E if H happens to be true and the *posterior probability (density function)* $P(H|E)$ of H given measurement E or z. Rewritten in the terms of Chapter 2:

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)}. \tag{1.1}$$

For recursive calculation and incorporation of successive measurements, the posterior probability $p(x|z)$ is taken as the prior $p(x)$ in Equation (1.1). Kalman Filters follow this principle and are specialized for Gaussian random distributions.

1.1.6 Physical Model

In order to predict how a mobile robot behaves, a model approximating the robot's physical characteristics and dynamics to a sufficient degree must be formulated. Such a model consists of a system state and of mathematical functions evolving the system state over a certain time span. The detail of a model's desired accuracy controls the level of the robot's physical characteristics that need to be included in the equations of motion. This includes for example mass, drag coefficient, surface area for air drag, rotor coefficients et cetera.

Physical Models can be discrete or continuous, linear or non-linear, differential or absolute. An example for such a discrete, linear absolute representation of a sinusoidal motion is:

$$f(t_{k+1}) = sin(t_{k+1})$$

An example for the same motion but as a continuous and differential process model:

$$\frac{df(t)}{dt} = cos(t)$$

The physical models necessary for state estimation not only need to predict a system state, but also calculate expected observations from such an estimated state prediction. This component of the physical model is called the observation model, and it specific for each sensor used.

Chapter 3 takes a closer look at how the system state is constructed and how the dynamics were modeled in the case of the AR-100.

1.2 Problem Statement

The UAV AR-100 is equipped with a video camera and an IMU as visual and inertial sensors, among others. Their sensor modalities are completely different. Where the camera captures visual cues from which optic flow and subsequently translational direction and rotation information is calculated, the IMU captures linear accelerations and gyroscopic velocities, which is in one case the first derivative of the actual wanted information, linear velocity. These contrasting characteristics allow a combination of the measurements, capitalizing on their advantages and compensating for their disadvantages to receive an improved observation result.

Sensory and motor information of the mobile robot shall be combined intelligently to receive a better estimation of its egomotion than possible with each measurement taken alone. A qualified statistical method for state estimation as well as an appropriate architecture for sensor fusion shall be chosen. Furthermore, the necessity of a physical model of the dynamics of the mobile robot for estimation of the robot's egomotion is to be evaluated.

The goal of this thesis is the implementation of the chosen method for data fusion of inertial and visual measurements and state estimation extending on the already existing control framework in C++. Quantitative tests to evaluate the robustness and performance of the method are also to be conducted. The consequence of the inertial drift on the estimation of egomotion as well as how well the chosen method prevents this drift is to be examined.

1.3 Thesis Outline

The remainder of the thesis is organized as follows:

1.3.1 Main thesis chapters

Chapter 2 gives an introduction to optimal Gaussian approximate recursive Bayesian estimation, presenting the Kalman filter framework and highlighting the flaws of the extended Kalman filter. This reason, why a better solution for Gaussian approximate nonlinear estimation than the EKF is needed, leads into the second and main part of this chapter. It presents the Unscented Kalman filter and several variants thereof incrementally, to finally reach the developed solution implemented as part of this thesis.

Chapter 3 derives the physical models necessary for state estimation. The first part establishes prerequisite theoretical background required for the model formulations. This is followed by a short presentation of the hardware available for this thesis, i.e. the UAV and its base station, before subsequently developping the dynamic nonlinear process and observation models. The physical process model describes the dynamic properties of the UAV, how its state evolves over time. The observation model presents the mapping of the estimated system state to expected observations of said state.

Chapter 4 focuses on the conducted experiments and explains their setup in detail. The experiments examine parameters required for the physical model such as rotor and air drag coefficients. The obtained results are presented and discussed. Experiments concerning the performance of the state estimation algorithm are presented and the implications of their results discussed.

Chapter 5 discusses and summarizes the results of this work, drawing conclusions and indicating possible directions for future research.

1.3.2 Appendices

Some of the material is moved to the Appendices section in order to maintain the presentational flow of this document. These include a discussion on the inaccuracies of the EKF, additional experiment results and listings of the algorithms presented and developped in this work.

Chapter 2

Bayesian Estimation

2.1 Introduction

Autonomous mobile robots depend on knowledge about the state of their environments to be able to make effective goal-directed decisions. Lack thereof would mean an inability to react to changes in their environment, rendering the robots practically useless. Direct access to the world's internal state is only possible in simulated environments, where a programming design choice could be to have the robot have access to the system's hidden state information, e.g. current absolute position of the robot, wind velocities throughout the simulated world, locations and other properties of arbitrary objects etc. Considered from the perspective of real world scenarios, this kind of access remains wishful thinking.

In the real world, the system's (internal) state cannot be accessed directly but needs to be observed via its indirect effects on the robot's sensors. By perceiving events indirectly connected to the system's internal state, the robot can draw conclusions based on its own observation model mapping the measurements to system variables. This indirect observation introduces errors to the calculation of the system state, degrading the robot's performance in making an optimal decision. Here is where estimation theory enters the stage. Its task is to take the noisy or incomplete measurements and, based on a mathematical process model of the system together with the above mentioned observation model defining a mapping between system variables and measurements, to remove the noise as much as possible and calculate an estimate of the system state.

The problem statement to create such an estimator can be formulated as the question: "How do we optimally estimate the hidden system variables in a recursive fashion as noisy or incomplete measurements come available online?"[12]. This chapter will present solutions to that question, focusing on variations of recursive Bayesian estimation.

In estimation theory, when working with assumptions about random measurement noise, imprecise measurement equipment or other unmodeled effects supporting random modeling, it is customary to regard the measurement $\mathbf{z}$ as a random variable or random vector when multi-dimensional. A random variable has a probability distribution. In the case of discrete systems, this is called the probability function, for continuous systems it is called the density function. The observation vector's probability density function (*PDF*) $p(\mathbf{z}|\mathbf{x})$, also called the likelihood function, is assumed to be dependent on the inner state's configuration. It represents the probability of observation $\mathbf{z}$ being encountered if the system state is $\mathbf{x}$.

In Bayesian estimation, the state $\mathbf{x}$ is also regarded as a random variable having a known prior probability $p(\mathbf{x})$. It encompasses all that is known of the system state before the experimental outcome is observed. The corrected probability of the system state, $\mathbf{x}$, given an observed measurement, $\mathbf{z}$, is $p(\mathbf{x}|\mathbf{z})$. Bayes' rule in Equation (2.1) details how the posterior should be calculated to incorporate the new information from the received measurement. How to turn this rule into a recursively applicable method will be discussed in the next section.

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})} \tag{2.1}$$

2.1.1 Chapter Outline

After a short introduction to recursive Bayesian estimation in the next section, the major remaining part of the chapter covers different versions of Kalman Filters. The part on Kalman filters starts with a detailed look at the discrete and linear Kalman Filter, orienting itself at the description given by Welch and Bishop in [13]. Subsequently, the nonlinear alternative, the Extended Kalman Filter (*EKF*), is presented, commenting on its disadvantages. Having seen two discrete time versions, the continuous time Kalman-Bucy-Filter algorithm is discussed before moving on to the Unscented Kalman Filter (*UKF*), explaining the original in detail as well as presenting two of its extensions or modifications.

2.1.2 Recursive Bayesian Estimation

As discussed above, Equation (2.1), Bayes' Theorem, is the basis for this estimation method. A belief about a hypothesis, $p(\mathbf{x})$, is updated as measurements arrive. The system state, $\mathbf{x}$, can also be described as a Markov Process and the measurements, $\mathbf{z}$, as the observed outputs of a hidden Markov Model. A Markov Process of first order is a mathematical model for describing the evolution of future states depending only on the current state of the system. Systems conferred with this Markov property are also called memory-less systems. If the calculation of future states of a Markov process includes and depends on more past system states than only the current state, then such a process is called a higher order Markov process. A Hidden Markov Model (*HMM*) is a model of such a Markov process whose state is only indirectly (not directly) observable.

Figure 2.1 shows the stochastic relations between system states and measurements of a first order Hidden Markov Model at successive time instants. It is clear to see that at each time step k, the next future state, $\mathbf{x}_{k+1}$, only depends on the current system state $\mathbf{x}_k$. This stochastic relationship, formulated as a conditional probability distribution equation,

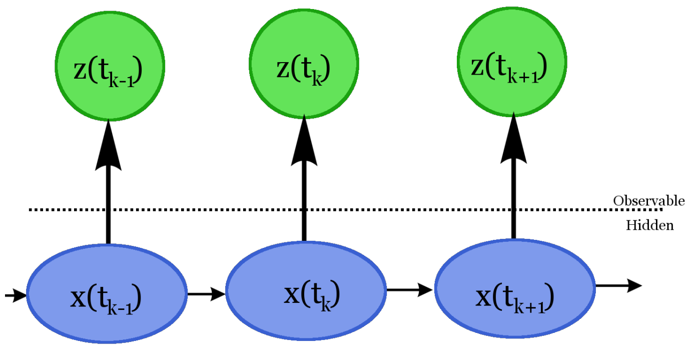


Figure 2.1: First Order Hidden Markov Model.
A Hidden Markov Model of a first order Markov process.

is given as

$$p\left(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{x}_{k-2},...,\mathbf{x}_0\right)=p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right). \tag{2.2}$$

Also illustrated in Figure 2.1 above is the conditional independence between a measurement $\mathbf{z}_k$ and system states at time instants different from k. A measurement $\mathbf{z}_k$ only depends on the system state at its time instant k, $\mathbf{x}_k$, i.e.

$$p\left(\mathbf{z}_k|\mathbf{x}_k,\mathbf{x}_{k-1},...,\mathbf{x}_0\right)=p\left(\mathbf{z}_k|\mathbf{x}_k\right). \tag{2.3}$$

The denominator of Bayes' rule is the probability distribution of the measurement, $p\left(\mathbf{z}_k\right)$. It is a scalar constant relative to the system state and can be computed over the set of all possible system states as

$$\begin{aligned} p\left(\mathbf{z}_k|\mathbf{z}_{k-1}\right) &= \int p\left(\mathbf{z}_k|\mathbf{x}_k\right)p\left(\mathbf{x}_k|\mathbf{z}_{k-1}\right)d\mathbf{x}_k \\ &= p\left(\mathbf{z}_k\right) \\ &= \int p\left(\mathbf{z}_k|\mathbf{x}_k\right)p\left(\mathbf{x}_k\right)d\mathbf{x}_k \\ &= \alpha^{-1}. \end{aligned} \tag{2.4}$$

Therefore, for the posterior probability distribution, the calculation of $p\left(\mathbf{z}_k\right)$ is usually neglected and only the nominator is computed and normalized, so that its integral is unitary.

Because of this, only $p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k)$ must be specified to calculate the posterior with Bayes' theorem. Applying this information and the above equations, the Bayes' rule of Equation (2.1) can be calculated:

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{z}_k) &= \frac{p\left(\mathbf{z}_k|\mathbf{x}_k\right)p\left(\mathbf{x}_k\right)}{p\left(\mathbf{z}_k\right)} \\ &= \alpha p\left(\mathbf{z}_k|\mathbf{x}_k\right)p\left(\mathbf{x}_k\right) \end{aligned}$$

To reiterate, $p(\mathbf{x}_k)$ is the a-priori defined prior probability distribution of the possible configurations of the system state. The term $p(\mathbf{z}_k|\mathbf{x}_k)$ is the conditional probability of encountering observation $\mathbf{z}_k$ if the system state happens to be $\mathbf{x}_k$. The posterior, $p(\mathbf{x}_k|\mathbf{z}_k)$, taking into account all measurements $\mathbf{z}_{1:k}$,

$$\mathbf{z}_{1:k} = \{\mathbf{z}_k, \mathbf{z}_{k-1}, ..., \mathbf{z}_1\},$$

comprises the complete solution to the recursive estimation problem, allowing us to calculate any optimal approximate of the state, such as the conditional mean:

$$\hat{\mathbf{x}}_k = E[\mathbf{x}_k|\mathbf{z}_{1:k}] = \int \mathbf{x}_k p(\mathbf{x}_k|\mathbf{z}_k)\, d\mathbf{x}_k.$$

This enables the reformulation of the problem statement:
"How do we recursively compute the posterior density as new measurements arrive online?"

Recursive Bayesian estimation is the optimal solution for this problem. By applying Bayes' rule and conditional independence between measurements as well as using the mathematical process and observation models, we can deduce a formula for the solution:

$$\begin{aligned}
p(\mathbf{x}_k|\mathbf{z}_{1:k}) &= \frac{p(\mathbf{z}_{1:k}|\mathbf{x}_k)\, p(\mathbf{x}_k)}{p(\mathbf{z}_{1:k})} \\
&= \frac{p(\mathbf{z}_k, \mathbf{z}_{1:k-1}|\mathbf{x}_k)\, p(\mathbf{x}_k)}{p(\mathbf{z}_k, \mathbf{z}_{1:k-1})} && (2.5) \\
&= \frac{p(\mathbf{z}_k|\mathbf{z}_{1:k-1}, \mathbf{x}_k)\, p(\mathbf{x}_k)\, p(\mathbf{z}_{1:k-1}|\mathbf{x}_k)}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})\, p(\mathbf{z}_{1:k-1})} \\
&= \frac{p(\mathbf{z}_k|\mathbf{z}_{1:k-1}, \mathbf{x}_k)\, p(\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{z}_{1:k-1})\, p(\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})\, p(\mathbf{z}_{1:k-1})\, p(\mathbf{x}_k)} && (2.6) \\
&= \frac{p(\mathbf{z}_k|\mathbf{z}_{1:k-1}, \mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} && (2.7) \\
&= \frac{p(\mathbf{z}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} && (2.8)
\end{aligned}$$

In (2.6) Bayes' Rule was applied and in (2.8) the conditional independence between measurements was used. To grasp the Bayesian recursive process better, we will divide (2.8) into two main steps, prediction and correction, and show the computations happening in them.

Time Update - "Prediction"

First in the prediction step, the posterior at time $k-1$, $p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1})$, is extrapolated forward in time to calculate the prior at time k:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})\, d\mathbf{x}_{k-1}. \qquad (2.9)$$

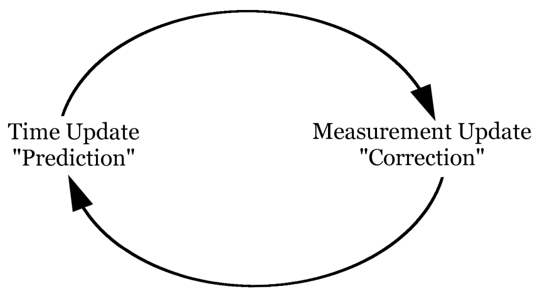


Figure 2.2: Predict & Correct Cycle.

This is done using the probabilistic process model, here function $\mathbf{f}$, given as

$$\mathbf{x}_k = \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k; \mathbf{w}\right). \tag{2.10}$$

Here, function $\mathbf{f}$'s arguments are the state to project forward in time, $\mathbf{x}_{k-1}$, potential known external influences acting on the system such as control input, $\mathbf{u}_k$, and the process model's inherent noise, $\mathbf{v}_k$. The process model $\mathbf{f}$ as well as the observation model $\mathbf{h}$ can be parametrized by a vector $\mathbf{w}$. The models are covered in detail in Chapter 3.

The probability distribution $p\left(x_k|x_{k-1}\right)$ is also called the state transition density, since it describes the probability distribution to go from one state x_{k-1} to the next in time:

$$p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) = \int \delta\left(\mathbf{x}_k - \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k; \mathbf{w}\right)\right) p\left(\mathbf{v}_k\right) d\mathbf{v}_k, \tag{2.11}$$

where $\delta\left(\cdot\right)$ denotes the Dirac-delta function. The Dirac-delta function is a generalized function that is equal to zero for all arguments but zero, where it has an infinitely large peak[14]. It is not strictly a function, because its integral is not zero, but it can be manipulated like a function in many cases.

Measurement Update - "Correction"

The next step is correcting the prediction by integrating the incoming noisy measurement into our posterior density. With the help of the observation likelihood, $p\left(\mathbf{z}_k|\mathbf{x}_k\right)$, the posterior is updated to reflect the current strength of belief into the system state after incorporating the measurement:

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right) = \frac{p\left(\mathbf{z}_k|\mathbf{x}_k\right) p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right)}{p\left(\mathbf{z}_k|\mathbf{z}_{1:k-1}\right)}. \tag{2.12}$$

From Equation (2.4) we know that $\alpha = \left(p\left(\mathbf{z}_k|\mathbf{z}_{1:k-1}\right)\right)^{-1}$ is a scalar constant. The scaling factor α can be calculated with

$$p\left(\mathbf{z}_k|\mathbf{z}_{1:k-1}\right) = \int p\left(\mathbf{z}_k|\mathbf{x}_k\right) p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right) d\mathbf{x}_k \tag{2.13}$$

and equation (2.12) can be written as

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right) = Cp\left(\mathbf{z}_k|\mathbf{x}_k\right)p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right).$$

The observation likelihood is computed similarly to the state transition density:

$$p\left(\mathbf{z}_k|\mathbf{x}_k\right) = \int \delta\left(\mathbf{z}_k - \mathbf{h}\left(\mathbf{x}_k, \mathbf{n}_k; \mathbf{w}\right)\right)p\left(\mathbf{n}_k\right)d\mathbf{n}_k, \tag{2.14}$$

where the observation model is implemented in function $\mathbf{h}$, given as

$$\mathbf{z}_k = \mathbf{h}\left(\mathbf{x}_k, \mathbf{n}_k; \mathbf{w}\right),$$

accepting as arguments the system state and the observation noise and an optional parameter vector, $\mathbf{w}$. $p\left(\mathbf{n}_k\right)$ is the observation noise distribution as introduced by the measuring instruments.

To summarize, figure 2.3 displays the important calculation steps for the recursive process.

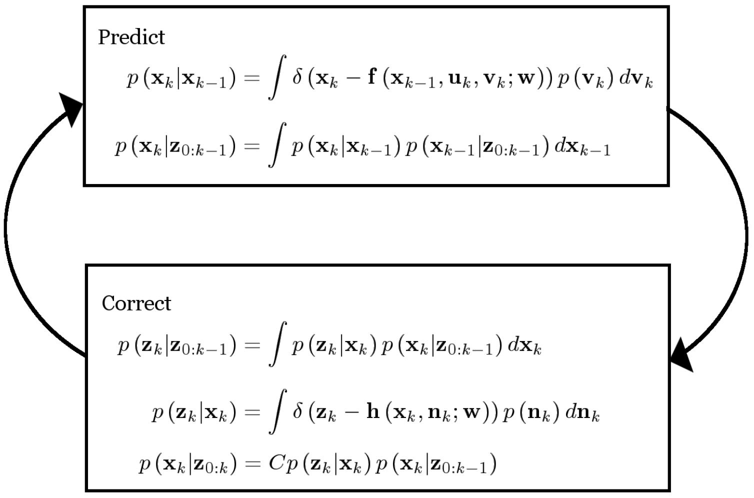

Predict
$p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) = \int \delta\left(\mathbf{x}_k - \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k; \mathbf{w}\right)\right)p\left(\mathbf{v}_k\right)d\mathbf{v}_k$
$p\left(\mathbf{x}_k|\mathbf{z}_{0:k-1}\right) = \int p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right)p\left(\mathbf{x}_{k-1}|\mathbf{z}_{0:k-1}\right)d\mathbf{x}_{k-1}$
Correct
$p\left(\mathbf{z}_k|\mathbf{z}_{0:k-1}\right) = \int p\left(\mathbf{z}_k|\mathbf{x}_k\right)p\left(\mathbf{x}_k|\mathbf{z}_{0:k-1}\right)d\mathbf{x}_k$
$p\left(\mathbf{z}_k|\mathbf{x}_k\right) = \int \delta\left(\mathbf{z}_k - \mathbf{h}\left(\mathbf{x}_k, \mathbf{n}_k; \mathbf{w}\right)\right)p\left(\mathbf{n}_k\right)d\mathbf{n}_k$
$p\left(\mathbf{x}_k|\mathbf{z}_{0:k}\right) = Cp\left(\mathbf{z}_k|\mathbf{x}_k\right)p\left(\mathbf{x}_k|\mathbf{z}_{0:k-1}\right)$

Figure 2.3: Recursive Bayesian Estimation Summary.

While this is the optimal solution to the recursive estimation problem, the computational cost to calculate the four multi-dimensional integrals is only manageable in linear Gaussian models of systems, for which the optimal solution is given by the Kalman Filter[15].

2.2 The Kalman Filter

The Kalman Filter, originally published by R.E. Kálmán in 1960[15], describes a recursive solution to the linear discrete-time estimation problem. In the close to 50 years afterward it has been applied countless times and modified into several versions to fit specific types of estimation problems, such as nonlinear data as well as continuous time driven systems. This does not mean that the Kalman Filter framework cannot be used on continuous problems, but only that the recursion is linear, its optimal terms given by:

$$\hat{\mathbf{x}}_k^- = E\left[\mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k\right)\right], \tag{2.15}$$

$$\hat{\mathbf{z}}_k^- = E\left[\mathbf{h}\left(\mathbf{x}_k^-, \mathbf{n}_k\right)\right], \tag{2.16}$$

$$\begin{aligned} \mathbf{K}_k &= E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)\left(\mathbf{z}_k - \hat{\mathbf{z}}_k^-\right)^T\right] E\left[\left(\mathbf{z}_k - \hat{\mathbf{z}}_k^-\right)\left(\mathbf{z}_k - \hat{\mathbf{z}}_k^-\right)^T\right]^{-1} \\ &= \mathbf{P}_{x_k \tilde{z}_k} \mathbf{P}_{\tilde{z}_k}^{-1}, \end{aligned} \tag{2.17}$$

as seen in [12]. Expectations of nonlinear functions of random variables need to be made here, which can only be calculated accurately for *linear* models with *Gaussian* random variables. This does not disallow its application to nonlinear systems, but requires further approximations to be made in order to be able to apply the framework to nonlinear systems. Some such approximations can be seen in the EKF (*Extended Kalman Filter*), as discussed in section 2.2.2.

Before we take a detailed look at the simple discrete-time Kalman Filter, further notational conventions need to be addressed. The most important are listed for convenient reference and quick checkups in Table 2.1.

In the linear case, the process model, $\mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k; \mathbf{w}\right)$, is a linear combination of a square state transition matrix, $\mathbf{F}$, multiplied with the prior state vector, $\mathbf{x}_{k-1}$, a square matrix $\mathbf{B}$ for the effect on the state transition of the optional control input, $\mathbf{u}_k$, and the process noise, $\mathbf{v}_k$. The observation model is similarly given by a square matrix $\mathbf{H}$ and the observation noise, $\mathbf{n}_k$. The process and measurement noises are assumed to be zero-mean, independent of each other with normal (Gaussian) probability distributions:[1]

$$\begin{aligned} p\left(\mathbf{v}_k\right) &= \mathcal{N}\left(0, \mathbf{Q}\right), \\ p\left(\mathbf{n}_k\right) &= \mathcal{N}\left(0, \mathbf{R}\right). \end{aligned}$$

The solution Kálmán derived calculates auxiliary factors such as the estimate error covariance, $\mathbf{P}$, and the *Kalman Gain*, $\mathbf{K}$, whose purposes are explained in the next section. Since this solution is recursive, we need to distinguish between our *a priori* and *a posteriori* variables, be they state estimates or covariances. To accomplish this, the superscript minus $^-$ is used to denote the *a priori* variable as in [13], e.g. the prior estimate, $\hat{\mathbf{x}}_k^-$, and the *a priori* estimate error covariance, $\mathbf{P}_k^-$. The same variable without superscript minus denotes its posterior version, such as the posterior state estmate, $\hat{\mathbf{x}}_k$, and the posterior estimate error covariance, $\mathbf{P}_k$. To be able to address the calculations in a more detailed manner, following assumptions are made without loss of generality: Let $\mathbf{x} \in \mathcal{R}^n$, $\mathbf{z} \in \mathcal{R}^m$ and $\mathbf{u} \in \mathcal{R}^r$. Likewise let the noises $\mathbf{v} \in \mathcal{R}^n$ and $\mathbf{n} \in \mathcal{R}^m$. This leads to dimensions of the matrices as given in Table 2.1.

[1]Although these matrices are presented as static, in practice they can change dynamically over time and be properly indexed, e.g. $\mathbf{F}_k$.

Symbol	Dimension	Description
$\mathbf{F}$	$n \times n$	state transition matrix
$\mathbf{B}$	$n \times r$	control input effect matrix
$\mathbf{H}$	$m \times n$	measurement mapping matrix
$\mathbf{Q}$	$n \times n$	symm. pos.-def. matrix of process noise covariance
$\mathbf{R}$	$m \times m$	symm. pos.-def. matrix of observation noise covariance
$\mathbf{P}$	$n \times n$	estimate error covariance matrix
$\mathbf{K}$	$n \times m$	Kalman Gain, blending factor matrix
$\mathbf{P}_k^-$	$n \times n$	prior estimate error covariance matrix

Table 2.1: Notational Conventions.

2.2.1 Linear Estimation

The original (Discrete) Kalman Filter estimates a system state directed by a linear discrete-time process model, $\mathbf{f}$, from Equation (2.10):

$$\begin{aligned}\mathbf{x}_k &= \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k\right) \\ &= \mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{v}_k,\end{aligned} \tag{2.18}$$

and relates system states to its noisy measurements, $\mathbf{z}_k$, via a linear observation model, $\mathbf{h}$:

$$\begin{aligned}\mathbf{z}_k &= \mathbf{h}\left(\mathbf{x}_k, \mathbf{n}_k\right) \\ &= \mathbf{H}\mathbf{x}_k + \mathbf{n}_k.\end{aligned} \tag{2.19}$$

The Kalman filter is an optimal solution to the recursive Bayesian estimation problem: how to calculate the posterior distribution accurately in a recursive way[15]. Therefore, the formulae derived by Kálmán follow the same structure of recursive Bayesian estimation. For this reason, Kálmán's solution is presented in the same order as before: First, the time update step is presented, followed by the measurement update step.

Time Update - "Prediction"

The equation to forward the system state estimate in time is similar to (2.18), given as

$$\hat{\mathbf{x}}_k^- = \mathbf{F}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k, \tag{2.20}$$

but without the effect of the process noise, equal to $\mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, 0\right)$. $\mathbf{F}$ is a square matrix containing the process model calculations for progressing a system state in time. It is called the *state transition* matrix or function. The result of the time progression is linearly combined with the effects of the optional input variable, $\mathbf{u}_k$, on the system state. For example, $\mathbf{u}_k$ can be the control input for a robot. Since the input is independent from the system state, it offers additional information that the process model cannot extract from the state estimate alone. How the control input affects the state is defined in the $n \times m$ matrix

B. The process noise cannot be ignored to properly estimate the state, so it is treated by calculating the *a priori* estimate error covariance $\mathbf{P}_k^-$:

$$\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}. \tag{2.21}$$

Equation (2.21) includes the process noise that overlays the system state by adding the noise covariance to the forwarded error covariance. The state estimate error covariance $\mathbf{P}_k$ represents how strong we believe the state estimate $\hat{\mathbf{x}}_k$ is polluted by noise.

Measurement Update - "Correction"

In the measurement update step an observation, $\mathbf{z}_k$, is received from a sensor that perceived indirect effects of the actual system state, $\mathbf{x}_k$, on the environment. This measurement is used to correct the current *a priori* estimate of the system state, $\hat{\mathbf{x}}_k^-$. Formulae behave similarly to the process noise equations by not using the measurement noise directly but by adding the measurement noise covariance to the error covariance, $\mathbf{P}$. Furthermore, they calculate another auxiliary factor, $\mathbf{K}$.

Let's take a look at all measurement update equations and examine them in order:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T\left(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}\right)^{-1}, \tag{2.22}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-\right), \tag{2.23}$$

$$\mathbf{P}_k = \left(\mathbf{1} - \mathbf{K}_k\mathbf{H}\right)\mathbf{P}_k^-. \tag{2.24}$$

In (2.22), the *a priori* estimate error covariance and observation model are used to calculate the *Kalman Gain* $\mathbf{K}_k$. Its purpose is to minimize the posterior estimate error covariance, $\mathbf{P}_k = E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T\right]$, so that the filter stays optimal and does not degrade. There are various mathematical formulations for calculation of the Kalman Gain, yet these forms are algebraically the same and differ only in their computational complexities. In other words, $\mathbf{K}$ is the *gain* or *blending factor* by which the *a priori* estimate $\hat{\mathbf{x}}_k^-$ is adjusted with the innovative difference $\tilde{\mathbf{z}}_k$,

$$\tilde{\mathbf{z}}_k = \left(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-\right),$$

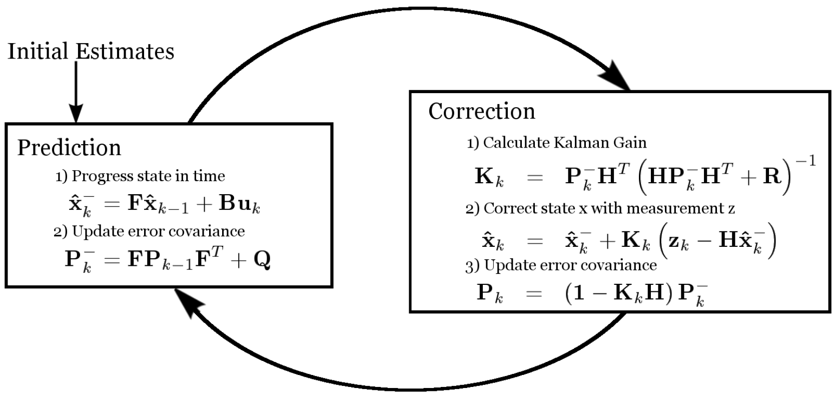
between the actual measurement $\mathbf{z}_k$ and the expected $\hat{\mathbf{z}}_k$,

$$\hat{\mathbf{z}}_k = \mathbf{H}\hat{\mathbf{x}}_k^-$$

in (2.23). The difference $\tilde{\mathbf{z}}_k$ is also called the *innovation* or *residual*, embodying the information gained by the observation. The Kalman Gain represents the strength of the influence of the innovation on the state estimate, i.e. if the trust is stronger in the prior estimate or in the measurement. A detailed look at the derivation of the Kalman gain is presented in [16]. After the correcting adjustment of the state estimate, the *a posteriori* estimate error covariance, $\mathbf{P}_k$, is updated in Equation (2.24) to portray the belief in the current estimate, where $\mathbf{1}$ is the identity matrix with appropriate dimension. Various forms for the computation of $\mathbf{P}_k$ exist as well.

Homologous to recursive Bayesian estimation, the Kalman Filter runs in a predict-correct cycle as portrayed in Figure 2.2. To start the loop, initial values are necessary for the variables in the mathematical formulae, i.e. here $\hat{\mathbf{x}}_0$ and $\mathbf{P}_0$. Figure 2.4 of the Kalman filter's predict-correct-cycle reflects this necessity.

Regarding the Kalman Filter framework as a whole, after each *predict* and *correct* pair the cycle loops the process and uses the previous *a posteriori* estimate, $\hat{\mathbf{x}}_{k-1}$, to predict the new *a priori* estimates, $\hat{\mathbf{x}}_k^-$ and $\mathbf{P}_k^-$. This recursive characteristic is the reason why the Kalman Filter can be applied to real world problems as the optimal solution for linear models in discrete time intervals. Instead of having to calculate offline on the *complete* sets of estimates and measurements at each step, as e.g. the Wiener Filter does, the Kalman Filter computes the estimates of the current step and reuses them for the next recursion step. To illustrate this fact, the figure showing an abstract sketch of the predict and correct cycle



Initial Estimates
Prediction
1) Progress state in time
$\hat{\mathbf{x}}_k^- = \mathbf{F}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k$
2) Update error covariance
$\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}$
Correction
1) Calculate Kalman Gain
$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T\left(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}\right)^{-1}$
2) Correct state x with measurement z
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-\right)$
3) Update error covariance
$\mathbf{P}_k = \left(\mathbf{1} - \mathbf{K}_k\mathbf{H}\right)\mathbf{P}_k^-$

Figure 2.4: The Detailed Kalman Filter Loop.

is extended to contain the equations for the update steps in Figure 2.4, and an algorithmic formulation of the Kalman filter is provided with Algorithm C.1 in Appendix C.

2.2.2 Nonlinear Estimation

The Kalman Filter framework can be applied to linear and nonlinear systems. In the nonlinear case certain approximations must be made to continue to adhere to Kálmán's assumptions. His assumptions are (from [12, 15]):

1. Consistent minimum variance estimates of system random variables (and consequently the posterior state distribution) can be calculated by maintaining only their *first* and *second order moments* (*means* and *covariances*, resp.)

2. The posterior update is a *linear* function of the prior knowledge of the system, $p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right)$, and the new observed information, $p\left(\mathbf{z}_k|\mathbf{x}_k\right)$. In other words, it is assumed that Equation (2.12) can be accurately approximated by a linear function.

3. Accurate predictions of the state and of the observation can be calculated (using process and observation model). These are needed to approximate first and second order moments of $p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right)$ and $p\left(\mathbf{z}_k|\mathbf{x}_k\right)$.

Written and implied, but not emphasized, is the fact that the densities do not have to be Gaussian, but the estimator only retains the Gaussian elements (mean and covariance) of these densities. This suffices only if the first assumption holds, that minimum variance estimates can be calculated (from means and covariances) in a consistent way. If this is true, “then the resulting Gaussian approximate posterior [distribution] will have significant support overlap with the true posterior”, signifying that the true distribution is fairly similar to a Gaussian distribution.

Assumption 2 is made to permit computational feasibility and practical realization of the estimator using efficient implementations of linear algebra methods. For linear models to which the discrete Kalman Filter can be applied, this requirement is already satisfied. Nonlinear model descriptions necessitate linearization for the update step in the Kalman Filter framework. Such a linearization of (2.12) is applied in the *Extended Kalman Filter (EKF)*. The EKF has become a well-established method in research and industry used to reduce noise in a wide variety of applications.

The equations of the EKF are similar to those of the linear Kalman Filter, in fact, Figure 2.4 can be transferred almost completely to the Extended Kalman Filter, barring some few adjustments. The main differences are the process and observation models. Here, the functions $\mathbf{f}$ and $\mathbf{h}$ describing these models are nonlinear:

$$\mathbf{x}_k = \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k\right), \tag{2.25}$$

$$\mathbf{z}_k = \mathbf{h}\left(\mathbf{x}_k, \mathbf{n}_k\right). \tag{2.26}$$

The optimal terms for the filter recursion are still (2.15)-(2.17):

$$\begin{aligned}
\hat{\mathbf{x}}_k^- &= E\left[\mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k\right)\right], \\
\hat{\mathbf{z}}_k^- &= E\left[\mathbf{h}\left(\mathbf{x}_k, \mathbf{n}_k\right)\right], \\
\mathbf{K}_k &= E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)\left(\mathbf{z}_k - \hat{\mathbf{z}}_k^-\right)^T\right] E\left[\left(\mathbf{z}_k - \hat{\mathbf{z}}_k^-\right)\left(\mathbf{z}_k - \hat{\mathbf{z}}_k^-\right)^T\right]^{-1}.
\end{aligned}$$

The non-linearity of the process and observation model functions is approximated by *Taylor Series Expansion*. Taylor Series Expansion is the method of approximating a *differentiable* function around a point by a *finite* number of terms of a Taylor Series. Furthermore, a Taylor Series is an infinite sum of Taylor polynomials of an arbitrary *differentiable* function f around a given point (2.28). These Taylor polynomials are terms whose coefficients are calculated from the derivatives at the given point $(a, f(a))$ of function f, as in (2.28). Taylor's Theorem states that a differentiable function can be approximated locally around a given point by orders of the specific Taylor Series. Especially, it states that an *infinite*

Taylor Series indeed represents the function f at that point:

$$\begin{aligned} f(x) &\approx f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots && (2.27) \\ &= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n . && (2.28) \end{aligned}$$

Calculating the Taylor Series to an infinite depth for mathematically accurate results is computationally not feasible, so a compromise has to be made to only calculate the terms up to an n-th order. The resulting error between the approximate and the actual value should be small for x near enough to a, in order to comply to Kálmán's assumptions to stay an optimal estimator. Taylor presents methods to estimate precisely how small the error is.

The Extended Kalman Filter applies the Taylor Expansion to multiple variables, the formulation using multi-index notation[2] given as

$$f(x) \approx \sum_{|\boldsymbol{\alpha}|=k}^{n} \frac{1}{\boldsymbol{\alpha}!} \frac{\partial^{\boldsymbol{\alpha}} f(a)}{\partial x^{\boldsymbol{\alpha}}} (x-a)^{\boldsymbol{\alpha}},$$

where

$$\begin{aligned} \boldsymbol{\alpha} &= (\alpha_1, \alpha_2, \ldots, \alpha_n), \\ |\boldsymbol{\alpha}| &= \alpha_1 + \alpha_2 + \cdots + \alpha_n, \\ \boldsymbol{\alpha}! &= \alpha_1! \cdot \alpha_2! \cdots \alpha_n!, \\ \mathbf{x}^{\boldsymbol{\alpha}} &= x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}, \\ \partial^{\boldsymbol{\alpha}} &= \partial_1^{\alpha_1} \partial_2^{\alpha_2} \cdots \partial_n^{\alpha_n}, \end{aligned}$$

and where

$$\partial_i^{\alpha_i} = \frac{\partial^{\alpha_i}}{\partial x_i^{\alpha_i}}.$$

The EKF approximates only to the first order, calculating the *Jacobian* matrices ($\mathbf{F}$ and $\mathbf{H}$) for the nonlinear (and differentiable) functions of the process model $\mathbf{f}$ and the observation model $\mathbf{h}$, as well as for the noise variables ($\mathbf{v}_k$ and $\mathbf{n}_k$) of the models, i.e. the matrices $\mathbf{V}$ and $\mathbf{L}$. This results in the matrices as defined by the following partial derivatives:

$$\begin{aligned} \mathbf{F}_k &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_k} && \Rightarrow \left[\mathbf{F}_k\right]_{[i,j]} = \left.\frac{\partial \mathbf{f}_{[i]}}{\partial \mathbf{x}_{[j]}}\right|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_k} && (2.29) \\ \mathbf{H}_k &= \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1}^-} && \Rightarrow \left[\mathbf{H}_k\right]_{[i,j]} = \left.\frac{\partial \mathbf{h}_{[i]}}{\partial \mathbf{x}_{[j]}}\right|_{\hat{\mathbf{x}}_{k-1}^-} && (2.30) \\ \mathbf{V}_k &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{v}}\right|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_k} && \Rightarrow \left[\mathbf{V}_k\right]_{[i,j]} = \left.\frac{\partial \mathbf{f}_{[i]}}{\partial \mathbf{v}_{[j]}}\right|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_k} && (2.31) \\ \mathbf{L}_k &= \left.\frac{\partial \mathbf{h}}{\partial \mathbf{n}}\right|_{\hat{\mathbf{x}}_{k-1}^-} && \Rightarrow \left[\mathbf{L}_k\right]_{[i,j]} = \left.\frac{\partial \mathbf{h}_{[i]}}{\partial \mathbf{n}_{[j]}}\right|_{\hat{\mathbf{x}}_{k-1}^-} . && (2.32) \end{aligned}$$

[2]More on multi-index notation can be found in [17].

The matrices from (2.32) are recalculated every time step, as indicated by the index k.

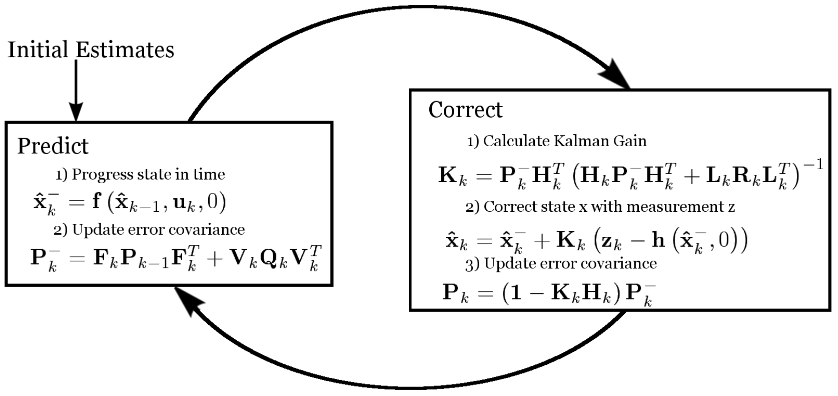

Initial Estimates
Predict
1) Progress state in time
$\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, 0)$
2) Update error covariance
$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{V}_k \mathbf{Q}_k \mathbf{V}_k^T$
Correct
1) Calculate Kalman Gain
$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{L}_k \mathbf{R}_k \mathbf{L}_k^T)^{-1}$
2) Correct state x with measurement z
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_k^-, 0))$
3) Update error covariance
$\mathbf{P}_k = (\mathbf{1} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-$

Figure 2.5: The Detailed Extended Kalman Filter Loop.

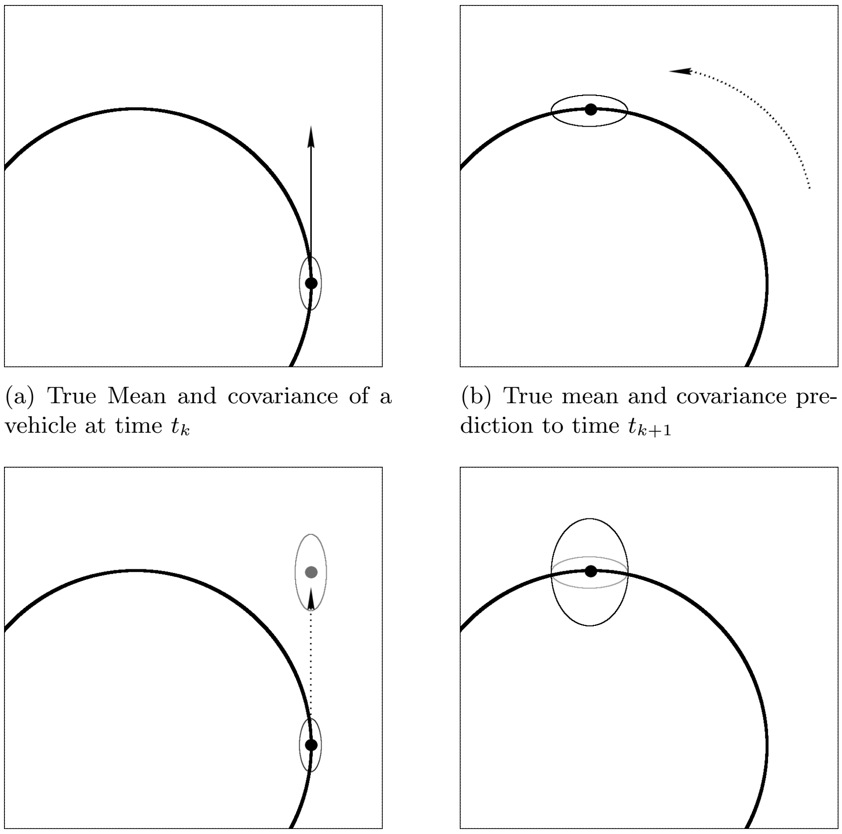
Figure 2.5 shows the familiar predict-correct cycle with the EKF equations. An algorithmic formulation of the extended Kalman filter is presented in Algorithm C.2.

Inaccuracies of the EKF

Unfortunately, the linearization by approximation using first-order truncated Taylor Expansion in general breaks the “optimal” property of the Kalman Filter framework. Since the EKF only considers the zeroth and first elements of the series in its approximation, all higher order derivatives need to be effectively zero in order for the approximations to be valid. Put differently, the zeroth and first order terms (mean and covariance) of Equation (2.27) must dominate over the higher order derivatives. Figure 2.6 gives an example for this weak point, showing a vehicle traveling on a circular path with uncertainty in speed.

Another disadvantage of the EKF is that its covariance usually underestimates the effect of noise on the state estimate, overrating the confidence or belief in the state estimate. Furthermore, although the probability distribution of the random variable x (as captured by its covariance $\mathbf{P}_x$) is important for the validity of the EKF’s linearization, the fact that x is a random variable with an intrinsic uncertainty is completely ignored during the linearization process. That is, by linearizing around a single point (the current state estimate) the crucial expectation operator E from Equations (2.15)-(2.17) is lost. The same applies to the noise variables v and n.

Should the estimate (state or covariance) stray too far from the correct state, then the EKF will diverge quickly.
Further detailed mathematical and experimental discussion of the flaws of the extended Kalman filter can be found in [12], with a short reproduction thereof presented in Appendix A.



(a) True Mean and covariance of a vehicle at time t_k

(b) True mean and covariance prediction to time t_{k+1}

(c) EKF prediction of mean with linear covariance propagation

(d) EKF prediction adjusted to compensate for linearization error

Figure 2.6: EKF Linearization Error Example (from [18]).
A) The true state and covariance terms at time instant t_k. **B)** The true state and covariance terms at next time step t_{k+1}. **C)** The EKF linearizes a nonlinear process function of the vehicle's motion at time instant t and predicts the position and error covariance based on the linearized model. **D)** The posterior state and covariance after correction with the measurement of the actual state was applied.

2.3 The Unscented Kalman Filter

The Unscented Kalman Filter is, as the name implies, a recursive estimator based on the optimal Gaussian approximate Kalman filter framework addressing some of the shortcomings of the EKF. To reiterate the previous section, the two discussed deficiencies of the EKF were:

1. Disregard of probabilistic uncertainty of underlying system state and noise random variables during linearization.

2. Limited accuracy of first order Taylor Expansion.

It is necessary to address these issues in order to improve the accuracy, consistency and efficiency of the EKF. The proposal by Julier et al.[19] considers a completely different strategy to approximate the optimal state estimate, observation estimate and optimal gain of the Kalman Filter from Equations (2.15) to (2.17). Julier's estimator, the *Unscented Kalman Filter* (UKF), is a deterministic sampling-based Kalman Filter without the need to calculate derivatives of the system equations. The EKF uses these derivatives to linearize the nonlinear functions by approximating them at a single point with a first order truncated Taylor Series expansion. The truncation at the first order often introduces large estimation errors in the estimated statistics of the posterior distributions of the states, especially if the models $\mathbf{f}$ and $\mathbf{h}$ are highly nonlinear and the local linearity assumption breaks down, meaning the effects of the higher order terms of the Taylor series become significant.

The UKF, on the other hand, does not try to linearize the nonlinear process and observation models but uses the *true* nonlinear models to propagate the random variables and consecutively approximates the posterior probability distribution of the state random variable. The UKF is nonetheless a Kalman filter and therefore treats the state random variable as a Gaussian random variable, but the distribution is defined by deterministically chosen sample points, capturing its true mean and covariance completely. These sample points are propagated through the nonlinear system (in addition to the state random variable) and afterward capture the posterior mean and covariance precisely to the 2^{nd} order for any non-linearity. The deterministic method to choose the sample points is called the *Unscented Transformation* and is elaborated in the next section. A generalizing extension of the Unscented Transformation, the *Scaled Unscented Transformation*, constitutes the major algorithmic improvement of the unscented Kalman filter.

2.3.1 Unscented Transformation

The Unscented Transformation (UT) [18] is based on the approach that estimating a probability distribution is easier than approximating an arbitrary nonlinear function. It is a procedure to calculate the statistics (not limited to mean and covariance) of a random variable subjected to a nonlinear transformation. The general problem is having an N-dimensional random vector $\mathbf{x}$, with mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_x$, propagated through an arbitrary nonlinear function $\mathbf{g}$, given as

$$\mathbf{y} = \mathbf{g}\left(\mathbf{x}\right),$$

and wanting to predict the posterior mean $\bar{\mathbf{y}}$ and covariance $\mathbf{P}_y$ of the M-dimensional random variable $\mathbf{y}$. In Kalman filtering specifically, there are two such nonlinear transformations, the process and observation models $\mathbf{f}$ and $\mathbf{h}$ as in Equations (2.25) and (2.26), used to predict the state at the next time step $\hat{\mathbf{x}}_k^-$ or the corresponding observation $\hat{\mathbf{z}}_k$.

The general method applied by the UT is to deterministically choose a number of sample points, e.g. $2N+1$, to generate a discrete distribution with the same first and second (and possibly higher) moments as the prior distribution, then directly transform these points through the nonlinear functions. The posterior mean and covariance of this transformed set can then be calculated, representing the estimate of the nonlinear transformation of the original distribution. Dividing this basic method into the three major steps results in:

1. Compute set of sigma points as matrix $\mathbf{X}$.
2. Transform each point in the set: $\mathbf{Y}_i = \mathbf{g}(\mathbf{X}_i)$, columns of matrix $\mathbf{Y}$.
3. Calculate $\bar{\mathbf{y}}$ and $\mathbf{P}_y$ by computing mean and covariance of points $\mathbf{Y}_i$.

Figure 2.7 summarizes this unscented transformation routine in a schematic diagram.

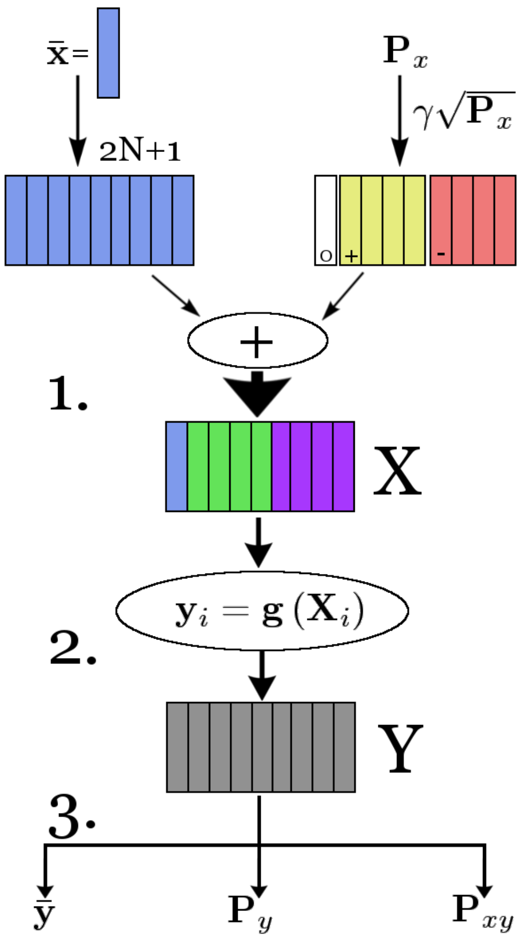


Figure 2.7: Schematic diagram of the unscented transformation.
The three major steps of the SUT are emphasized. **1.)** Calculate Sigma Points from state and covariance. **2.)** Propagate through nonlinear function. **3.)** Calculate posterior mean, covariance and cross-covariance.

The three major steps to calculate the posterior statistics, only mean and covariance for now, are explained in detail below:

1. Given the prior mean, $\bar{\mathbf{x}}$, and covariance, $\mathbf{P}_x$, of an N-dimensional Gaussian probability distribution of $\mathbf{x}$, the set $\mathcal{S}$ of $2N+1$ weighted samples, called *sigma points*, is calculated, capturing the true mean and covariance. One strategy to select appropriate sigma points is given as follows,

$$\begin{aligned}
&\mathbf{X}_0 = \bar{\mathbf{x}} && , w_0 = \frac{\kappa}{N+\kappa} && (i = 0)\,,\\
&\mathbf{X}_i = \bar{\mathbf{x}} + \left(\sqrt{(N+\kappa)\,\mathbf{P}_x}\right)_i && , w_i = \frac{1}{2\,(N+\kappa)} && (0 < i \leq N)\,,\\
&\mathbf{X}_i = \bar{\mathbf{x}} - \left(\sqrt{(N+\kappa)\,\mathbf{P}_x}\right)_{i-N} && , w_i = \frac{1}{2\,(N+\kappa)} && (N < i \leq 2N)\,.
\end{aligned}$$

Where $\mathbf{X}$ is a $N \times (2N+1)$ matrix of sigma points, with each column $\mathbf{X}_i$ being a sigma point in the state space. $\left(\sqrt{(N+\kappa)\,\mathbf{P}_x}\right)_i$ is the ith column of the square root of the weighted covariance matrix, w_i is the weight for the ith sigma point and κ is a scaling parameter controlling the effects of fourth and higher order moments. The matrix square-root of a positive-definite matrix is not deterministically unique, therefore any orthonormal transformation of the square root (and thereby also of the set of sigma points) can be chosen as the square root (and as the set of sigma points). This invalidates the restrictions of using orthogonal or symmetric matrix square roots that are numerically sensitive and computationally expensive to find, allowing the application of efficient and stable methods such as the Cholesky decomposition. In addition, alternative sigma point selection techniques can be employed to capture higher order moments, commonly requiring a larger amount of sigma points.

2. Subsequently to calculating the set of sigma points, each point $\mathbf{X}_i$ is propagated through the nonlinear function as follows,

$$\mathbf{Y}_i = \mathbf{g}\left(\mathbf{X}_i\right) \qquad \forall \quad i,$$

producing the set $\{\mathbf{Y}_i\} \subset \mathbb{R}^M$ so that $\mathbf{Y} \subset \mathbb{R}^{M \times 2N+1}$.

3. During the next step, the approximate mean, covariance and cross-covariance of $\mathbf{y}$ are computed as follows:

$$\begin{aligned}
\bar{\mathbf{y}} &\approx \sum_{i=0}^{2N} w_i \mathbf{Y}_i,\\
\mathbf{P}_y &\approx \sum_{i=0}^{2N} w_i \left(\mathbf{Y}_i - \bar{\mathbf{y}}\right)\left(\mathbf{Y}_i - \bar{\mathbf{y}}\right)^T,\\
\mathbf{P}_{xy} &\approx \sum_{i=0}^{2N} w_i \left(\mathbf{X}_i - \bar{\mathbf{x}}\right)\left(\mathbf{Y}_i - \bar{\mathbf{y}}\right)^T.
\end{aligned} \tag{2.33}$$

Figure 2.8 illustrates the advantage of this method using the example from Figure 2.6 of a vehicle traveling on a circular path with an uncertainty in speed.

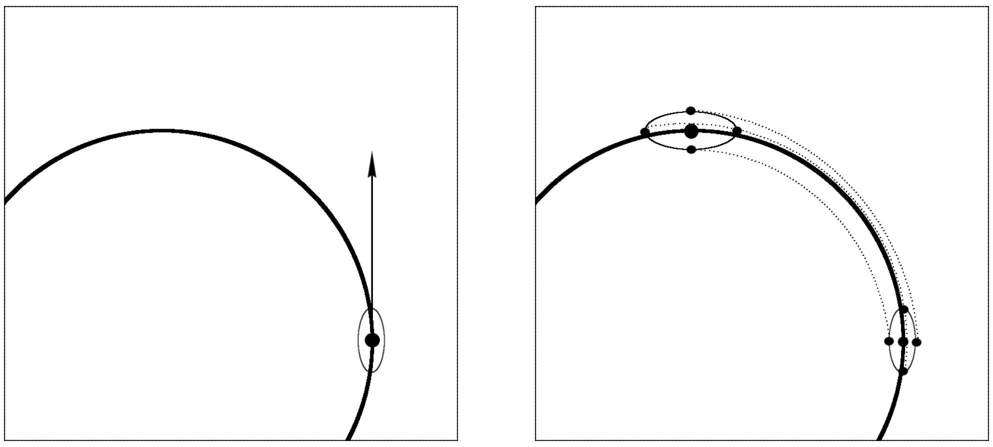

(a) Vehicle on circular path with uncertainty in speed (b) UKF predicts mean and covariance correctly using UT

Figure 2.8: UKF Example for Unscented Transform.

Analyzing the performance of this sigma point calculation scheme, the approximations are calculated accurately to the second order for *any* nonlinear function and up to the third order of the Taylor series expansion for true Gaussian priors. Errors are introduced above the respective second or third order moments, but are scaled by the parameter κ, as discussed later. This reflects a great improvement over the accuracy of the EKF's linearization technique.

Before turning to the effects of the parameter κ, the properties concerning the remaining variables of the unscented transformation must be discussed. As N (dimension of state space) increases, so does the radius of the sphere encompassing all sigma points, amplifying the risk of including non-local effects within the sampling, although the posterior mean and covariance are still captured correctly to the second order. Depending on the magnitude of the non-linearity however, this can lead to significant difficulties. The parameter κ was introduced to deal with that dilemma, enabling the scaling of the sigma points closer to and farther away from the prior mean $\bar{\mathbf{x}}$. This distance $|\mathbf{X}_i - \bar{\mathbf{x}}|$ is proportional to $\sqrt{N+\kappa}$. The parameter κ affects this distance in the following way: If $\kappa = 0$, then the distance is simply proportional to $\sqrt{N}$. For $\kappa > 0$ the sigma points are adjusted *away* from $\bar{\mathbf{x}}$ and when $\kappa < 0$ then the points are scaled *towards* $\bar{\mathbf{x}}$. The special case of $\kappa = 3 - N$ cancels the effect of N, but when $\kappa < 0$ then the weight $w_0 < 0$ and the event of the calculated covariance becoming non-positive semi-definite is suddenly possible.
In order to solve this problem, the *scaled* unscented transformation was developed[20].

2.3.2 Scaled Unscented Transformation

The scaled unscented transformation introduces another scaling parameter, α, and replaces the sigma points of the UT set $\mathcal{S} = \{w_i, \mathbf{X}_i | 0 \leq i \leq 2N\}$ with those of the set

$\mathcal{S}' = \{w'_i, \mathbf{X}'_i | 0 \leq i \leq 2N\}$, where

$$\mathbf{X}'_i = \mathbf{X}_0 + \alpha\left(\mathbf{X}_0 - \mathbf{X}_i\right) \qquad (0 \leq i \leq 2N)\,,$$
$$w'_i = \begin{cases} \frac{w_0}{\alpha^2} + \left(1 - \frac{1}{\alpha^2}\right) & (i = 0)\,, \\ \frac{w_i}{\alpha^2} & (0 < i \leq 2N)\,. \end{cases}$$

The parameter α with $0 \leq \alpha \leq 1$ scales the "radius" of the sigma point distribution to control (and even avoid) the impact of sampling non-local effects when the non-linearities are strong. Ideally, α should be a small number.

The two steps of calculating a sigma point set $\mathcal{S}$ and then scaling that set to receive $\mathcal{S}'$ can be merged into a single step. For that single step, the sigma point selection and weighting use a new variable, λ, as a function of α, κ and N, given as

$$\lambda = \alpha^2\left(N + \kappa\right) - N.$$

The formulae for sigma point selection and weights are consequently adjusted to:

$$\mathbf{X}_0 = \bar{\mathbf{x}} \qquad , w_0^m = \frac{\lambda}{N + \lambda} \qquad (i = 0), \tag{2.34}$$
$$\mathbf{X}_i = \bar{\mathbf{x}} + \left(\sqrt{\left(N + \lambda\right)\mathbf{P}_x}\right)_i \quad (0 < i \leq N), w_0^c = \frac{\lambda}{N + \lambda} + \left(1 - \alpha^2 + \beta\right) \qquad (i = 0), \tag{2.35}$$
$$\mathbf{X}_i = \bar{\mathbf{x}} - \left(\sqrt{\left(N + \lambda\right)\mathbf{P}_x}\right)_{i-N} (N < i \leq 2N), w_i^m = \quad w_i^c = \frac{1}{2\left(N + \lambda\right)} \qquad (0 < i \leq 2N). \tag{2.36}$$

Here are two things to note: Firstly, the zeroth terms of the mean and the covariance have differing weights: w_m is the weight vector for the mean and w_c for the covariance with elements w_i^m and w_i^c. Secondly, because the magnitude of the errors of the fourth and higher order moments is directly affected by the weighting of the zeroth sigma point[20], a third parameter β is introduced to the zeroth term of the covariance weight vector w_0^c. β is non-negative ($\beta \geq 0$) and can be used to incorporate prior knowledge of the fourth (kurtosis) or higher order Taylor series' terms. This knowledge is used to minimize the aforementioned errors at the higher orders. In the special case that $\mathbf{x}$ is Gaussian distributed, $\beta = 2$ is optimal[12]. The choice of parameters is problem specific. Research of optimization of the SUT's parameters is still ongoing. Some default values are given in the summary below.

To conclude the extension of the standard ("*simplex*") unscented transformation, an algorithmic notation is given in Algorithm C.3 and the Scaled Unscented Transformation is summarized as follows:

1. Choose *problem-specific* Parameters α, β and κ:

    - $0 \leq \alpha \leq 1$ controls the degree of spread of the sigma point distribution around $\bar{\mathbf{x}}$, default $\alpha = 0.001$.
    - $\beta \geq 0$ scales effect of zeroth sigma point, for minimization of 4^{th} or higher order moment errors, if prior knowledge is available. ($\beta = 2$ optimal for Gaussian).

- $\kappa \in \mathbb{R}$ scales sigma points towards or away from the mean, default $\kappa = 0$.

2. Calculate Sigma Points, using Equations (2.34)-(2.36).
3. Propagate each Sigma Point through given nonlinear function $\mathbf{g}(\cdot)$,

$$\mathbf{y}_i = \mathbf{g}(\mathbf{X}_i) \qquad \forall \quad i \in \{0, \ldots, 2n\}.$$

4. Calculate posterior mean, covariance and cross-covariance with Equations (2.33).

To depict the improvement of the Scaled Unscented Transformation in comparison to the EKF's linearization technique, Figure 2.9 is reproduced from [12]. In the Unscented Kalman

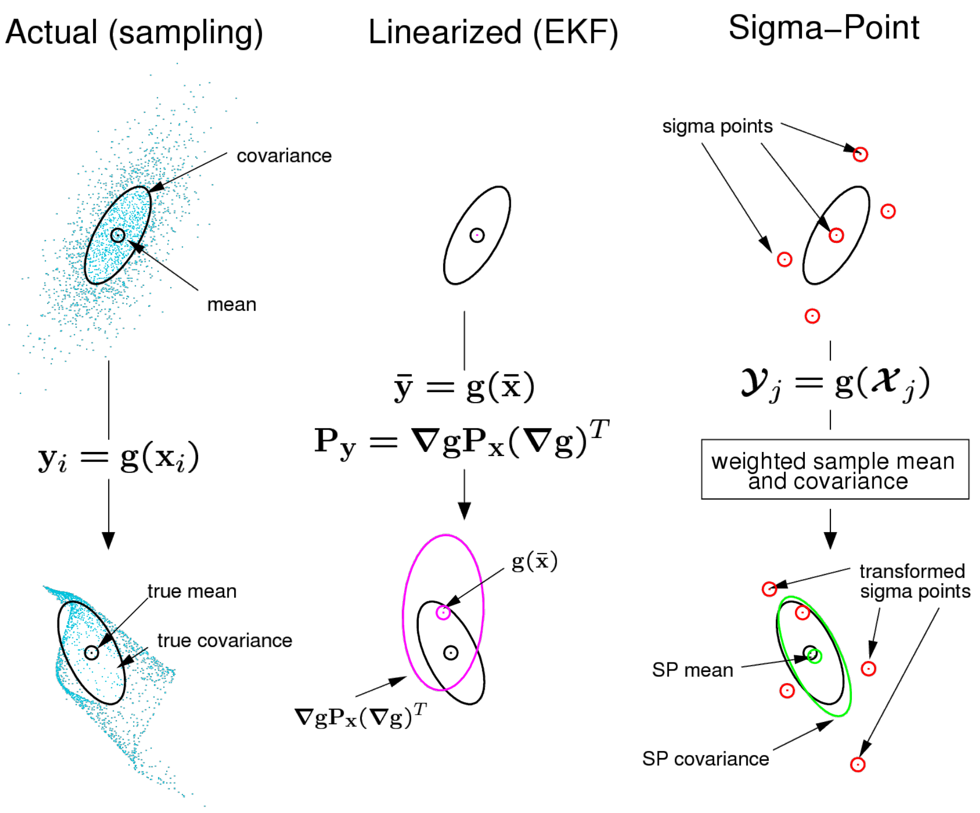

Actual (sampling)
Linearized (EKF)
Sigma–Point
covariance
mean
$\mathbf{y}_i = \mathbf{g}(\mathbf{x}_i)$
true mean
true covariance
$\bar{\mathbf{y}} = \mathbf{g}(\bar{\mathbf{x}})$
$\mathbf{P_y} = \nabla\mathbf{g}\mathbf{P_x}(\nabla\mathbf{g})^T$
$\mathbf{g}(\bar{\mathbf{x}})$
$\nabla\mathbf{g}\mathbf{P_x}(\nabla\mathbf{g})^T$
sigma points
$\mathcal{Y}_j = \mathbf{g}(\mathcal{X}_j)$
weighted sample mean and covariance
transformed sigma points
SP mean
SP covariance

Figure 2.9: Accuracy comparison between EKF and SUT from [12].
From left to right, the posterior statistics are calculated for a random variable that is propagated through an arbitrary nonlinear transformation: **A)** True statistics, **B)** 1^{st} order linearization (EKF), **C)** SUT. The superior performance of the SUT approach is evident.

Filter, the posterior statistics of the Scaled Unscented Transformation are recursively used as prior statistics of the next time step. The UKF method is formulated in Algorithm C.4.

2.3.3 Continuous Time Estimation

All estimators discussed above work in discrete time steps to propagate the random variables through linear or nonlinear models. The systems that these models represent, though, usually run in continuous time in the real world and are very common in engineering and physics application:

$$\begin{aligned}\frac{d}{dt}\mathbf{x}\left(t\right) &= \mathbf{f}\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right), t\right) + \mathbf{v}\left(t\right), \\ \mathbf{z}\left(t\right) = \frac{d}{dt}\mathbf{y}\left(t\right) &= \mathbf{h}\left(\mathbf{x}\left(t\right), t\right) + \mathbf{n}\left(t\right),\end{aligned}$$

where $\mathbf{z}\left(t\right)$ is the *formal* derivative of the measurement process. Therefore, a continuous time formulation of the models and consequently of the filters should intuitively yield better results, since the continuous time stochastic process formulation models the underlying system more accurately. Another advantage of a continuous time version is the numerical stability concerning the progression of the estimate error covariance. In contrast to a discrete time formulation of the noise process, where the covariance can become non-positive definite, a continuous time formulation of the noise processes cannot force the estimate error covariance to become non-positive definite. The continuous time version of the simple Kalman Filter is given as the *Kalman-Bucy-Filter.* A continuous time version of the EKF exists as the *extended* Kalman-Bucy filter[21, 22]. However, as demonstrated in the previous section, the EKF is far inferior to the UKF's performance, accuracy and consistency, therefore the focus of this section will be on the continuous time definition of the Unscented Kalman Filter. This section orientates itself around one such formulation of the UKF given by Särkkä[23].

Linear Estimation

For linear process and observation models, a continuous time formulation has been developed as the Kalman-Bucy filter[21, 22]. The process and observation models the KBF is based on are given as

$$\begin{aligned}\frac{d}{dt}\mathbf{x}\left(t\right) &= \mathbf{F}\left(t\right)\mathbf{x}\left(t\right) + \mathbf{v}\left(t\right), \\ \mathbf{z}\left(t\right) = \frac{d}{dt}\mathbf{y}\left(t\right) &= \mathbf{H}\left(t\right)\mathbf{x}\left(t\right) + \mathbf{n}\left(t\right),\end{aligned}$$

where $\mathbf{v}\left(t\right) \sim \mathcal{N}\left(0, \mathbf{Q}\left(t\right)\right)$ and $\mathbf{n}\left(t\right) \sim \mathcal{N}\left(0, \mathbf{R}\left(t\right)\right)$.

The differential equations for the filter update are as follows,

$$\begin{aligned}\frac{d}{dt}\hat{\mathbf{x}}\left(t\right) &= \mathbf{F}\left(t\right)\hat{\mathbf{x}}\left(t\right) + \mathbf{K}\left(t\right)\left(\mathbf{z}\left(t\right) - \mathbf{H}\left(t\right)\hat{\mathbf{x}}\left(t\right)\right), \\ \frac{d}{dt}\mathbf{P}\left(t\right) &= \mathbf{F}\left(t\right)\mathbf{P}\left(t\right) + \mathbf{P}\left(t\right)\left(\mathbf{F}\left(t\right)\right)^T + \mathbf{Q}\left(t\right) - \mathbf{K}\left(t\right)\mathbf{R}\left(t\right)\left(\mathbf{K}\left(t\right)\right)^T, \\ \mathbf{K}\left(t\right) &= \mathbf{P}\left(t\right)\left(\mathbf{H}\left(t\right)\right)^T\left(\mathbf{R}\left(t\right)\right)^{-1}.\end{aligned} \tag{2.37}$$

This simplified version considers noise properties in continuous time, under which condiction the observation noise covariance $\mathbf{R}(t)$ also represents the covariance of the innovation. Moreover, the distinction between the prediction and correction steps does not exist in continuous time[24]. These equations must be solved for a point in time to be able to draw a conclusion concerning the estimates at that time point. However, this is difficult because of the nonlinear terms in the covariance differential equation, achieving optimal estimates in few cases.

Nonlinear Estimation

For nonlinear estimation, the linearization concept of the EKF can be applied to the Kalman-Bucy filter, but this option has the same shortcomings of the EKF attached. A continuous time version of the UKF would be more desirable. Särkkä presents a formulation of the continuous UKF, also called the *unscented Kalman-Bucy filter*, which we will describe here[23]. Before that however, additional notational conventions are necessary to tidy up the presentation for easier reading.

Notational Conventions To clean up the notation, we write the expressions in matrix form:

$$\mathbf{Y} = g\left(\mathbf{X}\right),$$

where $\mathbf{X} \in \mathbb{R}^{N\times(2N+1)}$ is the matrix of (scaled) sigma points (with columns $\mathbf{X}_i$), g an arbitrary nonlinear function and $\mathbf{Y}$ the matrix of transformed sigma points (in our case $\mathbf{Y}$ will be the propagated state estimates matrix $\hat{\mathbf{X}}^{-}$ as well as the expected observations matrix $\hat{\mathbf{Z}}$).

The matrix of the scaled unscented transformation, $\mathbf{X}$, is calculated in matrix form with

$$\mathbf{X}\left(t\right) = \left[m\left(t\right) \quad \cdots \quad m\left(t\right)\right] + \gamma\left[0 \quad \sqrt{\mathbf{P}_x\left(t\right)} \quad -\sqrt{\mathbf{P}_x\left(t\right)}\right], \tag{2.38}$$

where $\gamma = \sqrt{N+\lambda}$ and $m\left(t\right) = \bar{\mathbf{x}}\left(t\right)$ is the prior mean at time t. The posterior mean, covariance and cross-covariance are then calculated after the nonlinear transformation was applied using the following equations,

$$\begin{aligned}
\bar{\mathbf{y}}\left(t\right) &= \mathbf{Y}\left(t\right) w_m, \\
\mathbf{P}_y\left(t\right) &= \mathbf{Y}\left(t\right) W \left(\mathbf{Y}\left(t\right)\right)^T, \\
\mathbf{P}_{xy}\left(t\right) &= \mathbf{X}\left(t\right) W \left(\mathbf{Y}\left(t\right)\right)^T.
\end{aligned}$$

Here, W is the weight matrix with dimension $(2N+1)\times(2N+1)$, calculated from the two weight vectors w_m and w_c from Equations (2.34) to (2.36) as follows:

$$W = \left(\mathbf{1} - \left[w_m \quad w_m \quad \cdots \quad w_m\right]\right) diag\left(w_c\right) \left(\mathbf{1} - \left[w_m \quad w_m \quad \cdots \quad w_m\right]\right)^T. \tag{2.39}$$

$\mathbf{1}$ is the identity matrix with appropriate dimension, and $diag\left(w_c\right)$ is a diagonal matrix with the elements of the weight vector w_c as its diagonal band and the rest equal to zero. Table 2.2 shows another summary of symbols used.

Symbol	Dimension	Description
$\mathbf{X}(t_k)$	$N \times (2N+1)$	Sigma point matrix at point in time t_k
$\mathbf{A}(t_k)$	$N \times N$	Cholesky decomposition of $\mathbf{P}_x(t_k)$
$\mathbf{\Phi}(\cdot)$	$N \times N$	Auxiliary function to get lower triangular part of matrix
$\mathbf{Z}(t)$	$M \times (2N+1)$	Matrix with measurement $\mathbf{z}(t)$ in each column
$\hat{\mathbf{Z}}(t)$	$M \times (2N+1)$	Matrix of expected measurements
$\bar{\mathbf{Z}}(t)$	$M \times (2N+1)$	Matrix with mean of expected measurements in each column

Table 2.2: UKF Notational Conventions.

Unscented Kalman-Bucy filter: By taking the equations of the discrete time UKF (Algorithm C.4) to the formal limit, the stochastic differential equations corresponding to the UKF time and measurement update processes can be derived [23] as follows,

$$\begin{aligned}\mathbf{K}(t) &= \mathbf{X}(t)\,W\,(\mathbf{h}(\mathbf{X}(t),t))^T\,[\mathbf{R}(t)]^{-1}\,,\\ \frac{d}{dt}m(t) &= \mathbf{f}(\mathbf{X}(t),t)\,w_m + \mathbf{K}(t)\,[\mathbf{z}(t) - \mathbf{h}(\mathbf{X}(t),t)\,w_m]\,,\\ \frac{d}{dt}\mathbf{P}(t) &= \mathbf{X}(t)\,W\,(\mathbf{f}(\mathbf{X}(t),t))^T + \mathbf{f}(\mathbf{X}(t),t)\,W\,(\mathbf{X}(t))^T\\ &\quad + \mathbf{Q}(t) - \mathbf{K}(t)\,\mathbf{R}(t)\,(\mathbf{K}(t))^T\,,\end{aligned} \tag{2.40}$$

where $\mathbf{z}(t)$ is defined as the formal derivative of the nonlinear observation model,

$$\mathbf{z}(t) = \frac{d}{dt}\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t),t) + \mathbf{n}(t)\,,$$

also called the differential measurement.

Although the mathematical equations are correct, often errors arise due to computational limitations such as finite numerical precision of computer arithmetic. For this reason Kalman filter equations are often implemented in such a way that the matrix square roots of covariance matrices are used instead of the matrices' actual values [25]. Since the UKF already uses the matrix square roots in its sigma points selection, the numerically stable square root version of continuous time UKF is attained by formulating the filter equations (2.40) as a differential equation for sigma points [23].

Square Root Unscented Kalman-Bucy filter: Here, the sigma point matrix is defined similar to Equation (2.38):

$$\mathbf{X}(t) = [m(t) \quad m(t) \quad \cdots \quad m(t)] + \gamma\,[0 \quad \mathbf{A}(t) \quad -\mathbf{A}(t)]\,, \tag{2.41}$$

where $\mathbf{A}(t)$ is the lower triangular matrix of the Cholesky factorization (the square root) of the covariance $\mathbf{P}(t)$, given as

$$\mathbf{A}(t) = \mathit{chol}(\mathbf{P}(t))\,.$$

Therefore the initial $\mathbf{A}(t_0)$ can be calculated from the initial covariance $\mathbf{P}(t_0)$ by Cholesky decomposition. The initial matrix square root, $\mathbf{A}(t_0)$, is the only one that needs to be calculated by actual Cholesky decomposition. The square root $\mathbf{A}(t)$ for all $t > 0$ can be extracted from $\mathbf{X}(t)$ at any step by very simple methods (see M, Φ and their applications in (2.42)). As a consequence, the covariance $\mathbf{P}(t)$ of the state never needs to be evaluated in the implementation of the algorithm.

The filter equations are given as

$$\begin{aligned}
\mathbf{K}(t) &= \mathbf{X}(t)\,W\left(\mathbf{h}(\mathbf{X}(t),t)\right)^T\left[\mathbf{V}(t)\,\mathbf{R}(t)\left(\mathbf{V}(t)\right)^T\right]^{-1}, \\
M(t) &= \left(\mathbf{A}(t)\right)^{-1}\Big[\mathbf{X}(t)\,W\left(\mathbf{f}(\mathbf{X}(t),t)\right)^T + \mathbf{f}(\mathbf{X}(t),t)\,W\left(\mathbf{X}(t)\right)^T \\
&\quad + \mathbf{L}(t)\,\mathbf{Q}(t)\left(\mathbf{L}(t)\right)^T \\
&\quad - \mathbf{K}(t)\,\mathbf{V}(t)\,\mathbf{R}(t)\left(\mathbf{V}(t)\right)^T\left(\mathbf{K}(t)\right)^T\Big]\left(\left(\mathbf{A}(t)\right)^{-1}\right)^T, \\
\frac{d}{dt}\left[\mathbf{X}(t)\right]_i &= \mathbf{f}(\mathbf{X}(t),t)\,w_m + \mathbf{K}(t)\left[\mathbf{z}(t) - \mathbf{h}(\mathbf{X}(t),t)\,w_m\right] \\
&\quad + \gamma\left[0 \quad \mathbf{A}(t)\,\Phi(M(t)) \quad -\mathbf{A}(t)\,\Phi(M(t))\right]_i,
\end{aligned} \tag{2.42}$$

where $\Phi(\cdot)$ is a simple function returning the lower triangular part of its argument matrix, defined as

$$\Phi_{ij}(M) = \begin{cases} M_{ij} & (i > j) \\ \frac{1}{2}M_{ij} & (i = j) \\ 0 & (i < j), \end{cases} \tag{2.43}$$

where i is the row number and j the column number of the element of matrix M. The process and observation noise covariance matrices have been divided into two matrices each. The previous symbol of the process noise covariance, $\mathbf{Q}(t)$, here denotes the diagonal dispersion matrix for the process noise and matrix $\mathbf{L}$ represents an arbitrary time-varying matrix independent of the state and measurement, so that

$$\mathbf{v}(t) \sim \mathcal{N}\left(0, \mathbf{L}(t)\,\mathbf{Q}(t)\left(\mathbf{L}(t)\right)^T\right).$$

Analogously, the previous symbol representing the observation noise covariance, $\mathbf{R}$, now represents the diagonal dispersion matrix of the measurement noise and $\mathbf{V}$ an arbitrary time-varying matrix independent of the state and measurement, so that

$$\mathbf{n}(t) \sim \mathcal{N}\left(0, \mathbf{V}(t)\,\mathbf{R}(t)\left(\mathbf{V}(t)\right)^T\right).$$

Continuous-Discrete UKF: With this numerically stable, continuous time formulation of the UKF, continuous time prediction equations can be derived as special cases of Equations (2.42)[26]. These prediction equations can be used in a continuous-discrete version of the UKF, which is an Unscented Kalman filter with a continuous time process model and a discrete time observation model (continuous time prediction and discrete correction step). The differential square root equations for the continuous time prediction step are given as

$$\begin{aligned} M(t) &= (\mathbf{A}(t))^{-1} \Big[\mathbf{X}(t) W (\mathbf{f}(\mathbf{X}(t),t))^T + \mathbf{f}(\mathbf{X}(t),t) W (\mathbf{X}(t))^T \\ &\quad + \mathbf{L}(t) \mathbf{Q}(t) (\mathbf{L}(t))^T \Big] \left((\mathbf{A}(t))^{-1} \right)^T, \\ \frac{d}{dt} [\mathbf{X}(t)]_i &= \mathbf{f}(\mathbf{X}(t),t) w_m + \gamma \left[0 \quad \mathbf{A}(t) \Phi(M(t)) \quad -\mathbf{A}(t) \Phi(M(t)) \right]_i . \end{aligned}$$

To obtain the mean estimate, integration is started from t_0 and continues from step to step over the time interval $[t_{k-1}, t_k]$. The correction step equations are given by the correction equations of the discrete time UKF from Algorithm C.4. Square root versions thereof have also been developed[27], presented in Algorithm C.6. In order to calculate square roots efficiently, e.g. of the measurement covariance, $\mathbf{P}_{y,k} = \mathbf{Y}_k W \mathbf{Y}_k^T$, additional operators are introduced:

qr $\{\cdot\}$ QR-Decomposition, modified:

The standard QR-Decomposition of a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, not necessarily of square dimension but with $n \geq m$, returns two matrices, an orthogonal matrix,$\mathbf{Q} \in \mathbb{R}^{n \times n}$, with $\mathbf{Q}\mathbf{Q}^T = \mathbf{1}$ and an upper (right) triangular matrix, $\mathbf{R} \in \mathbb{R}^{n \times m}$, with the bottom $(n-m)$ rows of $\mathbf{R}$ consisting entirely of zeroes, given as

$$\mathbf{A}^T = \mathbf{Q}\mathbf{R}.$$

Here, a modified version of the QR-decomposition returns the transpose of the $n \times n$ upper right triangular matrix of $\mathbf{R}$. The computational complexity of a QR-Decomposition is $\mathcal{O}\left(nm^2\right)$, but note that performing a Cholesky factorization directly on $\mathbf{P} = \mathbf{A}\mathbf{A}^T$ is $\mathcal{O}\left(m^3/6\right)$ plus $\mathcal{O}\left(nm^2\right)$ for forming $\mathbf{A}\mathbf{A}^T$.

The argument supplied to calculate the observation error covariance square root matrix is a matrix consisting of the columns $(\hat{\mathbf{Z}}_{i,k} - \bar{\mathbf{z}}_k)$ for $0 < i \leq 2N$. The zeroth column needs to be treated separately (see **cholupdate**), because its weight, w_0^c, can be negative.

cholupdate $\{\cdot,\cdot,\cdot\}$ Cholesky Factor Update:

If $\mathbf{A}$ is the original lower triangular Cholesky factor of $\mathbf{P} = \mathbf{A}\mathbf{A}^T$, then the updated Cholesky factor, $\check{\mathbf{A}}$, of the rank-1 updated (downdate for negative coefficients) matrix, $\check{\mathbf{P}}$, given as

$$\check{\mathbf{P}} = \mathbf{P} \pm \sqrt{c}\mathbf{u}\mathbf{u}^T,$$

is denoted as

$$\check{\mathbf{A}} = \mathbf{cholupdate}\{\mathbf{A}, \mathbf{u}, \pm c\}.$$

If $\mathbf{u}$ is a matrix instead of a vector, then the result is m consecutive updates of $\mathbf{A}$ using the m columns of $\mathbf{u}$. The computational complexity of this algorithm is only $\mathcal{O}\left(m^2\right)$.

Adjusted to the formulation and notation of this paper, the discrete time square root measurement update equations are given as follows:

$$\begin{aligned}
\bar{\mathbf{z}}_k &= \hat{\mathbf{Z}}_k w_m = \mathbf{h}\left(\hat{\mathbf{X}}_k^-, t_k\right) w_m, \\
\mathbf{A}'_{y,k} &= \mathbf{qr}\left\{\sqrt{w_1^c}\left(\hat{\mathbf{Z}}_k - \bar{\mathbf{z}}_k\right)_{1:2N,k} \quad \sqrt{\mathbf{R}}\right\}, \\
\mathbf{A}_{y,k} &= \mathbf{cholupdate}\left\{\mathbf{A}'_{y,k}, \left(\hat{\mathbf{Z}}_k - \bar{\mathbf{z}}_k\right)_{0,k}, w_0^c\right\}, \\
\mathbf{P}_{xy,k} &= \mathbf{X}_k^- W \left(\hat{\mathbf{Z}}_k\right)^T
\end{aligned} \tag{2.44}$$

with which the Kalman Gain is calculated by

$$\mathbf{K}_k = \mathbf{P}_{xy,k}\left((\mathbf{A}_{y,k})^{-1}\right)^T (\mathbf{A}_{y,k})^{-1}, \tag{2.45}$$

and the state estimate corrected as follows,

$$\begin{aligned}
m_k &= m_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \bar{\mathbf{z}}_k\right), \\
\mathbf{U} &= \mathbf{K}_k \mathbf{A}_{y,k}, \\
\mathbf{A}_k &= \mathbf{cholupdate}\left\{\mathbf{A}_k^-, U, -1\right\}.
\end{aligned} \tag{2.46}$$

Equations (2.44) propagate the predicted state estimate, $\hat{\mathbf{X}}^-(t_k)$, through the nonlinear observation model function, then calculate the observation mean, covariance square root and cross-covariance. Given $\mathbf{P}_{xy,k}$, the Kalman Gain can be calculated in Equation (2.45) followed by the correction of the estimated mean, m_k^-, with the current measurement, $\mathbf{z}_k$, and the update of the *a posteriori* estimate error covariance square root matrix, $\mathbf{A}_k$, in Equations (2.46).

The calculation of the matrix square roots of the measurement error covariance, $\sqrt{\mathbf{Y}_k W \mathbf{Y}_k^T}$, and the posterior state estimate covariance, $\mathbf{A}_k$, has been divided into two separate steps: a QR-Decomposition and a Cholesky update. This has been done mainly for two reasons: Firstly, it is more efficient than a direct Cholesky factorization of $\mathbf{Y}_k W \mathbf{Y}_k^T$, and secondly, the zeroth column needs to be treated separately for numerical stability, since its weight, w_0^c, can be negative. This algorithm can be further extended for the implementation with the application of sequential measurement updates, as presented in the next section.

2.3.4 Sequential Updates

A modification to the normal Kalman Filter was developed to address difficulties of the observation model encountered through the decades, namely the adjustment to *Sequential Updates*. This modification only affects the filter equations of the measurement update step, therefore this section will not list time update equations. The original approach of sequential updates was to decompose the measurement vector into scalar quantities and then to process these scalar measurements sequentially. The main purpose of this scalar conversion was to avoid the matrix inversions necessary in the computation of the Kalman Gain matrix, and thereby gain a reduction in computational time and hence an increase in real-time capability[28].

In the case of several sensors as measurement sources for the filter, these sensors usually supply data of different dimensions and at individual rates from each other. Thus such a complex sensor fusion and synchronization problem would be "solved" by forcing the sensors to a crude virtual common clock frequency equal to the lowest frequency of the sensors. Such an approach opens up new questions and challenges, such as what to do with observation data of sensors that have reported either multiple times or not yet during the periodic time interval? Since all measurements are being combined in a single measurement update function, simply ignoring the missing measurement data would not work. To solve this problem, sequential updates introduce a formulation where each sensor has its own observation model function.

The observation model equations for an Unscented Kalman filter with sequential updates are then, given the state estimate $\hat{\mathbf{X}}_k^- = \mathbf{f}\left(\hat{\mathbf{X}}(t_{k-1}), t_k\right)$ and the error covariance matrix $\mathbf{P}_k^- = \mathbf{P}_x^-(t_k)$, or the square root equivalent $\mathbf{A}(t)$, changed to accommodate the new measurement vector $\mathbf{z}_k$, given as

$$\mathbf{z}_k = \mathbf{z}(t_k) = \begin{pmatrix} \mathbf{z}_1(t_k) \\ \vdots \\ \mathbf{z}_r(t_k) \end{pmatrix}.$$

The measurement vector in general is $\mathbf{z}(t_k) \in \mathbb{R}^M$, but consists of r individual measurement data vectors[3], $\mathbf{z}_l(t_k)$, of possibly varying dimensions. Each of these component vectors corresponds to one single sensor supplying its measurement data. In consequence, each sensor is based on its own observation model $\mathbf{h}_l$ with $\mathbf{n}_l \sim \mathcal{N}(0, \mathbf{R}_l)$, given as

$$\mathbf{z}_l(t_k) = \mathbf{h}_l(\mathbf{x}(t_k), t_k) + \mathbf{n}_l(t_k),$$

where the filter observation model is represented by a concatenation of the single sensor models as follows,

$$\mathbf{h}\left(\hat{\mathbf{X}}(t), t\right) = \begin{pmatrix} \mathbf{h}_1\left(\hat{\mathbf{X}}(t), t\right) \\ \vdots \\ \mathbf{h}_r\left(\hat{\mathbf{X}}(t), t\right) \end{pmatrix}.$$

The sigma point matrix of state estimates is then mapped to a matrix of expected measurements by the nonlinear observation process, defined by

$$\hat{\mathbf{Z}}_l(t_k) = \mathbf{h}_l\left(\hat{\mathbf{X}}_k^-, t_k\right).$$

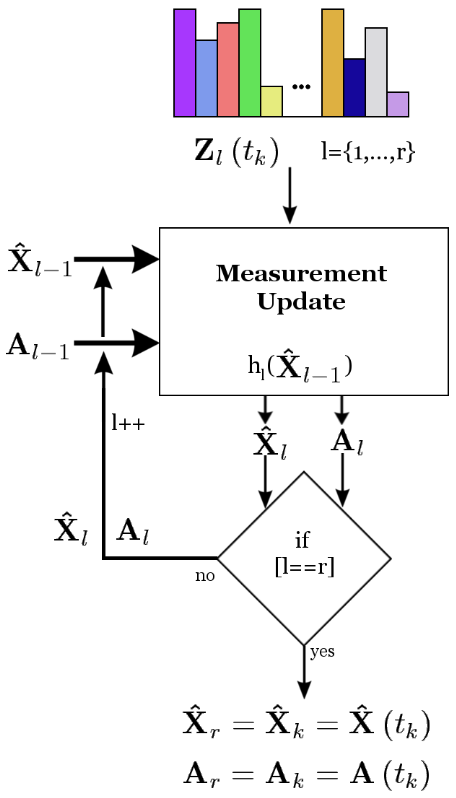

$\mathbf{Z}_l\left(t_k\right)$
l={1,...,r}
$\hat{\mathbf{X}}_{l-1}$
$\mathbf{A}_{l-1}$
Measurement Update
$h_l(\hat{\mathbf{X}}_{l-1})$
l++
$\hat{\mathbf{X}}_l$
$\mathbf{A}_l$
$\hat{\mathbf{X}}_l$
$\mathbf{A}_l$
if [l==r]
no
yes
$\hat{\mathbf{X}}_r = \hat{\mathbf{X}}_k = \hat{\mathbf{X}}\left(t_k\right)$
$\mathbf{A}_r = \mathbf{A}_k = \mathbf{A}\left(t_k\right)$

Figure 2.10: Square Root UKF Sequential Updates.
Measurement vectors $\mathbf{z}_l\left(t_k\right)$, illustrated as different colored vectors of various dimensions, are applied sequentially. The estimation variables are calculated recursively.

The design of the sequential updates is the following (illustrated in Figure 2.10): At every measurement update step, the available observation corrections are applied in the sequential order they were made. Let the number of aiding measurements be r, resulting in r small measurement updates. To denote the number of measurement updates that have been applied, the equation terms of time instant t_k are indexed with that number, e.g. $\hat{\mathbf{X}}_l$, and the index k for the time instant is removed from the formulations, since all terms in the correction step are at time step k. The initial terms for $l = 0$ are given as $\hat{\mathbf{X}}_0 = \hat{\mathbf{X}}_k^- = \hat{\mathbf{X}}^-\left(t_k\right)$, $\mathbf{P}_0 = \mathbf{P}_k^- = \mathbf{P}^-\left(t_k\right)$ or $\mathbf{A}_0 = \mathbf{A}_k^- = \mathbf{A}^-\left(t_k\right)$.

[3]Starting to count at 1, not 0, i.e. $l = 1, \ldots, r$.

For measurement $\mathbf{z}_l\left(t_k\right)$, the adjusted correction step equations for the corresponding sequential update step l are then given as:

$$\begin{aligned} \hat{\mathbf{Z}}_l &= \mathbf{h}_l\left(\hat{\mathbf{X}}_{l-1}, t_l\right), \\ \overline{\mathbf{z}}_l &= \hat{\mathbf{Z}}_l w_m, \\ \mathbf{P}_{y,l} &= \hat{\mathbf{Z}}_l W \hat{\mathbf{Z}}_l^T + \mathbf{R}_l, \\ \mathbf{P}_{xy,l} &= \hat{\mathbf{X}}_l W \hat{\mathbf{Z}}_l^T, \end{aligned} \tag{2.47}$$

and

$$\begin{aligned} \mathbf{K}_l &= \mathbf{P}_{xy,l}\left(\mathbf{P}_{y,l}\right)^{-1}, \\ \hat{\mathbf{x}}_l &= \hat{\mathbf{x}}_{l-1} + \mathbf{K}_l\left(\mathbf{z}_l - \overline{\mathbf{z}}_l\right), \\ \mathbf{P}_l &= \mathbf{P}_{l-1} - \mathbf{K}_l \mathbf{P}_{y,l}\left(\mathbf{K}_l\right)^T. \end{aligned} \tag{2.48}$$

Their square root versions are

$$\begin{aligned} \hat{\mathbf{Z}}_l &= \mathbf{h}_l\left(\hat{\mathbf{X}}_l, t_l\right), \\ \overline{\mathbf{z}}_l &= \hat{\mathbf{Z}}_l w_m, \\ \mathbf{A}_{y,l} &= \sqrt{\hat{\mathbf{Z}}_l W \hat{\mathbf{Z}}_l^T + \mathbf{R}_l}, \\ \mathbf{P}_{xy,l} &= \hat{\mathbf{X}}_l W \hat{\mathbf{Z}}_l^T, \end{aligned} \tag{2.49}$$

and

$$\begin{aligned} \mathbf{K}_l &= \mathbf{P}_{xy,l}\left(\mathbf{A}_{y,l}\right)^{-1}\left(\left(\mathbf{A}_{y,l}\right)^{-1}\right)^T, \\ \hat{\mathbf{x}}_l &= \hat{\mathbf{x}}_{l-1} + \mathbf{K}_l\left(\mathbf{z}_l - \overline{\mathbf{z}}_l\right), \\ U &= \mathbf{K}_l \mathbf{A}_{y,l}, \\ \mathbf{A}_l &= \mathbf{cholupdate}\left\{\mathbf{A}_{l-1}, U, -1\right\}. \end{aligned} \tag{2.50}$$

See the definitions of QR-Decomposition and Cholesky update in the previous section for calculating the square root efficiently. At the end of the sequential updates in the correction step, the r-th terms are set to be the posterior variables: $\hat{\mathbf{X}}_r = \hat{\mathbf{X}}_k = \hat{\mathbf{X}}\left(t_k\right)$, $\mathbf{P}_r = \mathbf{P}_k = \mathbf{P}\left(t_k\right)$ and $\mathbf{A}_r = \mathbf{A}_k = \mathbf{A}\left(t_k\right)$.

The sequential updates modification alters the observation model insofar, as it allows for an arbitrary number of measurement sensors to have their own observation model functions (and corresponding noise statistics). Additionally, the measurements will only be integrated in the measurement update step if sensor measurements are available for the filter at that time instant, enduring (and ignoring) missing measurements while applying available measurements in the correction process. This simplifies the addition of new sensors into the observation model as well as the synchronization of sensors with differing data update rates.

2.3.5 Latency Compensation

The goal of an estimator, as discussed in this paper, is to optimally estimate a system state variable, given sensor measurements of that state's indirect effects. To apply these measurements optimally, their time instances are obviously critical for the correction process if the underlying system is dynamical. In the case that a delayed measurement is applied by the observation update process to correct a state estimate at the wrong time instant, errors are introduced that should be avoided. This lagged measurement is an observation of effects of a state estimate that were valid earlier in time, but since the system has evolved during the time between the making of the observation and its usage in the correction process, the state that the measurement portrays is not the one that the system is in at the time of the correction step.

To be specific, at time instant t_k a measurement $\mathbf{z}(t_{k-n})$ with information of effects of a past state $\mathbf{x}(t_{k-n})$ is used to correct state estimate $\hat{\mathbf{x}}(t_k)$, as illustrated in Figure 2.11. Between t_{k-n} and t_k are $n-1$ correction steps, and at the n-th step the delayed

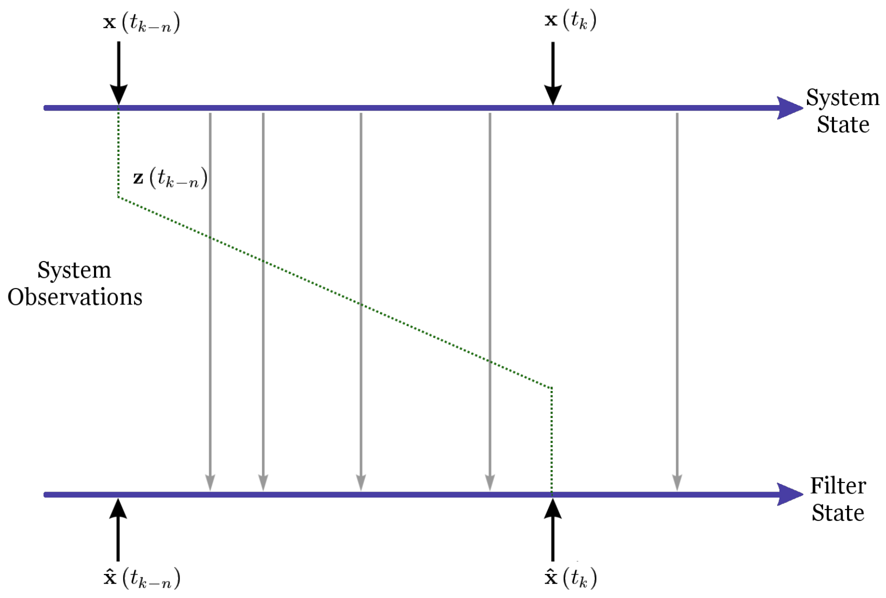


Figure 2.11: Delayed sensor measurement applied at delayed instant in time.

measurement is supposed to be used for correction purposes. An example for such delayed sensor measurements could be egomotion information from visual cues such as optic flow, which takes time to calculate depending on the resolution of the images supplied to the optic flow and egomotion calculation algorithms, as well as on the complexity of said algorithms. Another example are GPS sensors with their processing delay, calculating the Geodetic coordinates from received satellite data.

Merwe and Wan implemented an idea by Julier to accommodate for this delay at the correction process level [12]. In the implementation, they view the problem not as extrapolating a measurement forward in time, but rather as taking a cross-correlation backwards through time. To fuse the delayed measurement correctly, Equation (2.23) becomes

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \tilde{\mathbf{z}}_{k-n}, \tag{2.51}$$

where the delayed innovation is given as

$$\tilde{\mathbf{z}}_{k-n} = \mathbf{z}_{k-n} - \hat{\mathbf{z}}_{k-n}. \tag{2.52}$$

The Kalman Gain for the delayed fusion, where $lag = k - n$, is calculated as

$$\mathbf{K}_k = \mathbf{P}_{x_k \tilde{z}_{lag}} \left(\mathbf{P}_{\tilde{z}_{lag}}\right)^{-1}, \tag{2.53}$$

where the first term, given as

$$\mathbf{P}_{x_k \tilde{z}_{lag}} = E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)\left(\mathbf{z}_{lag} - \hat{\mathbf{z}}_{lag}\right)^T\right], \tag{2.54}$$

is not easily calculated. Equation (2.54) can be approximated for the UKF by

$$\mathbf{P}_{x_k \tilde{z}_{lag}} \approx \mathbf{X}_k^- W \left(\mathbf{h}\left(\hat{\mathbf{X}}_{lag}^-\right)\right)^T.$$

The maintenance of the temporal cross-covariance matrices during the time span until the delayed measurement arrives can be accomplished through augmentation of the system state and redefinition of the process and observation models. The state vector is augmented to contain the state estimate at t_{lag}, the time the lagged measurement is physically taken, in addition to the original state of the current time, $\mathbf{x}_k$, defined as

$$\mathbf{x}_k^a = \begin{pmatrix} \mathbf{x}_k \\ \mathbf{x}_{lag} \end{pmatrix}. \tag{2.55}$$

The process model is augmented to

$$\begin{aligned} \mathbf{x}_k^a &= \breve{\mathbf{f}}\left(\mathbf{x}_{k-1}^a, t_k, \mathbf{v}_k\right) && (2.56)\\ &= \begin{pmatrix} \mathbf{f}\left(\mathbf{x}_{k-1}, t_k, \mathbf{v}_k\right) \\ \mathbf{x}_{lag} \end{pmatrix}, && (2.57) \end{aligned}$$

where $\mathbf{f}(\cdot)$ is the original process model and $\mathbf{v}_k$ the original process noise. The augmented process model will therefore update the first component of the augmented state vector as before, using the original process model, while the system state of the delayed measurement, $\mathbf{x}_{lag}$, is kept constant.

Using the previous notation, let's assume that at time $t_{k-n} = t_{lag}$ the lagged sensor makes its physical measurement, but will only output the result n correction steps later at time t_k. Consequently, at time t_{k-n} the state estimate is augmented to

$$\hat{\mathbf{x}}^a_{k-n} = \begin{pmatrix} \hat{\mathbf{x}}_{k-n} \\ \hat{\mathbf{x}}_{k-n} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{x}}_{lag} \\ \hat{\mathbf{x}}_{lag} \end{pmatrix}, \tag{2.58}$$

and the covariance becomes

$$\mathbf{P}_{x^a_{k-n}} = \begin{pmatrix} \mathbf{P}_{x_{k-n}} & \mathbf{P}_{x_{k-n}} \\ \mathbf{P}_{x_{k-n}} & \mathbf{P}_{x_{k-n}} \end{pmatrix}. \tag{2.59}$$

These terms are propagated as usual through the (augmented) UKF time update equations to result in the following state and covariance estimates:

$$\hat{\mathbf{x}}^{a-}_{k-n+1} = \begin{pmatrix} \hat{\mathbf{x}}^-_{k-n+1} \\ \hat{\mathbf{x}}_{k-n} \end{pmatrix} \tag{2.60}$$

$$\mathbf{P}^-_{x^a_{k-n+1}} = \begin{pmatrix} \mathbf{P}^-_{x_{k-n+1}} & \mathbf{P}^-_{x_{k-n+1}x_{k-n}} \\ \mathbf{P}^-_{x_{k-n}x_{k-n+1}} & \mathbf{P}^-_{x_{k-n}} \end{pmatrix}. \tag{2.61}$$

The off-diagonal sub-blocks are the important cross-covariance matrices needed to fuse the time-delayed measurement using Equations (2.51), (2.52) and (2.53). Which leads to the remaining issue, the measurement update step: How should the augmented state be updated during the latency period when a normal (non time delayed) measurement arrives? One solution is to not update it at all, e.g. Equation (2.62) by using a *Schmidt-Kalman filter* that introduces an indicator matrix $\mathbf{M}$, an identity matrix where the elements not to be updated are set to zero, in our case the bottom half.

$$\mathbf{K}_k = \mathbf{M}\mathbf{P}_{x_k \tilde{z}_{k-n}} \left(\mathbf{P}_{\tilde{z}_{k-n}}\right)^{-1} \tag{2.62}$$

Another way to handle measurement updates during the latency phase is to not use the indicator matrix but apply the measurement updates normally. This results in the original state estimate $\hat{\mathbf{x}}_{lag}$ in the augmented system state not staying constant. It will be corrected based on subsequently observed information, in fact *smoothing* the state estimate $\hat{\mathbf{x}}_{lag}$, in the sense that past estimates can be improved in some way based on successive future observations, a well established fact in signal-processing filter theory [29]. In other words, while waiting for the lagged sensor to report its observation during the latency period, intermittent measurements are used to improve the past estimate of the system state that the sensor will be reporting on. This way, once the measurement arrives and the innovation is calculated with Equation (2.52), the expected measurement for the delayed observation, $\hat{\mathbf{z}}_{lag}$, will be more accurate, since the lagged state estimate, $\hat{\mathbf{x}}_{lag}$, used to calculate it will now be smoothed.

Augmented Observation Model The observation model is adjusted for the augmented state vector and covariance in such a way, that for all normal (non-delayed) measurements the observation model calculates the observation prediction from the current state estimate part of the state vector, and in the case of time instant t_k if $t_{lag} = t_{k-n}$ when the delayed measurement arrives, the correction process uses the measurement to correct the smoothed delayed state estimate part of the vector, giving the following equations for the model:

$$\begin{aligned}\hat{\mathbf{z}}(t_i) &= \breve{\mathbf{h}}(\hat{\mathbf{x}}_i^a, \mathbf{v}_i) \\ &= \begin{cases}\mathbf{h}(\hat{\mathbf{x}}_i, \mathbf{v}_i) & {\scriptstyle (i \neq lag)} \\ \mathbf{h}(\hat{\mathbf{x}}_{lag}, \mathbf{v}_{lag}) & {\scriptstyle (i = lag).}\end{cases}\end{aligned} \tag{2.63}$$

After the delayed measurement is used to correct the current estimate of the state (upper part of augmented vector), the state vector and covariance are cropped to their original sizes, discarding the dragged-along delayed state estimate and its covariance information.

The latency compensation technique introduced above assumes that the filter knows either the fixed lag that a sensor suffers or the frequency with which it delivers its measurements, so that missing deliveries can be detected. For that reason the method is sometimes called a *Fixed Lag Smoother*. In the case of a sensor's fixed latency, the latency length can be timed, as e.g. for some GPS sensors. For the other case, when the filter needs to detect that a measurement should have arrived for the correction step, a sensor event map can be built in an initialization procedure of the filter, where average firing rates for sensors are calculated from their actual behavior. Given the knowledge of the length of the fixed latency or the sensor's firing rate, a definition of an Unscented Kalman filter with latency compensation can be formulated.

SRUKF Formulation

The approach above can be applied directly to the Sigma Point Kalman filter framework, expanding the state vector to the form of Equations (2.55) and (2.58) and its covariance using Equation (2.59). From these data the sigma points are drawn as usual using the chosen sigma point selection scheme. These points are then propagated through the augmented process model, and the new sigma points and estimate error covariance are updated with the regular square root UKF equations. In the correction step, properly constructed observation models predict the measurements from these sigma points, incorporating the time-varying nature of measurement vectors.

Let's assume that the state vector is expanded in the instant of time t_{lag}, so the mean state estimate $\bar{\mathbf{x}}_{lag}$ as well as the matrix square root $\mathbf{A}$ of its covariance are augmented as given in Equations (2.58) and (2.59). During the times when one or more state estimates for delayed measurements are maintained, the only thing that changes are the dimensions of some vectors and matrices as well as the process and observation models to their augmented complements. During times without maintenance of a delayed estimate term, the UKF formulations are the same, only the mentioned dimensions are back to normal. The discrete time Square Root Unscented Kalman filter formulations are given below as seen during expectation of delayed measurements:

1. From the augmented state and square root covariance the sigma points are drawn according to Equations (2.34)-(2.36), forming the matrix $\mathbf{X}\left(t_{lag}\right)$, given as

$$\begin{aligned}\bar{\mathbf{x}}_{k-1} &= \begin{pmatrix}\hat{\mathbf{x}}_{lag}\\ \hat{\mathbf{x}}_{lag}\end{pmatrix},\\ \mathbf{X}_{k-1} &= \left[\bar{\mathbf{x}}_{k-1} \quad \bar{\mathbf{x}}_{k-1}+\gamma\mathbf{A}_{k-1} \quad \bar{\mathbf{x}}_{k-1}-\gamma\mathbf{A}_{k-1}\right].\end{aligned} \tag{2.64}$$

2. Next, the sigma points are propagated through the augmented process model from Equation (2.57), their mean and square root covariance matrix updated and the new sigma points, $\mathbf{X}^{-}_{lag+1}$, calculated[4] as

$$\begin{aligned}\mathbf{X}'_k &= \breve{\mathbf{f}}\left(\mathbf{X}_{k-1}, t_k, \mathbf{v}_k\right),\\ \bar{\mathbf{x}}^{-}_k &= \hat{\mathbf{X}}'_k w_m,\\ \mathbf{A}'_k &= \mathbf{qr}\left\{\sqrt{w^c_1}\left(\hat{\mathbf{X}}_k-\bar{\mathbf{x}}^{-}_k\right)_{1:2N,k} \quad \sqrt{\mathbf{Q}}\right\},\\ \mathbf{A}^{-}_k &= \mathbf{cholupdate}\left\{\mathbf{A}'_k, \left(\hat{\mathbf{X}}_k-\bar{\mathbf{x}}^{-}_k\right)_{0,k}, w^c_0\right\}.\end{aligned} \tag{2.65}$$

3. The sigma points are recalculated using the predicted mean and estimate error covariance square root matrix in the following way:

$$\mathbf{X}^{-}_k = \left[\bar{\mathbf{x}}^{-}_k \quad \bar{\mathbf{x}}^{-}_k+\gamma\mathbf{A}^{-}_k \quad \bar{\mathbf{x}}^{-}_k-\gamma\mathbf{A}^{-}_k\right]. \tag{2.66}$$

4. These predicted sigma points are then propagated through the (augmented) observation model from Equation (2.63) to calculate the expected measurements, the mean of predicted measurements, the innovation covariance square root as well as the cross-covariance, given as

$$\begin{aligned}\bar{\mathbf{z}}_k &= \hat{\mathbf{Z}}_k w_m = \breve{\mathbf{h}}\left(\mathbf{X}^{-}_k, t_k, \mathbf{n}_k\right) w_m,\\ \mathbf{A}'_{y,k} &= \mathbf{qr}\left\{\sqrt{w^c_1}\left(\hat{\mathbf{Z}}_k-\bar{\mathbf{z}}_k\right)_{1:2N,k} \quad \sqrt{\mathbf{R}}\right\},\\ \mathbf{A}_{y,k} &= \mathbf{cholupdate}\left\{\mathbf{A}'_{y,k}, \left(\hat{\mathbf{Z}}_k-\bar{\mathbf{z}}_k\right)_{0,k}, w^c_0\right\},\\ \mathbf{P}_{xy,k} &= \mathbf{X}^{-}_k W\left(\hat{\mathbf{Z}}_k\right)^T.\end{aligned} \tag{2.67}$$

5. This enables the calculation of the Kalman gain and the subsequent correction of the state estimate using the innovation gained from the current measurement, $\tilde{\mathbf{z}}_k$, no

[4]The “prime” operator of $\mathbf{X}'$ only signifies an auxiliary matrix different from $\mathbf{X}$, it is not used as denotation for first derivative.

matter if delayed or not.

$$\begin{aligned}
\mathbf{K}_k &= \mathbf{P}_{xy,k}\left(\left(\mathbf{A}_{y,k}\right)^{-1}\right)^T\left(\mathbf{A}_{y,k}\right)^{-1},\\
\hat{\mathbf{x}}_k &= \bar{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \bar{\mathbf{z}}_k\right),\\
\mathbf{U} &= \mathbf{K}_k\mathbf{A}_{y,k},\\
\mathbf{A}_k &= \textbf{cholupdate}\left\{\mathbf{A}_k^-, U, -1\right\}.
\end{aligned} \tag{2.68}$$

6. If the measurement, $\mathbf{z}_k$, is a delayed measurement, then its corresponding lagged state estimate being maintained in the augmented state vector is now obsolete, as well as the information contained in the augmented covariance square root. Therefore, the posterior state estimate vector along with the covariance square root matrix are cropped to remove the obsolete information as follows,

$$\begin{aligned}
\hat{\mathbf{x}}_k^a &= \begin{pmatrix}\hat{\mathbf{x}}_k\\ \hat{\mathbf{x}}_{lag}\end{pmatrix} \Rightarrow \hat{\mathbf{x}}_k,\\
\mathbf{A}_{x_k^a}^a &= \begin{pmatrix}\mathbf{A}_{x_k} & \mathbf{A}_{x_k x_{lag}}\\ \mathbf{A}_{x_{lag}x_k} & \mathbf{A}_{x_{lag}}\end{pmatrix} \Rightarrow \mathbf{A}_{x_k}.
\end{aligned} \tag{2.69}$$

7. Recursion Step: The posterior mean and covariance square root calculated (and cropped) above are subsequently used as prior variables for the next recursion step, calculating the sigma points for the next time update.

The SR-UKF with latency compensation is summarized in Algorithm C.7.

Chapter 3

Process and Observation Models

3.1 Introduction

The main purpose compelling an autonomous mobile entity within some environment to “think” or compute is the question *What do I want to do next?* For robots, this task is the main force driving computations about consideration of possible actions. They compare the predicted consequences of their options to decide which to take. While for human beings this often happens subconsciously, there are many occurrences of logical reasoning and comparisons of possible next steps. To properly decide which action to take next, the consequences of that action must be known or must be able to be predicted by the autonomous agent. In order to conjecture the effects of actions on both the agent as well as its environment, the development of physical models of the agent and its environment is necessary.

A model is a representation or description of an object or system. It is designed to show the workings of the object or system using a certain descriptive expression. That type of expression can be a plan properly defined in written language, or it could be a scaled sculptural or mechanical replication of the original. The choice always depends on what aspect of the object or system is supposed to be modeled.

In physics along with robotics, models of objects and systems are usually formulated mathematically. These models should be a representation as accurate as possible or necessary of the material properties, characteristics and dynamics of the original. Such models are also called physical models, as they regard the original from the physics’ point of view, i.e. describing its physical properties.

In the particular case of mobile robots, the goal is to describe the dynamics of the robot in its specific environment mathematically, with the intent that algorithms deliver precise predictions and estimations on the robot’s motion trajectories. For a robot to be able to give accurate predictions, it needs to assemble information on consequences of its actions that are not completely defined in the model, as well as perceive the current state of the dynamic system or environment it is in. To that effect a robot is equipped with an arbitrary number of sensors appropriate for its scope of duties. The mapping of these sensors' data to the model's interior representation is described by the physical model. Though sometimes it is separated into an extra formulation as part of the complete model, to better cooperate with an existing estimation framework.

The physical models necessary for robots to predict the development of their egomotion over time on one side, and on the other side collect information on the environment and the effects of their actions on it, are called the *Process and Observation Models*, respectively. Both models are derived in this chapter, after explaining the mathematical tools and conventions used in their formulations.

3.1.1 Chapter Outline

This chapter covers the mathematical representations of the dynamic system of a quadrocopter in its environment. The dynamic properties of the robot as well as the mappings of external influences of the environment or information on the environment to the robot are derived.

First, the mathematical notation used to simplify the angular and linear equations is introduced. While this notation, called *Spatial Vector Notation*, is powerful and interposes its own algebra with changes to some vector and matrix operations, only a small part of its versatility is needed and used. Only these decisive parts are explained.

Once the spatial algebra is established, the different frames of reference are defined: the body reference frame and the world reference frame. With the reference frames established, the representation of orientation is treated. Instead of Euler angles, quaternion notation is used and its advantages are presented.

To understand the technique quadrocopters employ to affect their flight motion, quadrocopter control is explained in detail.

Having introduced all required mathematical notations and methods, the system state of the unscented Kalman filter is defined. Its components as well as their respective reference frames are determined, followed by the derivations of the process model and the observation models of available sensors integrated in the filter.

3.2 Prerequisites

In order to understand the derivation of the process and observation models, a basic knowledge from several disciplines is required. This section introduces the reader to some of that prerequisite knowledge in a short, summarized way.

3.2.1 Spatial Algebra

To describe the dynamics of a robot system, the dynamic equations of motion provide the relationships between actuation and contact forces acting on the robot and the acceleration and motion trajectories that result. In usual mathematical notation, separate equations for angular and linear components are used to describe the complete dynamics of the system. This results in a complex set of formulations that need to be implemented for the robot. This complexity is reflected in the implementation, resulting in code that is not as efficient as it could be. In addition to the need for computational efficiency, clarity and ease of development are important factors for formulation of algorithms. To this extent, spatial vector notation has been applied to effectively simplify the formulations of the dynamic equations of motion. Spatial vector algebra is a concise vector notation for describing rigid-body acceleration, velocity, inertia, etc. using six-dimensional vectors and tensors.

There is no single standard in robotics for notation of robot dynamics. Various notations are used including 3-D vectors, 4×4 matrices and several types of 6-D vector notations. In general, six-dimensional formulations are the best, offering both a cleaner and more compact alternative to 3-D vectors and a more powerful notation than 4×4 matrices. Therefore, 6-D vectors are used in the description of the robot dynamics. Specifically, the spatial vector algebra described in [30] is used to formulate the dynamic equations of motion.

In this thesis, vectors are usually represented as bold letters (e.g. $\mathbf{z}, \mathbf{v}$). To differentiate between vectors of arbitrary dimensions and spatial vectors, spatial vectors will be denoted using bold letters (e.g. $\mathbf{z}, \mathbf{v}$), while arbitrary (3-D or other) vectors will be denoted using bold italic letters (e.g. $\boldsymbol{z}, \boldsymbol{v}$). In addition, *only in this section on spatial algebra*, the coordinate vectors are underlined to distinguish them from the vectors they represent (e.g. $\underline{\boldsymbol{z}}$ and $\underline{\mathbf{z}}$, representing $\boldsymbol{z}$ and $\mathbf{z}$).

Motion and Force

Based on mathematical reasons, distinguishing between those vectors that represent motions of rigid objects and those representing the forces acting on them is helpful. Therefore, in spatial vector algebra motion vectors are placed in a vector space called M^6, and force vectors in a space called F^6. This assists in defining mathematical operators acting on motion and force vectors for a compact formulation. Motion vectors describe quantities such as acceleration, velocity, directions of motion freedom; force vectors describe force, momentum, contact normals, et cetera.

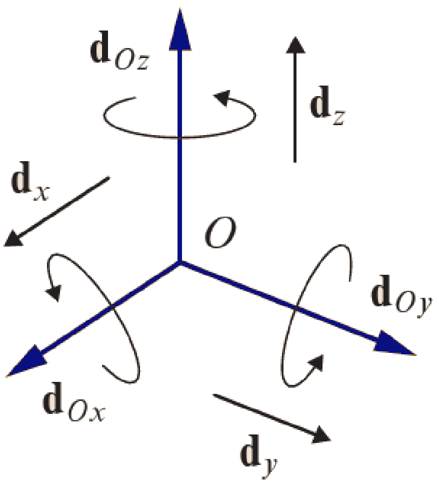
Basis Vectors

In 3-D vector space, suppose that $\boldsymbol{v}$ is a 3-D vector and $\underline{\boldsymbol{v}} = (v_x, v_y, v_z)^T$ is the Cartesian coordinate vector representing $\boldsymbol{v}$ in an orthonormal basis $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}\}$. The relationship between $\boldsymbol{v}$ and $\underline{\boldsymbol{v}}$ is then given by

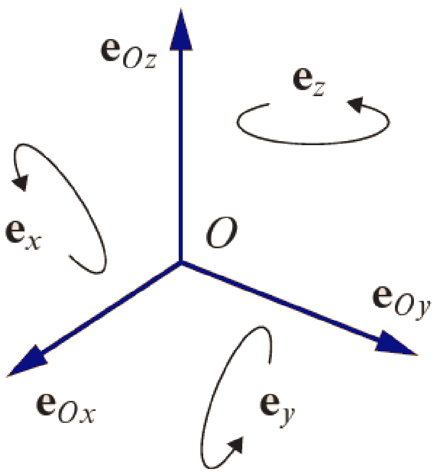
$$\boldsymbol{v} = \hat{\boldsymbol{x}} v_x + \hat{\boldsymbol{y}} v_y + \hat{\boldsymbol{z}} v_z.$$

For spatial vectors the same idea applies, only instead of Cartesian coordinate systems we use Plücker coordinate systems.

Plücker coordinates are six-dimensional vectors fixed by a Plücker basis. This results in a total of 12 basis vectors, six for the motion vector space and six for the force vector space. Given a Cartesian coordinate frame, O_{xyz}, the Plücker basis vectors are defined as follows: three unit rotations about the directed lines $\overline{Ox}, \overline{Oy}, \overline{Oz}$, denoted by $\mathbf{d}_{\overline{Ox}}, \mathbf{d}_{\overline{Oy}}, \mathbf{d}_{\overline{Oz}}$; three unit translations in the directions x, y, z, denoted by $\mathbf{d}_x, \mathbf{d}_y, \mathbf{d}_z$; three unit couples about the directions x, y, z denoted by $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$; and three unit forces in the direction of the directed lines $\overline{Ox}, \overline{Oy}, \overline{Oz}$ denoted by $\mathbf{e}_{\overline{Ox}}, \mathbf{e}_{\overline{Oy}}, \mathbf{e}_{\overline{Oz}}$. Figure 3.1 illustrates these relationships.



(a) Plücker basis vectors for motion vector space.



(b) Plücker basis vectors for force vector space.

Figure 3.1: Plücker coordinate frames.

Spatial Velocity and Force

Traditionally, given any point O, the velocity of a rigid body can be described by two 3-D vectors, $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$, representing the angular velocity and the linear velocity of the body-fixed point currently at O, respectively. It is important to note that $\boldsymbol{v}_O$ is not the velocity of O itself, but only the velocity of the body-fixed point that happens to coincide with O at the current instant.

In spatial vector notation, the velocity of this same rigid body can be described by only a single spatial motion vector, $\mathbf{v} \in \mathsf{M}^6$. To obtain $\mathbf{v}$ from $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$, a Cartesian coordinate frame O_{xyz} with its origin at O is introduced first. This frame defines a Cartesian coordinate system for $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$ as well as a Plücker coordinate system for $\mathbf{v}$. From these coordinate systems, the following relationship can be shown:

$$\mathbf{v} = \mathbf{d}_{\overline{Ox}}\omega_x + \mathbf{d}_{\overline{Oy}}\omega_y + \mathbf{d}_{\overline{Oz}}\omega_z + \mathbf{d}_x v_{\overline{Ox}} + \mathbf{d}_y v_{\overline{Oy}} + \mathbf{d}_z v_{\overline{Oz}}. \tag{3.1}$$

where $\omega_x, \ldots, v_{\overline{Oz}}$ are the Cartesian coordinates of $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$ in O_{xyz}. Consequently, the Plücker coordinates of $\mathbf{v}$ are the concatenated Cartesian coordinates of $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$. The coordinate vector representing $\mathbf{v}$ in O_{xyz} can be written as

$$\underline{\mathbf{v}}_O = \begin{pmatrix} \omega_x \\ \vdots \\ v_{\overline{Oz}} \end{pmatrix} = \begin{pmatrix} \underline{\boldsymbol{\omega}} \\ \underline{\boldsymbol{v}}_O \end{pmatrix}. \tag{3.2}$$

As a convenient abbreviation of the list of Plücker coordinates, the notation on the far right of equation (3.2) can be used.

Spatial force vectors are defined in an analogous way. Again, given any point O, any system of forces acting on a single rigid body can be expressed equivalently in two components, a single force $\boldsymbol{f}$ acting on a line passing through O, together with a pure couple, $\boldsymbol{n}_O$, as the moment of the force system around O. In this manner, these two vectors describe the forces acting on a rigid body in much the same way that $\boldsymbol{\omega}$ and $\boldsymbol{v}_O$ describe its velocity. The same system of forces can be described by a single spatial force vector, $\mathbf{f} \in \mathsf{F}^6$. Going back to the frame O_{xyz}, it can be shown that

$$\mathbf{f} = \mathbf{e}_x n_{\overline{Ox}} + \mathbf{e}_y n_{\overline{Oy}} + \mathbf{e}_z n_{\overline{Oz}} + \mathbf{e}_{\overline{Ox}} f_x + \mathbf{e}_{\overline{Oy}} f_y + \mathbf{e}_{\overline{Oz}} f_z, \tag{3.3}$$

where $n_{\overline{Ox}}, \ldots, f_z$ are the Cartesian coordinates of $\boldsymbol{n}_O$ and $\boldsymbol{f}$ in O_{xyz}. The coordinate vector corresponding to $\mathbf{f}$ is then written as

$$\underline{\mathbf{f}}_O = \begin{pmatrix} n_{\overline{Ox}} \\ \vdots \\ f_z \end{pmatrix} = \begin{pmatrix} \boldsymbol{n}_O \\ \boldsymbol{f} \end{pmatrix}. \tag{3.4}$$

These are the Plücker coordinates of $\mathbf{f}$ in O_{xyz} and the rightmost notation can be used as a convenient short-form for the list of Plücker coordinates.

Addition and Scalar Multiplication

Addition and scalar multiplication of spatial vectors is the same as it is for arbitrary vectors: element-wise. That means that if, for example two forces, $\mathbf{f}_1$ and $\mathbf{f}_2$, act on the same rigid body, the resultant force is $\mathbf{f} = \mathbf{f}_1 + \mathbf{f}_2$; similarly if two distinct rigid bodies have velocities of $\mathbf{v}_1$ and $\mathbf{v}_2$, then the velocity of the second body relative to the first is the difference $\mathbf{v}_2 - \mathbf{v}_1$. Furthermore, if $\mathbf{f}$ denotes a force of 1N acting along a certain line in space, then $\alpha\mathbf{f}$ denotes a force of αN acting along the same line. Addition between motion and force vectors is not allowed, since it does not exist in physics.

The scalar product between a spatial motion and force vector, $\mathbf{m}$ and $\mathbf{f}$, on the other hand is defined, unlike products such as $\mathbf{f} \cdot \mathbf{f}$ and $\mathbf{m} \cdot \mathbf{m}$. The scalar product obviously results in a scalar value and is given by

$$\mathbf{f} \cdot \mathbf{m} = \mathbf{m} \cdot \mathbf{f} = \underline{\mathbf{m}}^T \underline{\mathbf{f}}.$$

Coordinate Transformations

For motion and force vectors different coordinate transformation rules apply. Let A and B be two coordinate frames, where each defines a coordinate frame of the same name. Likewise, let $\underline{\mathbf{m}}_A$, $\underline{\mathbf{m}}_B$, $\underline{\mathbf{f}}_A$ and $\underline{\mathbf{f}}_B$ be coordinate vectors representing the spatial vectors $\mathbf{m} \in \mathsf{M}^6$ and $\mathbf{f} \in \mathsf{F}^6$ in A and B coordinates, respectively. The coordinate transformation rules are then

$$\underline{\mathbf{m}}_B = {}^B\mathbf{X}_A \underline{\mathbf{m}}_A \tag{3.5}$$

and

$$\underline{\mathbf{f}}_B = {}^B\mathbf{X}_A^F \underline{\mathbf{f}}_A, \tag{3.6}$$

where ${}^B\mathbf{X}_A$ and ${}^B\mathbf{X}_A^F$ are the transformation matrices from coordinate system A to B for motion and force vectors, respectively. The transformation matrices are related by the identity

$${}^B\mathbf{X}_A^F \equiv \left({}^B\mathbf{X}_A\right)^{-T} \equiv \left({}^A\mathbf{X}_B\right)^T. \tag{3.7}$$

As illustrated in Figure 3.2, let the position and orientation of frame B relative to frame A

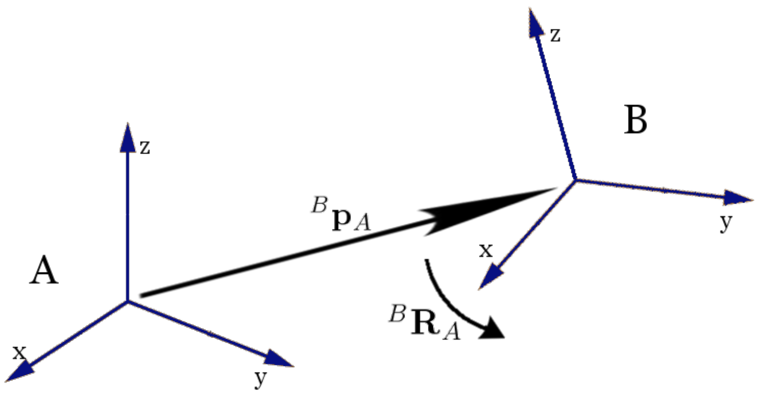


Figure 3.2: Coordinate Transformation from frame A to B.
Position vector ${}^B\boldsymbol{p}_A$ of coordinate frame B relative to A, 3×3 rotation matrix ${}^B\boldsymbol{R}_A$ describing the rotation of B relative to A.

be described by a 3-D position vector ${}^B\boldsymbol{p}_A$ and a 3×3 rotation matrix ${}^B\boldsymbol{R}_A$, respectively. The formula for the transformation matrix from A to B, ${}^B\mathbf{X}_A$, is then given as

$$\begin{aligned} {}^B\mathbf{X}_A &= \begin{pmatrix} {}^B\boldsymbol{R}_A & \mathbf{0} \\ \mathbf{0} & {}^B\boldsymbol{R}_A \end{pmatrix} \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ -\boldsymbol{S}\left({}^B\boldsymbol{p}_A\right) & \mathbf{1} \end{pmatrix} \\ &= \begin{pmatrix} {}^B\boldsymbol{R}_A & \mathbf{0} \\ {}^B\boldsymbol{R}_A \boldsymbol{S}\left({}^B\boldsymbol{p}_A\right)^T & {}^B\boldsymbol{R}_A \end{pmatrix}, \end{aligned} \tag{3.8}$$

and its inverse is

$$\begin{aligned} {}^{A}\mathbf{X}_{B} &= \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \boldsymbol{S}\left({}^{B}\boldsymbol{p}_{A}\right) & \mathbf{1} \end{pmatrix} \begin{pmatrix} {}^{A}\boldsymbol{R}_{B} & \mathbf{0} \\ \mathbf{0} & {}^{A}\boldsymbol{R}_{B} \end{pmatrix} \\ &= \begin{pmatrix} {}^{A}\boldsymbol{R}_{B} & \mathbf{0} \\ \boldsymbol{S}\left({}^{B}\boldsymbol{p}_{A}\right) {}^{A}\boldsymbol{R}_{B} & {}^{A}\boldsymbol{R}_{B} \end{pmatrix}. \end{aligned} \tag{3.9}$$

The sub-block $\boldsymbol{S}\left(\underline{\boldsymbol{p}}\right)$ of the transformation matrix is the 3×3 skew-symmetric matrix formulation of the vector cross-product $\boldsymbol{S}\left(\underline{\boldsymbol{p}}\right)\underline{\boldsymbol{v}} = \underline{\boldsymbol{p}} \times \underline{\boldsymbol{v}}$, where $\boldsymbol{S}\left(\underline{\boldsymbol{p}}\right)$ is defined as

$$\boldsymbol{S}\left(\underline{\boldsymbol{p}}\right) = \begin{pmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{pmatrix}. \tag{3.10}$$

Spatial force vectors obey different transformation rules to motion vectors. The transformation matrix for force vectors corresponding to the transformation from frame A to frame B, ${}^{B}\mathbf{X}_{A}^{F}$, is given by the relationship

$$\begin{aligned} {}^{B}\mathbf{X}_{A}^{F} \equiv \left({}^{B}\mathbf{X}_{A}\right)^{-T} &\equiv \left({}^{A}\mathbf{X}_{B}\right)^{T} \\ &\equiv \begin{pmatrix} {}^{A}\boldsymbol{R}_{B}^{T} & \left(\boldsymbol{S}\left({}^{B}\boldsymbol{p}_{A}\right) {}^{A}\boldsymbol{R}_{B}\right)^{T} \\ \mathbf{0} & {}^{A}\boldsymbol{R}_{B}^{T} \end{pmatrix} \\ &\equiv \begin{pmatrix} {}^{B}\boldsymbol{R}_{A} & -\boldsymbol{S}\left({}^{B}\boldsymbol{p}_{A}\right) {}^{B}\boldsymbol{R}_{A} \\ \mathbf{0} & {}^{B}\boldsymbol{R}_{A} \end{pmatrix}. \end{aligned} \tag{3.11}$$

Spatial Acceleration

The definition of spatial acceleration is given as the rate of change of spatial velocity. This means that spatial acceleration is not equal to the classical definition of rigid-body acceleration, or *classical acceleration* here. The essential difference is the following:

$$\underline{\mathbf{a}} = \begin{pmatrix} \underline{\dot{\boldsymbol{\omega}}} \\ \underline{\dot{\boldsymbol{v}}}_{O} \end{pmatrix} \qquad \text{and} \qquad \underline{\mathbf{a}}' = \begin{pmatrix} \underline{\dot{\boldsymbol{\omega}}} \\ \underline{\dot{\boldsymbol{v}}}'_{O} \end{pmatrix}, \tag{3.12}$$

where $\underline{\mathbf{a}}$ represents the spatial acceleration, $\underline{\mathbf{a}}'$ is the classical acceleration, $\underline{\dot{\boldsymbol{v}}}_{O}$ is the derivative of $\underline{\boldsymbol{v}}_{O}$ taking O as fixed in space, while $\underline{\dot{\boldsymbol{v}}}'_{O}$ is the derivative of $\underline{\boldsymbol{v}}'_{O}$ taking O to be fixed in the body. This difference in the fixing of O in space or in the body gives the relation between spatial and classical acceleration as

$$\underline{\mathbf{a}}' = \underline{\mathbf{a}} + \begin{pmatrix} \mathbf{0} \\ \underline{\boldsymbol{\omega}} \times \underline{\boldsymbol{v}}_{O} \end{pmatrix}. \tag{3.13}$$

Taking the derivatives of a position vector $\boldsymbol{r}$ representing the position of the body-fixed point at O relative to any fixed point, we get

$$\begin{aligned} \boldsymbol{v}_{O} &= \dot{\boldsymbol{r}}, \\ \dot{\boldsymbol{v}}'_{O} &= \ddot{\boldsymbol{r}}, \\ \dot{\boldsymbol{v}}_{O} &= \ddot{\boldsymbol{r}} - \boldsymbol{\omega} \times \boldsymbol{v}_{O}. \end{aligned}$$

This property of spatial acceleration results in easier application. Giving an example from [31], if two bodies B_1 and B_2 have velocities $\mathbf{v}_1$ and $\mathbf{v}_2$, respectively, and $\mathbf{v}_{rel}$ is the relative velocity of B_2 with respect to B_1, then

$$\mathbf{v}_2 = \mathbf{v}_1 + \mathbf{v}_{rel}.$$

Now, the correlation between their spatial accelerations can be calculated by simply differentiating the velocity formula:

$$\frac{d}{dt}\left(\mathbf{v}_2 = \mathbf{v}_1 + \mathbf{v}_{rel}\right) \quad \Rightarrow \quad \mathbf{a}_2 = \mathbf{a}_1 + \mathbf{a}_{rel}.$$

Note that there are no Coriolis or centrifugal effects and terms to worry about. Spatial accelerations are composed only of additions exactly like velocities. This represents a significant improvement over the formulation for calculation of classical acceleration.

In the modelling of the dynamic process and observation system equations, this characteristic of spatial acceleration will not be needed, but was given for reasons of completeness and to show its advantages over classical acceleration.

Spatial Inertia

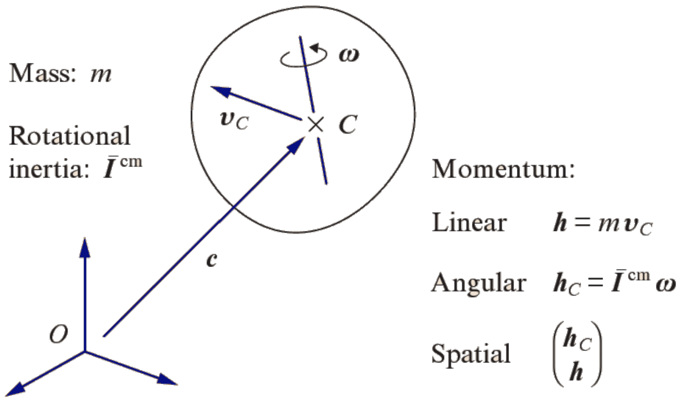

Mass: m
Rotational inertia: $\bar{\boldsymbol{I}}^{\text{cm}}$
$\boldsymbol{\omega}$
$\boldsymbol{v}_C$
C
$\boldsymbol{c}$
O
Momentum:
Linear $\boldsymbol{h} = m\boldsymbol{v}_C$
Angular $\boldsymbol{h}_C = \bar{\boldsymbol{I}}^{\text{cm}}\boldsymbol{\omega}$
Spatial $\begin{pmatrix}\boldsymbol{h}_C\\ \boldsymbol{h}\end{pmatrix}$

Figure 3.3: Spatial Momentum and Inertia

The spatial momentum of a rigid body, $\mathbf{h}$, is given by the concatenation of its angular and linear momenta (Figure 3.3). The momenta are the products of the angular and linear inertia with the body's angular and linear velocities, respectively. Suppose that this rigid body has a mass of m, with a center of mass at point in space C and rotational inertia $\boldsymbol{I}^{cm}$ about C and is traveling with linear velocity $\boldsymbol{v}_C$ and angular velocity $\boldsymbol{\omega}$, denoted as spatial velocity $\underline{\mathbf{v}}_C = \left(\boldsymbol{\omega}^T \boldsymbol{v}_C^T\right)^T$. Then its linear momentum, $\boldsymbol{h} = m\boldsymbol{v}_C$, and its angular momentum, $\boldsymbol{h}_C = \boldsymbol{I}^{cm}\boldsymbol{\omega}$, describe the spatial momentum of the rigid body with respect to the point C as follows,

$$\underline{\mathbf{h}}_C = \begin{pmatrix} \underline{\boldsymbol{h}}_C \\ \underline{\boldsymbol{h}} \end{pmatrix} = \begin{pmatrix} \overline{\boldsymbol{I}}^{cm}\underline{\boldsymbol{\omega}} \\ m\underline{\boldsymbol{v}_C} \end{pmatrix}.$$

For the angular momentum around any general point O it is true that $\boldsymbol{h}_O = \boldsymbol{h}_C + \boldsymbol{c} \times \boldsymbol{h}$, where $\boldsymbol{c} = \overrightarrow{OC}$. This leads to

$$\underline{\mathbf{h}}_O = \begin{pmatrix} \underline{\boldsymbol{h}}_O \\ \underline{\boldsymbol{h}} \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \boldsymbol{S}(\underline{\boldsymbol{c}}) \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \underline{\mathbf{h}}_C. \tag{3.14}$$

Since the spatial momentum is the product of spatial inertia and spatial velocity, $\mathbf{h}_C = \mathbf{I}_C\mathbf{v}_C$, this implies

$$\mathbf{I}_C = \begin{pmatrix} \overline{\boldsymbol{I}}^{cm} & \mathbf{0} \\ \mathbf{0} & m\mathbf{1} \end{pmatrix}. \tag{3.15}$$

This formulation for spatial inertia of a rigid body, expressed at its center of mass, together with Equation (3.14) enable the derivation of a formulation of the spatial inertia for any point O, given as

$$\begin{aligned}\underline{\mathbf{h}}_O &= \begin{pmatrix} \mathbf{1} & \boldsymbol{S}(\underline{\boldsymbol{c}}) \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} \overline{\boldsymbol{I}}^{cm} & \mathbf{0} \\ \mathbf{0} & m\mathbf{1} \end{pmatrix} \underline{\mathbf{v}}_C \\ &= \begin{pmatrix} \mathbf{1} & \boldsymbol{S}(\underline{\boldsymbol{c}}) \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} \overline{\boldsymbol{I}}^{cm} & \mathbf{0} \\ \mathbf{0} & m\mathbf{1} \end{pmatrix} \begin{pmatrix} \mathbf{1} & \mathbf{0} \\ \boldsymbol{S}(\underline{\boldsymbol{c}})^T & \mathbf{1} \end{pmatrix} \underline{\mathbf{v}}_O \\ &= \begin{pmatrix} \overline{\boldsymbol{I}}^{cm} + m\boldsymbol{S}(\underline{\boldsymbol{c}})\boldsymbol{S}(\underline{\boldsymbol{c}})^T & m\boldsymbol{S}(\underline{\boldsymbol{c}})^T \\ m\boldsymbol{S}(\underline{\boldsymbol{c}})^T & m\mathbf{1} \end{pmatrix} \underline{\mathbf{v}}_O.\end{aligned}$$

Since $\underline{\mathbf{h}}_O = \mathbf{I}_O\underline{\mathbf{v}}_O$, the spatial inertia for any point O is given as

$$\mathbf{I}_O = \begin{pmatrix} \overline{\boldsymbol{I}}^{cm} + m\boldsymbol{S}(\underline{\boldsymbol{c}})\boldsymbol{S}(\underline{\boldsymbol{c}})^T & m\boldsymbol{S}(\underline{\boldsymbol{c}})^T \\ m\boldsymbol{S}(\underline{\boldsymbol{c}})^T & m\mathbf{1} \end{pmatrix}, \tag{3.16}$$

which can also be written as

$$\mathbf{I}_O = \begin{pmatrix} \overline{\boldsymbol{I}}_O & m\boldsymbol{S}(\underline{\boldsymbol{c}})^T \\ m\boldsymbol{S}(\underline{\boldsymbol{c}})^T & m\mathbf{1} \end{pmatrix}, \tag{3.17}$$

where $\overline{\boldsymbol{I}}_O = \overline{\boldsymbol{I}}^{cm} + m\boldsymbol{S}(\underline{\boldsymbol{c}})\boldsymbol{S}(\underline{\boldsymbol{c}})^T$ is the angular inertia of the rigid body around O.

With respect to this thesis, the same is true for spatial inertia as for spatial acceleration above: The difference between classical and spatial inertia is not necessary in the actual formulations of the models. These apply only if the motion of the rigid body is described in an inertial frame of reference, not a non-inertial body-fixed frame of reference. The robot's inertia is given with respect to the robot's center of gravity, which is the origin of the body frame. Thereby the translation vector $\boldsymbol{c}$ in Equation (3.17) becomes zero. Furthermore, during the time update the body frame is considered stationary, turning $\mathbf{I}_O$ into a constant diagonal matrix.

More on spatial vector algebra and its applications can be found in the book by Roy Featherstone on dynamics algorithms in robotics [30]. The different reference frames are explained in the next section.

3.2.2 Reference Frames

The goal of localization, a subset of navigation, is being able to tell one's position and orientation with respect to some point of origin. In addition, for navigation, knowledge of properties concerning egomotion, acceleration and locations of other objects, for example obstacles, is fundamental. To be able to relate one coordinate information to another, these need to be grounded in coordinate frames, or frames of reference. These frames can have multiple coordinate systems attached as references, each with its own units and basis. If the information to be related is not based in the same frame, then the relationship between the different frames, the *coordinate transformation*, must be known in order to transform and compare the different based values.

This relationship can only be computed if the distinct frames are grounded to a common point and frame of reference. In mobile robotics, usually the reference frame for absolute values concerning motion and location is the inertial reference frame with its origin in the center of the Earth. It is often helpful for mathematical formulations and computational complexity to define additional coordinate frames and systems. Before presenting the specific definitions of the body and world reference frames, reference frames in general will be introduced.

Coordinate or reference frames can be static or dynamic, but always relative to some other reference frame, e.g. a coordinate system that is grounded at the center of mass of a moving robot regards itself as static, but views a reference frame with origin at the center of the Earth as moving the inverted actual speed of the robot.

A simple example is given in Figure 3.4, illustrating two persons looking at each other, each on the other side of a street. In the reference frame of each person, the directions *left*, *right* and *forward* are all the inversions to the directions in the reference frame of the other person. Only a coordinate frame grounded in a shared environment equally relevant to both persons, in this example the magnetic directions of North, East, South and West as given by the compass, is identical to both persons, even though they themselves are at different positions and orientations in their shared environment.

In the definition of a frame of reference, one can distinguish between *observational frames of reference* and *coordinate systems*. The term *observational state of reference* is usually used if the *state of motion* within the frame is addressed, rather than the choice of coordinates. On the other hand, a coordinate system can be used for many different applications, where the state of motion need not be important.

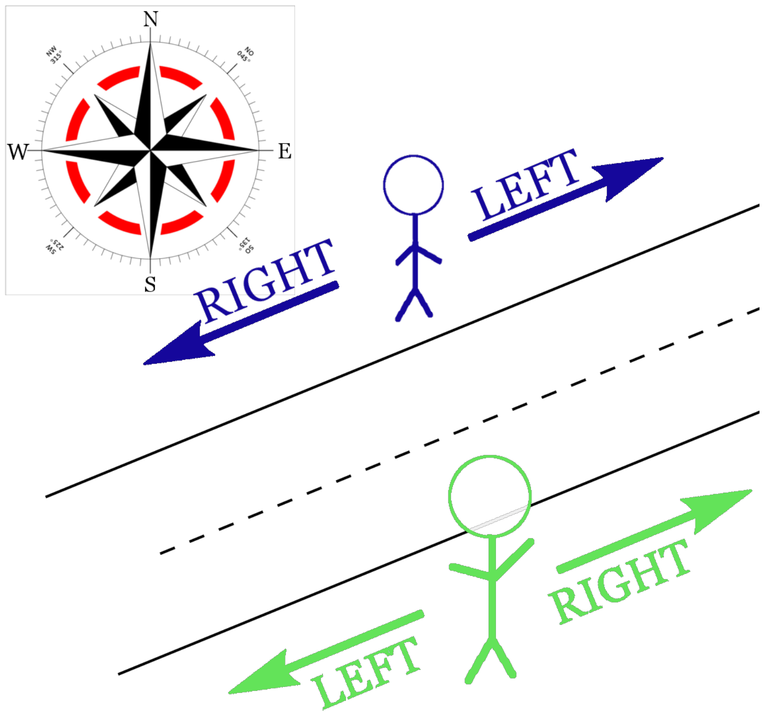


Figure 3.4: Different Reference Frames.
Two observational frames of reference are based in different persons that are rotated and translocated with respect to each other. Only the world based reference frame is a common reference system for the two persons.

The mathematical fundament of a coordinate system is a set of basis vectors, where linear combinations of their scalar multiples are used to describe the complete space of coordinates. One such a linear combination of an n-dimensional coordinate system is called a coordinate, written as a simple ordered set of n numbers: $\mathbf{v} = [u_1, u_2, \ldots, u_n]$. On the other hand, an observational frame of reference is a physical concept relating to an observer and the observer's motion. This state of motion is the only characterization of such an observational frame of reference. Various distinct coordinate systems can of course be linked to a single observational frame. Two types of observational frames of reference exist: *inertial* and *non-inertial*. In inertial frames of reference, every physical law takes the same form in each frame, i.e. the laws of physics apply without (major) modifications.

There are frames of reference that are used frequently in aerospace research and industry. Two varieties of those are used in this thesis, one of them a modified version of a widely used frame definition. The first is a non-inertial and the second an inertial frame of reference, here called the *body frame* and the *world frame*, respectively.

Body Frame

The body frame is a non-inertial frame of reference fixed in the moving robot. It defines a coordinate system with its origin set in the center of mass of the drone and its basis axes aligned with the body axes (Figure 3.5):

- The $\boldsymbol{x}$-axis points through the nose of the drone.
- The $\boldsymbol{y}$-axis points to the right of the $\boldsymbol{x}$-axis (facing in pilot's view direction), perpendicular to the $\boldsymbol{x}$-axis.
- The $\boldsymbol{z}$-axis points through the bottom of the craft, perpendicular to the $\boldsymbol{x}$-$\boldsymbol{y}$-plane, satisfying the Right-Handed rule.

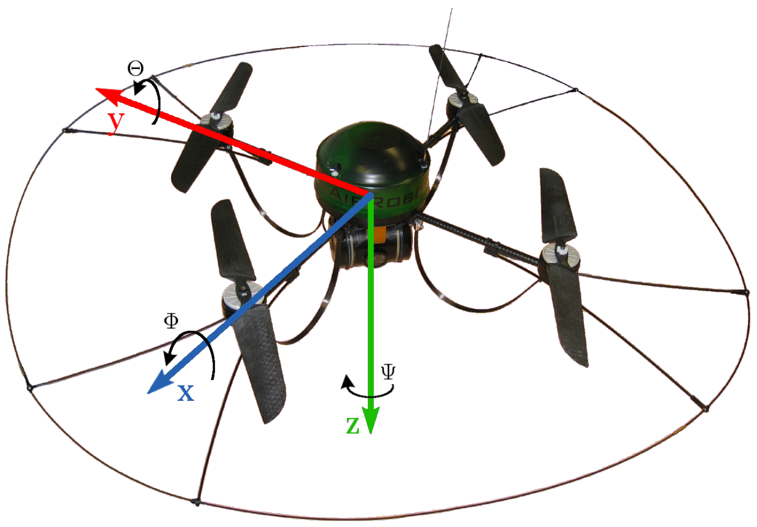


Figure 3.5: Body Frame of Reference.

Consequently, the coordinate system is fixed both in origin and orientation to the body of the drone. Since the drone is moving, so is the body frame. Rotations can be defined by Euler angles Φ, Θ, Ψ around the frame's axes(Figure 3.5). They are:

- Φ: Roll around $\boldsymbol{x}$-axis.
- Θ: Pitch around $\boldsymbol{y}$-axis.
- Ψ: Yaw around $\boldsymbol{z}$-axis.

If the coordinate frame of a vector in equations is not clear from the context, then the sub-index defining the respective frame of reference is essential. Vectors given in coordinates with respect to the body frame are sub-indexed by a capital B, e.g. $\mathbf{v}_B$.

World Frame

The world frame, an inertial frame of reference, defines a coordinate system *initially* based on the non-inertial NED coordinate system (*North-East-Down*). The *N*orth-*E*ast-*D*own system has its origin fixed at the center of gravity of the moving aircraft, while its axes are fixed along the geodetic directions defined by the Earth's surface:

- The $\boldsymbol{N}$-axis points north parallel to the geoid surface, in the polar direction.
- The $\boldsymbol{E}$-axis points east parallel to the geoid surface, along a latitude curve.
- The $\boldsymbol{D}$-axis points downward, toward the Earth's surface, anti parallel to the surface's outward normal.

Since the operational area spread and time duration of missions for such a small-sized lightweight drone are relatively small, the rotation of the Earth can be ignored in the formulations of the equations of motion.

Initially means that during an initialization phase, when the robot is not allowed to move but its sensors are taking measurements, the NED directions are calculated from the measurements and the current location corresponding to the center of gravity of the robot is marked as the origin, see section 3.4.4. After initialization the origin of that world frame stays static where the initialization took place while the robot can move about. This modifies the non-inertial NED frame into the inertial world frame. Vectors described in coordinates with respect to the coordinate system attached to the world frame are sub-indexed with a capital W, e.g. $\mathbf{a}_W$, when not clear from the context.

Coordinate Transformations

The respective coordinate transformations for spatial vectors from world to body frame and reverse can be calculated from the position vector, $\boldsymbol{p}$. defined in section 3.4.1. To summarize that section in one sentence, the position vector not only contains the current 3-D location of the robot, but also its current orientation in quaternion representation, both with respect to the world frame coordinate system.

The information contained in $\boldsymbol{p}$ suffices to calculate the transformation matrix from world to body frame, ${}^{B}\mathbf{X}_W$, its inverse, the transformation matrix from body to world frame, ${}^{W}\mathbf{X}_B$, as well as their spatial force vector counterparts. As seen in Section 3.2.1, the 3×3 rotation matrix, ${}^{B}\boldsymbol{R}_W$, and the translation vector, ${}^{B}\boldsymbol{p}_W$, are necessary for Equation (3.8). The 3-D translation vector is given as

$$ {}^{B}\boldsymbol{p}_W = \begin{pmatrix} p_N \\ p_E \\ p_D \end{pmatrix}, \tag{3.18}$$

where its elements are taken directly from the position vector, $\boldsymbol{p}$, of the current system state, see Equation (3.42).

The rotation matrix, ${}^{B}\boldsymbol{R}_W$, using the conversion from quaternion representation of rotations to 3×3 rotation matrix form, is defined as follows [31],

$$
{}^{B}\boldsymbol{R}_W = 2 \begin{pmatrix} q_0^2 + q_1^2 - 0.5 & q_1 q_2 + q_0 q_3 & q_1 q_3 - q_0 q_2 \\ q_1 q_2 - q_0 q_3 & q_0^2 + q_2^2 - 0.5 & q_2 q_3 + q_0 q_1 \\ q_1 q_3 + q_0 q_2 & q_2 q_3 - q_0 q_1 & q_0^2 + q_3^2 - 0.5 \end{pmatrix}, \tag{3.19}
$$

where quaternion $\boldsymbol{q} = (q_0, q_1, q_2, q_3)^T$ is given as the orientation of the drone, taken directly from $\boldsymbol{p}$ as well.

Given the translation vector and the rotation matrix, the transformation matrix can be calculated using Equations (3.8) and (3.10):

$$
{}^{B}\mathbf{X}_W = \begin{pmatrix} {}^{B}\boldsymbol{R}_W & \mathbf{0} \\ {}^{B}\boldsymbol{R}_W \boldsymbol{S}\left({}^{B}\boldsymbol{p}_W\right)^T & {}^{B}\boldsymbol{R}_W \end{pmatrix}. \tag{3.20}
$$

The inverse, the transformation matrix from body to world frame, ${}^{W}\mathbf{X}_B$, is calculated as follows, using Equation (3.9):

$$
\begin{aligned}
{}^{W}\mathbf{X}_B &= \begin{pmatrix} {}^{W}\boldsymbol{R}_B & \mathbf{0} \\ -\boldsymbol{S}\left({}^{W}\boldsymbol{p}_B\right) {}^{W}\boldsymbol{R}_B & {}^{W}\boldsymbol{R}_B \end{pmatrix} \\
&= \begin{pmatrix} {}^{B}\boldsymbol{R}_W^T & \mathbf{0} \\ \boldsymbol{S}\left({}^{B}\boldsymbol{p}_W\right) {}^{B}\boldsymbol{R}_W^T & {}^{B}\boldsymbol{R}_W^T \end{pmatrix}.
\end{aligned} \tag{3.21}
$$

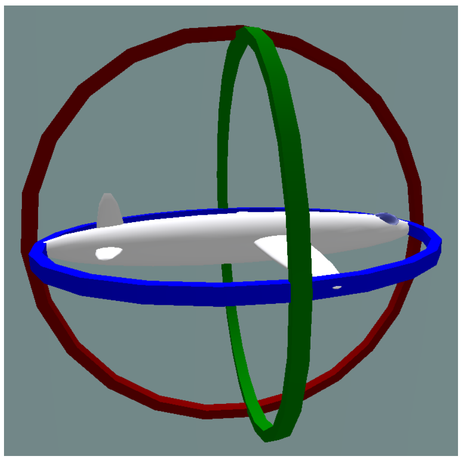
The transformation matrices for spatial force vectors can be calculated analogously using Equation (3.11). With these transformation matrices, all necessary coordinate transformations in the process and observation model equations can be performed. The matrices are dynamic, i.e. they change over time, corresponding to the change of $\boldsymbol{p}$ over time and are reevaluated at every update step.

3.2.3 Quaternions

The orientation of a vehicle can be described with the Euler angles Φ, Θ and Ψ, but Euler angles have a disadvantage called the *Gimbal Lock*. A Gimbal is a ring that can rotate around an axis. Three gimbals are mounted concentrically to compensate for rotations in three-dimensional space. Euler angles are used to represent rotations in 3-D space by three consecutive rotational movements about three axes perpendicular to each other. This similarity to linear coordinates makes Euler angles intuitive, but its intuitive simplicity comes with the cost of a big disadvantage: gimbal lock.

Gimbal lock occurs when two of the three gimbals are driven into the same place, consequently unable to compensate for rotations around one axis in three-dimensional space. The problem that occurs is not really a technical arrest of the gimbal's mobility, these are still able to rotate around their respective axes. But a loss of one degree of freedom nevertheless happens because of the parallel orientation of two gimbal axes, as illustrated in Figure 3.6.

One solution is the introduction of a fourth dimension, a fourth gimbal driven by a motor to keep large angles between roll and yaw gimbal axes. Another solution is to install a motor to the third gimbal which flips the gimbal by 180° if the angle between roll and yaw axes drops below a small threshold. This second alternative was used in the Apollo 11 moon mission with a threshold of 85° for the pitch angle, but the computer program crashed and froze the IMU[32].


(a) No Gimbal Lock.

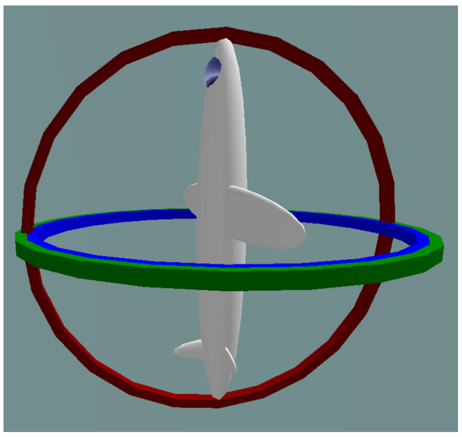

(b) Gimbal Lock.

Figure 3.6: Gimbal configurations.
If two gimbal axes are driven into parallel orientation, then these two gimbals can only compensate one degree of freedom instead of two, therefore one degree of freedom is lost.[1,2]

A mathematical solution is given by *Quaternions*, a four-dimensional non-commutative number system extending the complex numbers that can be employed to describe three-dimensional rotations gimbal-lock-free. A quaternion, $\boldsymbol{q}$, is represented by a four-tuple $\underline{\boldsymbol{q}}$ of real numbers, defined as

$$\underline{\boldsymbol{q}} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} . \tag{3.22}$$

The relationship between the four-tuple and the quaternion is defined by a linear combination as follows,

$$\boldsymbol{q} = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3, \tag{3.23}$$

where $\mathbf{i}, \mathbf{j}$ and $\mathbf{k}$ are imaginary numbers satisfying

$$\mathbf{ii} = \mathbf{jj} = \mathbf{kk} = \mathbf{ijk} = -1. \tag{3.24}$$

The non-commutativity of quaternions and these imaginary numbers should be emphasized again, i.e. $\mathbf{ij} \neq \mathbf{ji}$, but $\mathbf{ij} = -\mathbf{ji}$.

Additionally, because multiple rotation movements are necessary for Euler angles, the order of these is important and needs to be defined beforehand. A quaternion does not describe multiple consecutive rotational movements to arrive at the final rotation, but describes a single rotation. Intuitively, this can be understood as a different formulation for the *axis-angle* representation of a rotation: given an angle α and an axis directed by a normalized vector $\boldsymbol{n} = (n_x, n_y, n_z)^T$, the quaternion with the values of Equation (3.25) represents a single rotation of α radians around the axis driven by vector $\boldsymbol{n}$,

$$\underline{\boldsymbol{q}} = \begin{pmatrix} \cos\frac{\alpha}{2} \\ n_x \sin\frac{\alpha}{2} \\ n_y \sin\frac{\alpha}{2} \\ n_z \sin\frac{\alpha}{2} \end{pmatrix}. \tag{3.25}$$

Consequently, no gimbal lock can occur.

Quaternion rotations are combined by multiplication, i.e. given a first and a second successive rotation $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$, the combined rotation can be calculated as $\boldsymbol{q}_1\boldsymbol{q}_2$ or $\boldsymbol{q}_2\boldsymbol{q}_1$, depending on the frame of reference being a body frame moving with the rotating entity or an absolute world frame not moving with the rotating craft, respectively. To have *pure rotations* without scaling, the quaternion must have an absolute value of 1, i.e. it must be a unit quaternion. The norm of a quaternion is defined as the square root of the product of the quaternion $\boldsymbol{q}$ and its conjugate $\boldsymbol{q}^*$, see Equation (3.26). The conjugate of $\boldsymbol{q} = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$ is defined as $\boldsymbol{q}^* = q_0 - \mathbf{i}q_1 - \mathbf{j}q_2 - \mathbf{k}q_3$, resulting in the following equation for the norm of a quaternion,

$$\|\boldsymbol{q}\| = \sqrt{\boldsymbol{q}\boldsymbol{q}^*} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}. \tag{3.26}$$

Hence, if $\boldsymbol{q}$ is a unit quaternion, then its conjugate is its inverse, $\boldsymbol{q}^* = \boldsymbol{q}^{-1}$.

The unit prerequisite leads to another advantage of quaternion representation of rotations over matrix representation. A quaternion that lost its unit characteristic during computations because of numerical imprecisions can always be normalized, leading to a slightly inaccurate rotation, but a pure rotation nevertheless. This is unlike numerical inconsistencies in rotation matrices, where these matrices do not represent pure rotations anymore after numerical imprecision destroyed their orthogonality or changed the unity of their determinants.

Conversions

Conversion from Euler angle representation to quaternions is done using the following equations, where we choose the order of the angles as *Roll-Pitch-Yaw*. Formulated as about which body axis to rotate, the order is $\boldsymbol{x}$-$\boldsymbol{y}$-$\boldsymbol{z}$ and the angles are Φ, Θ, Ψ, respectively.

The quaternion elements can be calculated from the Euler angles as follows,

$$\begin{aligned} q_0 &= c_1c_2c_3 - s_1s_2s_3, \\ q_1 &= s_1c_2c_3 + c_1s_2s_3, \\ q_2 &= c_1s_2c_3 - s_1c_2s_3, \\ q_3 &= s_1s_2c_3 + c_1c_2s_3, \end{aligned} \tag{3.27}$$

where

$$\begin{aligned} c_1 &= \cos\frac{\Phi}{2}; & c_2 &= \cos\frac{\Theta}{2}; & c_3 &= \cos\frac{\Psi}{2}, \\ s_1 &= \sin\frac{\Phi}{2}; & s_2 &= \sin\frac{\Theta}{2}; & s_3 &= \sin\frac{\Psi}{2}. \end{aligned}$$

The proof for this conversion is done using the single rotations (as quaternions) applied in the correct order of Euler angles as defined above:

- Q_x the rotation element about the $\boldsymbol{x}$-axis
- Q_y the rotation element about the $\boldsymbol{y}$-axis
- Q_z the rotation element about the $\boldsymbol{z}$-axis

Thus, using Equation (3.25), these rotations are given as

$$\begin{aligned} Q_x &= \cos\frac{\Phi}{2} + \mathbf{i}\sin\frac{\Phi}{2} = c_1 + \mathbf{i}s_1 \\ Q_y &= \cos\frac{\Theta}{2} + \mathbf{j}\sin\frac{\Theta}{2} = c_2 + \mathbf{j}s_2 \\ Q_z &= \cos\frac{\Psi}{2} + \mathbf{k}\sin\frac{\Psi}{2} = c_3 + \mathbf{k}s_3. \end{aligned} \tag{3.28}$$

This gives the combined rotation

$$\begin{aligned} Q &= (Q_xQ_y)\,Q_z \\ &= ((c_1 + \mathbf{i}s_1)(c_2 + \mathbf{j}s_2))(c_3 + \mathbf{k}s_3) \\ &= (c_1c_2 + \mathbf{j}c_1s_2 + \mathbf{i}s_1c_2 + \mathbf{ij}s_1s_2)(c_3 + \mathbf{k}s_3) \\ &= c_1c_2c_3 + \mathbf{j}c_1s_2c_3 + \mathbf{i}s_1c_2c_3 + \mathbf{k}s_1s_2c_3 + \mathbf{k}c_1c_2s_3 + \mathbf{jk}c_1s_2s_3 + \mathbf{ik}s_1c_2s_3 + \mathbf{kk}s_1s_2s_3 \\ &= (c_1c_2c_3 - s_1s_2s_3) + \mathbf{i}(s_1c_2c_3 + c_1s_2s_3) + \mathbf{j}(c_1s_2c_3 - s_1c_2s_3) + \mathbf{k}(s_1s_2c_3 + c_1c_2s_3). \end{aligned} \tag{3.29}$$

For the conversion from quaternion representation to Euler angles, reverse trigonometric functions must be used. Since trigonometric functions are surjective, their inverse functions have many possible results. It is common to assume the functions return angles within certain intervals, i.e. $\arcsin(\cdot)$ returns the angle within $[-90°, 90°]$, and $\arctan 2(\cdot)$ returns the angle within $[-180°, 180°]$. The conversion equations for roll (Φ), pitch (Θ) and yaw (Ψ) are given as

$$\begin{aligned}
\Phi &= \arctan 2\left(2q_0q_1 - 2q_2q_3, q_0^2 - q_1^2 + q_2^2 - q_3^2\right), \\
\Theta &= \arcsin\left(2q_0q_3 + 2q_1q_2\right), \\
\Psi &= \arctan 2\left(2q_0q_2 - 2q_1q_3, q_0^2 + q_1^2 - q_2^2 - q_3^2\right).
\end{aligned}$$

These equations work correctly for all points except the singularities at pitch angles of $-90°$ and $90°$, where the roll and yaw angles would then be given as $0°$. This is the gimbal lock mentioned above. To detect gimbal lock and subsequently calculate correct values for these points, a test is introduced. In the quaternion representation, the pitch angle is calculated and compared to a threshold, e.g. 85° for the Apollo 11 project, and the angles adjusted accordingly. This comparison is given as

$$\begin{aligned}
&q_0q_3 + q_1q_2 > 0.499 * (q_0^2 + q_1^2 + q_2^2 + q_3^2), \quad \text{and for the other pole} \\
&q_0q_3 + q_1q_2 < -0.499 * (q_0^2 + q_1^2 + q_2^2 + q_3^2),
\end{aligned}$$

where 0.499 represents half of the cutoff threshold for the pitch angle quaternion component. The angles are then calculated as follows,

$$\begin{aligned}
\text{if} \quad & ((q_0q_3 + q_1q_2) > 0.499 * \left(q_0^2 + q_1^2 + q_2^2 + q_3^2\right)) \\
& \quad \Phi = 0, \\
& \quad \Theta = \frac{\pi}{2}, \\
& \quad \Psi = 2\arctan 2\left(q_1, q_0\right), \\
\text{else if} \quad & ((q_0q_3 + q_1q_2) < -0.499 * \left(q_0^2 + q_1^2 + q_2^2 + q_3^2\right)) \\
& \quad \Phi = 0, \\
& \quad \Theta = -\frac{\pi}{2}, \\
& \quad \Psi = -2\arctan 2\left(q_1, q_0\right), \\
& \text{else} \\
& \quad \Phi = \arctan 2\left(2q_0q_1 - 2q_2q_3, q_0^2 - q_1^2 + q_2^2 - q_3^2\right), \\
& \quad \Theta = \arcsin\left(2q_0q_3 + 2q_1q_2\right), \\
& \quad \Psi = \arctan 2\left(2q_0q_2 - 2q_1q_3, q_0^2 + q_1^2 - q_2^2 - q_3^2\right).
\end{aligned} \tag{3.30}$$

For representation of angular velocity (given in Euler angles) in quaternion representation, i.e. the derivative of a quaternion over time, the following equation can be derived[30]:

$$\begin{aligned}
\frac{d}{dt}\boldsymbol{q}(t) &= \frac{1}{2}\mathsf{Q}\omega^*(t) \\
&= \frac{1}{2}\begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix}\omega^*(t),
\end{aligned} \tag{3.31}$$

where $\omega^*(t)$ is the 3-D angular velocity at time instant t, $\omega(t)$, augmented to 4 dimensions, $\omega^*(t) = (a\|\omega\|(1 - \|\boldsymbol{q}\|), \omega_x, \omega_y, \omega_z)^T$, and Q is calculated from the orientation quaternion $\boldsymbol{q}$ at the start of integration. The resulting orientation is then calculated via numerical integration over the given time frame. $0 < a < 1$ is a scalar constant whose exact value is not critical, e.g. $a = 0.1$[30].

Thereby, the unit quaternion of $\boldsymbol{q} = (1, 0, 0, 0)^T$ is calculated to represent the orientation of the drone in world frame coordinates when each of the body frame axes $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ is parallel to the corresponding world frame axis N, E and D, respectively, i.e. $(\boldsymbol{x} \parallel N) \wedge (\boldsymbol{y} \parallel E) \wedge (\boldsymbol{z} \parallel D)$. In other words, $\boldsymbol{q} = (1, 0, 0, 0)^T$ represents the orientation when all Euler angles are equal to zero, $\Phi = \Theta = \Psi = 0°$.

3.2.4 Quadrocopter Control

In order to understand the basic principles regarding flight control of a quadrocopter, the steering of such a four-propelled aircraft is explained. As seen in various figures above, the rotors are positioned equidistant from each other around a central point along a rigid cross frame. This cross shape also gives them the name *X4-flyers*. The rotors can be controlled individually, enabling vertical take-off and landing as well as omnidirectional flight.

The omnidirectional control is achieved by differential command of the thrust generated by each of the four rotors (Figure 3.7). The up-down motion is controlled by collectively

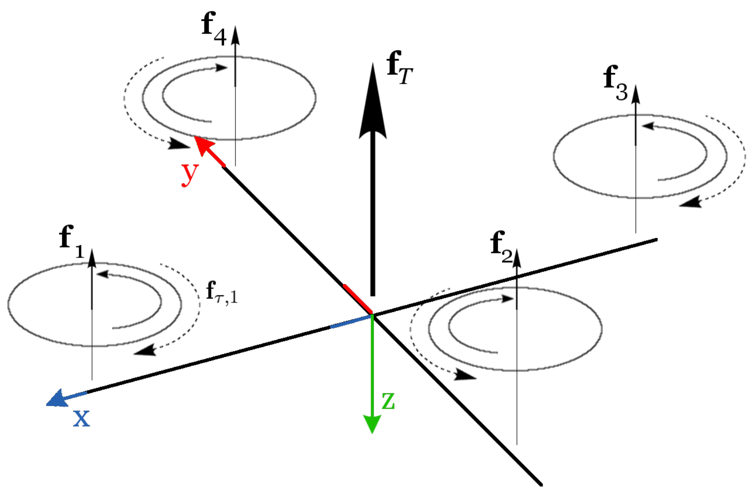


Figure 3.7: Quadrocopter Force Model.
The four rotors generate a collective thrust force contrary to the $\boldsymbol{z}$-axis. Opposite rotors rotate in the same direction, generating reactive torque forces (dotted arrows) employed in yaw rotation control.

increasing or decreasing the revolutions per minute (*RPMs*) of all four rotors. Sideways motion is accomplished by leaning into the desired direction and increasing collective thrust to counteract the craft's tendency to side-slip toward the ground. This *leaning* motion consists of a roll and pitch component around the $\boldsymbol{x}$- and $\boldsymbol{y}$-axis, respectively. The roll or pitch component results from decreasing the RPMs of the rotor in the desired direction and increasing the RPMs of the opposite rotor proportionately, that the collective thrust is preserved.

The yaw rotational element is based on a technique with a little more finesse: A turning rotor has to overcome air resistance depending on the rotor's speed and surface properties, its blade's area and pitch and other factors (Section 3.4.2). The reactive force from that rotor's air drag acts on the rotor in the opposite direction of the rotor's own rotation (dotted arrows in Figure 3.7). Rotors on the same axis rotate in the same direction, but contrary to the rotors on the other axis. Thereby, as long as all rotors turn at the same speed, the generated reactive torque forces of all rotors cancel each other out and result in a net reactive torque force of zero: The quadrocopter does not rotate around the $\boldsymbol{z}$-axis. If the speed of one set of rotors is increased, the induced torques fall out of their equilibrium and the aircraft rotates in the direction of the total torque. This yaw rotation has no translational effect in direction of the $\boldsymbol{x}$- or $\boldsymbol{y}$-axes, and the $\boldsymbol{z}$-axis component from increase of rotor speeds can be compensated by reducing the speed of the other diagonal pair.

3.3 UAV Hardware

The quadrocopter used for this thesis is the AirRobot AR-100[3]. It is a small, light-weight UAV. The AR-100 is an industrially engineered and produced vehicle that is already seeing application in military, public and private sectors. It is a remote-controlled quadrocopter, equipped with various sensors, that has been the subject of a European research project called μDRONES[4]. The drone is not completely autonomous, it needs a base station to send its mission or flight commands. This section presents the sensors and other physical characteristics of the AR-100 which are necessary for the process and observation models.

Unfortunately, much of the hard- and software used onboard the AR-100 is protected by non-disclosure agreements of the company. This protection against industrial espionage also makes the drone more difficult to work with, since many variables are unknown.

3.3.1 Quadrocopter

The rotors of the quadrocopter are specially made low-weight carbon fiber rotors. They are 14“ rotors (radius about 17.8cm) with a maximum blade width of 4.5cm. The motors that spin them are four brushless outrunner motors using a current of about 7A with a peak current of 35A. The UAV is powered by a 14.7V Lithium-ion battery with an

[3]More information at `www.airrobot.com`

[4]More information at `www.ist-microdrones.org`

electrical charge of up to 2400mAh. This combination leads to maximum flight times of 25 minutes, given good weather conditions and quasi-stationary flight behavior.

The rotor speeds cannot be set directly by the manual controller or by the base station software. The control of the quadrocopter is achieved by higher level commands, i.e. sending the desired roll, pitch or yaw angle to the UAV. Its onboard control software then converts these instructions to motor commands, thus offering only indirect control of the rotor speeds. The available manual control inputs are the default remote control, a gamepad connected to the base station or the keyboard of the base station. The two control pads use sticks to steer the UAV, using the positions of the sticks as actual control angle commands. The keyboard control, on the other side, can set the angles incrementally and remain there, until the angle is changed manually. This offers a slower but more accurate alternative to the control by gamepad or remote control. The keyboard control is used in some of the experiments, where a constant flight behavior was desired (Chapter 4).

3.3.2 Base Station

For operation of the AR-100, a base station is necessary, consisting of a computer with the control software[33], an uplink antenna for control commands, two down-link antennae, one combined with a frame grabber for video data and one for sensor-related and miscellaneous data, and optionally a separate remote control. The base station computer handles all of the computations concerning mission planning, localization, optic flow and egomotion calculations, Kalman filter calculations, et cetera.

Most of the time, the computer used in this scenario is an *Intel Pentium 4* desktop pc with 2.6GHz CPU clock frequency and 1GB of RAM. For the final experiments, an *Intel Core 2 Duo E6600* CPU with a 2.4GHz clock and 4GB RAM is employed. The operating systems in both cases are 32bit Linux operating systems, but the P4 runs the *OpenSUSE 11*[5] distribution with kernel 2.6.25.5 while the Core2Duo uses an up-to-date *Gentoo*[6] distribution with kernel 2.6.30-r7.

The control system software was written in the programming language C++ and compiled using the *Gnu Compiler* version 4.3.4.

The frame grabber is portable sized from the company *DIGITUS*, but can only grab 25 frames per second at a resolution of 720×576 pixels, i.e. at the PAL standard. The camera has a native resolution of 470 lines, though, so the frame grabber extrapolates the recorded pictures to 576 lines.

[5]`www.opensuse.org`

[6]`www.gentoo.org`

Manual control can be accomplished either through using a remote control which sends the commands directly to the AR-100, or through using the control software via keyboard of gamepad control input. The manual remote control is a must for flying vehicles in Germany because of laws governing the air space.

3.3.3 Sensors

The sensors that the AR-100 is equipped with are:

- Global Positioning sensor
- Inertial (and Gyroscopic) Measurement Unit
- Motor tachometer
- Video Camera
- Magnetometer
- Barometer

Since the GPS sensor was not yet operational under the Linux operating system, and the goal of this paper is to formulate an estimation system without GPS information, the GPS sensor is not used. The sensors do not time stamp their measurements, and the software running on the robot cannot be modified, so the best available time stamp is using the time instant of reception of data at the base station as the measurement time stamp. This way, the state estimate always lags the signal latency behind the actual state of the robot. The fastest measurements arrive at a time interval of about 24ms.

Inertial Measurement Unit

An IMU sensor is an inertial measurement unit consisting of three gyroscopes and three inertial sensors. These sensors are aligned in such a way, that it measures the three-dimensional angular velocities and linear accelerations of itself and of the object it is attached to. There exist different types of IMUs, namely free-moving and strap-down. Free-moving IMUs have actual gimbals inside and gimbal lock can occur, whereas strap-down IMUs are very small electronic devices without the danger of a gimbal lock. The IMU used in the AR-100 is a three-axis strap-down IMU, i.e. it measures the angular velocities and the linear accelerations in three dimensions each.

Unfortunately, the exact specifications of the implemented IMU are secret. This means that the standard deviation error and the maximal error of the sensor during a time interval could not be determined.

Motor Tachometer

The motor speeds can be acquired by an electronic device measuring the voltage and the current at the motor. The tachometer reports via I^{2}C protocol, *Inter-Integrated Circuit*, and the drone sends this information to the base station. The data sent is a dimensionless value in the interval $[64, 256]$ for each motor. Unfortunately, the mapping from that value to the actual rotational speed of the rotor is kept secret. For that reason, the essential proportionality constant is modeled and empirically tested in Section 4.2.1.

The resulting conversion formula is given as

$$w_i\,[RPM] = 0.01878x_i^2 + 13.58696x_i + 19.41708,$$

where x_i is the I^{2}C value of motor i, and w_i the resulting rotor speed in [RPM].

Video Camera

The optical sensor of the AR-100 is a digital camera able to record 25 color images per second at a resolution of 470 TV-lines. It uses an automatic white fader that cannot be switched off and its horizontal angle of aperture is $\pm 45°$. The camera is located within a horizontal barrel underneath the quadrocopter, which can be rotated around its horizontal axis within a frame of 130°. This rotation is controlled by certain commandos to the drone.

The video information is broadcasted via an analog signal to the receiver of the base station. This introduces errors to the video information, at times rendering some images useless for the optic flow and egomotion analyses.

Magnetometer

The AR-100 is also equipped with an onboard magnetic compass. The sensor readings range from 0° to 359° and represent the alignment of the body frame $\boldsymbol{x}$-axis toward magnetic north. To that effect, the following relationship is between the given measurements and the magnetic orientation of the $\boldsymbol{x}$-axis:

- 0° correlates to the $\boldsymbol{x}$-axis pointing towards magnetic North.
- 90° correlates to the $\boldsymbol{x}$-axis pointing towards magnetic East.
- 180° correlates to the $\boldsymbol{x}$-axis pointing towards magnetic South.
- 270° correlates to the $\boldsymbol{x}$-axis pointing towards magnetic West.

With the help of the magnetometer, the world frame is aligned during the initialization phase (Section 3.4.4). After the initialization, sensor measurements are incorporated into the state estimate correction (Section 3.4.3). In order to average the measurements in a sensible way, the degree unit representation of orientation is not very suitable. Take for example two measurements, one of 1° and the other of 359°. Both point in almost the same direction, with only a 2° difference. However, the arithmetic mean of the two values is 180°, the exactly opposite direction. Thus, the orientation given in units of degrees is converted to a two dimensional vector. This vector corresponds to the x and y coordinates of a unit vector pointing in the direction of the magnetometer observation. In other words, the magnetometer is regarded two-dimensionally as a unit circle, with its observation report in degrees taken as the polar coordinate of a unit vector corresponding to the $\boldsymbol{x}$-axis of the quadrocopter. This coordinate is then converted via trigonometric functions into cartesian coordinates. The cartesian representation suits the calculation of the arithmetic mean. The measurement for the angle β is then given as

$$\boldsymbol{z}_{Magnet} = \begin{pmatrix} \cos(\beta) \\ \sin(\beta) \end{pmatrix} . \tag{3.32}$$

Barometer

A common technique for measuring the height of an aircraft employs a barometer to measure air pressure. Air pressure decreases with higher altitude, but it also depends on air temperature and mean sea level. The specific model of barometer built into the AR-100 is not known, but common digital barometers are calibrated and temperature-compensating. The barometer measurements are already converted into meters [m] by the software running on the drone and then sent to the base station. An often observed problem that the barometer on the UAV shows, is that the drone tends to create fluctuating pressure fields, disturbing the altimeter related measurements of the barometer. These turbulences are created by the running rotors, pushing the air away. This unpredictability of the barometer is reflected in its noise covariance matrix (Section 3.4.3).

3.3.4 Inertia

For proper modeling of a mobile object, knowledge of its mass and inertia tensor is fundamental. As discussed below, for modeling purposes the body of the AR-100 is assumed to be a perfect rigid body with a diagonal inertia tensor. The shape of the inertial body is modeled to be the sum of a solid spherical body in the center and a concentrically placed circular loop in the $\boldsymbol{x}$-$\boldsymbol{y}$-plane. The sphere represents the central body containing the electronics, sensors, camera, center frame and miscellaneous items located close to the center of gravity. The circular loop models the four motors together with rotors, bumper wire and the second half of the weight of the frame axes. Since the drone frame is sufficiently point-symmetrical with respect to its center of gravity, this should be a good approximation of the actual inertia matrix, which can only be determined empirically. With these assumptions, the three-dimensional inertia tensor for the center sphere, $\boldsymbol{I}_{CS}$, is given as

$$\boldsymbol{I}_{CS} = \begin{pmatrix} \frac{2}{5} m_{CS} r_{CS}^2 & 0 & 0 \\ 0 & \frac{2}{5} m_{CS} r_{CS}^2 & 0 \\ 0 & 0 & \frac{2}{5} m_{CS} r_{CS}^2 \end{pmatrix} , \tag{3.33}$$

where m_{CS} is the mass in [kg] and r_{CS} the radius in [m] of the solid center sphere of the AR-100. Analogously, the inertia tensor for the outer circular loop, $\boldsymbol{I}_{OL}$, is defined as follows,

$$\boldsymbol{I}_{OL} = \begin{pmatrix} \frac{1}{2}m_{OL}r_{OL}^2 & 0 & 0 \\ 0 & \frac{1}{2}m_{OL}r_{OL}^2 & 0 \\ 0 & 0 & m_{OL}r_{OL}^2 \end{pmatrix}, \tag{3.34}$$

where m_{OL} is the mass in [kg] and r_{OL} the radius in [m] of the circular loop of the AR-100. The radius of the loop is equal to the distance of the rotors to the center of mass, hence it is

$$r_{OL} = d.$$

The distance d of the rotors to the robot's center of mass is given as

$$d = r_{OL} = 0.272m. \tag{3.35}$$

The total approximate three-dimensional inertia tensor for the quadrocopter is then given as

$$\begin{aligned} \boldsymbol{I}_{cm} &= \boldsymbol{I}_{CS} + \boldsymbol{I}_{OL} \\ &= \begin{pmatrix} \frac{2}{5}m_{CS}r_{CS}^2 & 0 & 0 \\ 0 & \frac{2}{5}m_{CS}r_{CS}^2 & 0 \\ 0 & 0 & \frac{2}{5}m_{CS}r_{CS}^2 \end{pmatrix} + \begin{pmatrix} \frac{1}{2}m_{OL}d^2 & 0 & 0 \\ 0 & \frac{1}{2}m_{OL}d^2 & 0 \\ 0 & 0 & m_{OL}d^2 \end{pmatrix} & (3.36) \\ &= \begin{pmatrix} \frac{2}{5}m_{CS}r_{CS}^2 + \frac{1}{2}m_{OL}d^2 & 0 & 0 \\ 0 & \frac{2}{5}m_{CS}r_{CS}^2 + \frac{1}{2}m_{OL}d^2 & 0 \\ 0 & 0 & \frac{2}{5}m_{CS}r_{CS}^2 + m_{OL}d^2 \end{pmatrix}. & (3.37) \end{aligned}$$

The radius of the central sphere is measured as

$$r_{CS} = 0.15m. \tag{3.38}$$

The total mass of the drone including charged batteries, barrel camera, rotors and bumper wire is measured as follows,

$$m = 0.9103kg. \tag{3.39}$$

Table 3.1 lists the masses of more or less arbitrarily separated components of the AR-100. The center sphere's mass consists of the body center, the barrel camera, the battery and half of the axes, which results in a mass of

$$m_{CS} = 0.52848kg,$$

and the mass of the outer loop is the sum of half of the axes, all motors, all four rotors and the bumper wire construction, which adds up to

$$m_{OL} = 0.38182kg.$$

Employing Equation (3.37), the three-dimensional inertia tensor for the AR-100 is given as

$$\boldsymbol{I}_{cm} = \begin{pmatrix} 0.018881 & 0 & 0 \\ 0 & 0.018881 & 0 \\ 0 & 0 & 0.033005 \end{pmatrix},$$

Component	Quantity	Unit Mass [kg]
Body center	1	0.22046
Battery	1	0.1902
Camera	1	0.07592
Axis	4	0.02095
Motor	4	0.060
Rotor	4	0.00918
Bumper wire	1	0.0632
Total		**0.9103**

Table 3.1: Masses of components.

and the spatial inertia matrix, using Equation (3.17), around the robot's center of gravity with respect to the body frame is given as

$$\mathbf{I} = \begin{pmatrix} 0.018881 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.018881 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.033005 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9103 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9103 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9103 \end{pmatrix}. \tag{3.40}$$

This concludes the section on the UAV hardware used during this thesis. In the next section the nonlinear dynamic physical models are presented.

3.4 Derivation

This section defines the filter's system state, explaining its components. Subsequently, the equations of the nonlinear process and observation models for the Kalman filter are derived.

3.4.1 System State

The *system state*, $\mathbf{x}$, is a multi-dimensional vector of values that can be in various units independent of each other. It is the state that the Kalman filter is supposed to estimate. For this thesis, the system state vector's dimension is 26, consisting of relevant information of the robot's movement chosen for sensible localization. The system state is updated via numerical integration in the *time update step* of the Kalman filter and corrected during the *measurement update step*, see Chapter 2.2 on Kalman Filters and Sections 3.4.2 and 3.4.3 on the process and observation model formulations.

In short, the system state contains the position, velocity, IMU bias, motor speeds and the estimated wind velocity concatenated to one 26-dimensional vector. This vector is given as

$$\mathbf{x} = \begin{pmatrix} \boldsymbol{p} \\ \mathbf{v} \\ \boldsymbol{b} \\ \boldsymbol{m} \\ \boldsymbol{w} \end{pmatrix} . \tag{3.41}$$

Each element is introduced below. The detailed explanations covering their dynamics, the process and observation models, are presented in the subsequent Sections 3.4.2 and 3.4.3.

Position

The position component, $\boldsymbol{p}$, of the system state is a seven dimensional (7-D) vector consisting of the drone's orientation in quaternion representation and its three-dimensional Cartesian location $\underline{\boldsymbol{p}}$. Both are in world frame coordinates, given as

$$\begin{aligned} \boldsymbol{p}_W &= \begin{pmatrix} \boldsymbol{q} \\ \underline{\boldsymbol{p}} \end{pmatrix} \\ &= (q_0, q_1, q_2, q_3, p_N, p_E, p_D)^T . \end{aligned} \tag{3.42}$$

Position $\boldsymbol{p}$ represents the information of how far the drone has moved away from its initial position and what its current orientation with respect to the world frame is. The position changes over time, and as covered in Section 3.2.2, $\boldsymbol{p}$ is used to calculate the coordinate transformation matrices.

Velocity

The next part of the system state is the robot's velocity, $\mathbf{v}$, in body frame coordinates. $\mathbf{v}$ is a spatial motion vector, i.e. a six dimensional (6-D) vector containing the robot's angular velocity in $\left[\frac{rad}{s}\right]$ and its linear velocity in $\left[\frac{m}{s}\right]$:

$$\begin{aligned} \mathbf{v}_B &= \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_l \end{pmatrix} \\ &= (\omega_x, \omega_y, \omega_z, v_x, v_y, v_z)^T . \end{aligned} \tag{3.43}$$

The velocity $\mathbf{v}$ is the current rate of change of the position of the craft. It is important to note that is is given in angles instead of quaternions, and with respect to the body frame instead of the world frame. The change in position (and orientation), $\Delta\boldsymbol{p}$, as computed via numerical integration must be transformed to world frame coordinates before adding it to the initial position.

Inertial Measurement Unit Bias

An IMU, or *Inertial Measurement Unit*, measures and reports on two different types of a craft's egomotion components. It measures the angular velocity and the linear accelerations of the vehicle, each in three dimensions. This information, when collected and integrated correctly, allows the tracking of the vehicle's position and orientation, a method called dead reckoning, as discussed in the introductory section 1.1.3. In general, the IMU sensor measurements are corrupted by various types of errors such as scale factors, misalignments, biases, and random noise. A major disadvantage of inertial sensing for dead reckoning is the fact that these errors of the IMU sensor measurement are accumulated. The integration procedure adds every detected change to the estimated position, and any measurement errors, no matter how small, are accrued from step to step. This leads to drift, an ever-increasing estimate error between the system's estimate of the craft's current position and the actual location. In order for the model to be able to incorporate these details, the IMU bias of linear acceleration and angular velocity measurements is modeled as a random vector.

The IMU bias is a six dimensional vector of scalar values and is used to estimate how strong the IMU is biased. It is given as

$$\boldsymbol{b} = \begin{pmatrix} b_{\omega_x} \\ b_{\omega_y} \\ b_{\omega_z} \\ b_{a_x} \\ b_{a_y} \\ b_{a_z} \end{pmatrix}. \tag{3.44}$$

In other words, $\boldsymbol{b}$ describes the prevailing offset of the IMU's measurements for each dimension in the following model,

$$\mathbf{z}_{IMU}\left(t\right) = \hat{\mathbf{z}}\left(t\right) + \boldsymbol{b} + \mathbf{n}_b.$$

More on the IMU's observation model is found in Section 3.4.3.

Motor Speeds

The vector $\boldsymbol{m}$ is a four dimensional vector containing the motor speeds of the quadrocopter's four motors. The motor speeds are taken as approximate rotor speeds, given in units of revolutions per minute, $[RPM]$. The vector $\boldsymbol{m}$ is given as

$$\boldsymbol{m} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}. \tag{3.45}$$

The arrangement of the numbering of the motors starts at the motor of the drone that lies on the $\boldsymbol{x}$-axis of the body frame. From there it continues in mathematically positive direction around the quadrocopter, as viewed from above in direction of the $\boldsymbol{z}$-axis (Figure 3.8).

Wind Velocity

The last element of the system state vector is a three-dimensional vector representing the velocity that is estimated for the prevailing wind. $\boldsymbol{w}$ only considers linear velocities

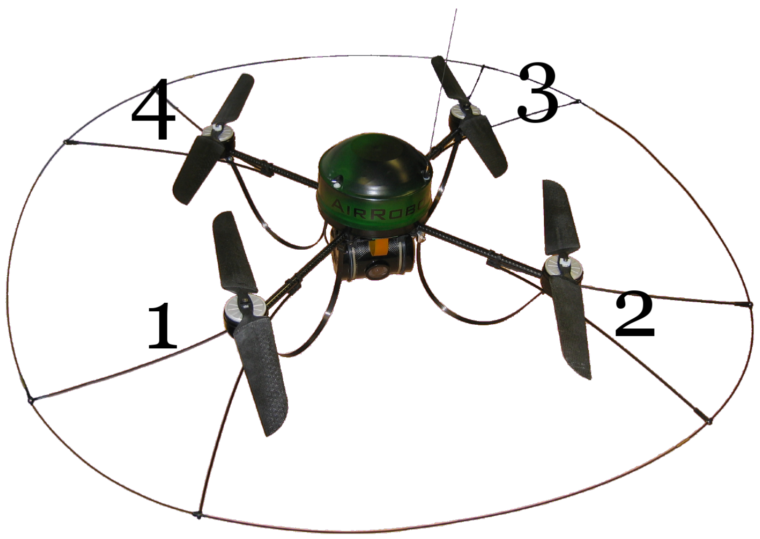


Figure 3.8: Motor Numbering.
Motor Number 1 is positioned in $\boldsymbol{x}$-axis direction from the robot center of mass and motor 4 lies along the $\boldsymbol{y}$-axis.

of the wind in world frame coordinates, given as

$$\boldsymbol{w}_W = \begin{pmatrix} w_N \\ w_E \\ w_D \end{pmatrix} . \tag{3.46}$$

It is assumed that the wind acts uniformly on the robot, which is modeled as an object with a spherical body (Section 3.3.4). Therefore possible generated torques as a consequence of linear wind forces are ignored.

This component is included for reasons of future extension, when wind velocity can be estimated reasonably well. For this thesis, the wind velocity is assumed to be zero, with minimal rate of change. Wind velocities are used for the prediction of the air drag of the quadrocopter.

3.4.2 Process Model

After having established the structure of the system state vector, we can look at how each of its components evolves over time. This section derives the differential process model equations for the Kalman filter time update.

The equations of motion are formulated with respect to the body frame, regarding the coordinate system as a non-moving frame of reference. This simplifies the equations, since the inertia matrix is constant instead of dynamic over time. Further assumptions concerning the robot to be modeled are:

1. The robot is assumed to be a perfect rigid body with diagonal three-dimensional inertial tensor.
2. The motors are considered to be first order systems.

These assumptions enable a simplification of the mathematical equations while still reflecting the actual craft's characteristics to a sufficient degree[34, 35].

As a first approximation of the navigational equation, we can ignore differences of reference frames insofar as assume all values given in world frame reference. The navigation equation is then given as

$$\ddot{\mathbf{p}} = \mathbf{a}.$$

Through integration, the values for velocity and position are obtained:

$$\begin{aligned}\dot{\mathbf{p}} = \mathbf{v} &= \int \mathbf{a} dt,\\ \mathbf{p} &= \int \mathbf{v} dt.\end{aligned}$$

The acceleration is the basis required for this navigation equation. In a force based model, the acceleration can be calculated from these forces.

Forces

A change of the position of the mobile robot in world frame coordinates is a consequence of forces acting on the robot. All forces add together to a final resultant force $\mathbf{f}_{res}$, a spatial force vector, at time instant t with respect to the body frame, given as

$$\mathbf{f}_{res} = \mathbf{f}_g + \mathbf{f}_w + \mathbf{f}_m, \tag{3.47}$$

where $\mathbf{f}_g$ is the gravitational force, $\mathbf{f}_w$ the effective force of air drag and wind on the robot, and $\mathbf{f}_m$ the force the robot's four motors and rotors generate collectively. These single forces can be calculated from an accumulation of predefined constants of the robot's environment and data contained in the system state $\mathbf{x}$. Each component is treated separately below.

The gravitational force in body frame coordinates is computed for time instant t by transforming the constant world frame gravity force vector, $\mathbf{f}_{g,W}$, into body frame coordinates using Equation (3.11) as follows,

$$\begin{aligned}\mathbf{f}_{g,B} &= {}^{B}\mathbf{X}_{W}^{F}\mathbf{f}_{g,W}\\ &= \begin{pmatrix} {}^{B}\boldsymbol{R}_W & -\boldsymbol{S}\left({}^{B}\boldsymbol{p}_W\right){}^{B}\boldsymbol{R}_W \\ \mathbf{0} & {}^{B}\boldsymbol{R}_W \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ f_{g,W} \end{pmatrix}.\end{aligned} \tag{3.48}$$

The force that the robot produces with its rotors, $\mathbf{f}_m$, is itself a sum of multiple forces, as presented in Section 3.2.4. $\mathbf{f}_m$ describes the linear forces as well as the torques created by the rotors' revolutions as follows,

$$\mathbf{f}_m = \begin{pmatrix} \mathbf{f}_\tau \\ \mathbf{f}_T \end{pmatrix}. \tag{3.49}$$

Let's first derive the thrust component of $\mathbf{f}_m$ before addressing the torques.

The linear thrust created by one rotor by pushing against air in direction of the $\boldsymbol{z}$-axis, $\mathbf{f}_{T,i}$, can be modeled as

$$\mathbf{f}_{T,i} = -bw_i^2\boldsymbol{z}, \tag{3.50}$$

where w_i is the speed of rotor i in $[RPM]$, $b > 0$ is a positive proportionality constant combining multiple constants into one, given in units of $[kg \cdot m]$, and $\boldsymbol{z}$ is the basis vector of the $\boldsymbol{z}$-axis. b depends on the robot's air drag coefficient, the orthogonal drag area, air density and rotor specific characteristics, for example the pitch angle of the rotor blades and additional geometric properties[7]. For quasi-stationary maneuvers in free, relatively still air it is a reasonable assumption that the scalar b is indeed a constant[35].

The total self-induced thrust acting on the robot is then given as the sum of the single thrust forces generated by all four rotors,

$$\begin{aligned}\mathbf{f}_T &= \sum_{i=1}^{4} \mathbf{f}_{T,i} \\ &= -b\sum_{i=1}^{4} w_i^2\boldsymbol{z}.\end{aligned} \tag{3.51}$$

As described in Section 3.2.4, rotations around the three body axes are used to control the flight motion into other directions than only up. Rotations around the $\boldsymbol{x}$- and $\boldsymbol{y}$-axes control the direction of the thrust force, $\mathbf{f}_T$.

The torques around the $\boldsymbol{x}$- and $\boldsymbol{y}$-axes of the body frame to induce the roll- and pitch-rotations, $\boldsymbol{f}_\tau|_x$ and $\boldsymbol{f}_\tau|_y$, result as a sum of the thrust forces of the rotors on the respectively opposite axis. In other words, the rotors on the $\boldsymbol{y}$-axis control the rotation about the $\boldsymbol{x}$-axis and the other way round. The coefficients of the thrust forces of the single rotors are defined with respect to right-handedness of the reference frames. The direction of a single thrust force is given with respect to the right-handed rotation direction around a specific axis (Figure 3.7). Positive rotation around the $\boldsymbol{x}$-axis lifts motor number 2 and lowers motor

[7] b also depends on the cube of the radius of the rotor blades, the number of blades, the chord length of the blades, the lift constant (linking angle of attack of the blade airfoil to the lift generated), the drag constant (associated with the airframe) and the geometry of the wake but particularly on the pitch angle of the rotor blades. For a detailed treatise of the aerodynamic model of a helicopter rotor, the reader is referred to any standard text in the field, e.g. [36].

number 4. The distance of the rotors to the body frame origin, the robot's center of gravity, is important as the rotor's thrust force acts as a leverage force concerning rotation. This yields for the roll torque about the $\boldsymbol{x}$-axis the equation given as

$$\begin{aligned} \boldsymbol{f}_\tau\big|_x &= \mathbf{f}_{T,2}\big|_x + \mathbf{f}_{T,4}\big|_x \\ &= db\left(w_2^2 - w_4^2\right), \end{aligned} \tag{3.52}$$

where $d > 0$ is the absolute displacement distance from the rotor to the center of mass of the robot and $\mathbf{f}_{T,i}\big|_x$ is the leverage force of rotor i about the $\boldsymbol{x}$-axis with respect to the right-handed definition of rotation. Analogously, the pitch torque about the $\boldsymbol{y}$-axis is given as

$$\begin{aligned} \boldsymbol{f}_\tau\big|_y &= \mathbf{f}_{T,1}\big|_y\ \mathbf{f}_{T,3}\big|_y \\ &= db\left(w_1^2 - w_3^2\right). \end{aligned} \tag{3.53}$$

The remaining reactive torque for yaw rotations, $\boldsymbol{f}_\tau\big|_z$, due to air drag generated by a rotor's rotation can be modeled as

$$\boldsymbol{f}_\tau\big|_z = kw_i^2, \tag{3.54}$$

where w_i is the speed of the rotor i in units of $[RPM]$ and $k > 0$ is a positive proportionality constant depending on the density of air and the same rotor characteristics as discussed for the linear thrust forces, but given in units of $\left[kg \cdot m^2\right]$. Since the rotors turn pairwise in opposite directions, the reactive torques of the single rotors add up to a resultant yaw torque, $\mathbf{f}_{\tau,res}\big|_z$, with coefficients depending on their directions with respect to the body frame, taken from Figure 3.7. The resultant torque is given as

$$\mathbf{f}_{\tau,res}\big|_z = k\sum_{i=1}^{4}(-1)^{i-1}w_i^2, \tag{3.55}$$

where the index i is the number of the rotor according to Figures 3.8 and 3.7. The reactive yaw torque, $\mathbf{f}_{\tau,res}\big|_z$, is given in units of $[Nm]$ $\left(= \frac{kg \cdot rad \cdot m^2}{s^2}\right)$.

Combining the derived equations for thrust forces and torques into the spatial force vector from Equation (3.49), we receive

$$\begin{aligned} \mathbf{f}_m &= \begin{pmatrix} \mathbf{f}_\tau \\ \mathbf{f}_T \end{pmatrix} \\ &= \boldsymbol{\Omega}\boldsymbol{m}^2 \\ &= \begin{pmatrix} 0 & db & 0 & -db \\ db & 0 & -db & 0 \\ -k & k & -k & k \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -b & -b & -b & -b \end{pmatrix}\boldsymbol{m}^2, \end{aligned} \tag{3.56}$$

where $\boldsymbol{m}^2$ is defined as the vector of element-wise squared rotor speeds, given as

$$\boldsymbol{m}^2 = \begin{pmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{pmatrix}.$$

$\boldsymbol{\Omega}$ is defined as a constant 6×4 matrix to calculate the force $\mathbf{f}_m$, as given in Equation (3.56).

The quadrocopter must also overcome air drag. The reactive force of air drag, $\mathbf{f}_w$, acting on the drone's body can be modeled as

$$\mathbf{f}_w = \frac{1}{2}\rho \|\mathbf{v}_{rel}\| \mathbf{v}_{rel}, \tag{3.57}$$

where $\rho > 0$ is a positive proportionality constant depending on air density, reference area for air drag, and a drag coefficient, a dimensionless constant quantifying the object's resistance in a fluid environment associated with the object's before mentioned reference area. The drag force, $\mathbf{f}_w$, is by definition the force component in the direction of the air flow velocity. $\mathbf{v}_{rel}$ represents the relative velocity of the air flow to the vehicle, given as

$$\mathbf{v}_{rel} = \mathbf{v}_w - \mathbf{v}, \tag{3.58}$$

where the first term is the wind velocity from the system state vector, Equation (3.41), given in coordinates with respect to the body frame. Defined as follows,

$$\begin{aligned} \mathbf{v}_w(t) &= \mathbf{w}_B(t) \\ &= {}^B\mathbf{X}_W(t)\,\mathbf{w}_W(t), \end{aligned} \tag{3.59}$$

where the linear component of spatial motion vector $\mathbf{w}_W(t)$ is taken directly from the current system state vector,$\mathbf{x}$, its angular component set to zero. The second term of Equation (3.59) is the current velocity of the vehicle, $\mathbf{v}(t)$ with its angular component set to zero as well. $\mathbf{v}(t)$ is already expressed with respect to the body frame.

The three component forces described above can then be combined using Equation (3.47), resulting in the equation given as

$$
\begin{aligned}
\mathbf{f}_{res} &= \mathbf{f}_g + \mathbf{f}_w + \mathbf{f}_m \\
&= {}^B\mathbf{X}_W^F \mathbf{g}_W + \frac{1}{2}\rho \|\mathbf{v}_{rel}\| \mathbf{v}_{rel} + \boldsymbol{\Omega}\boldsymbol{m}^2 \qquad (3.60)\\
&= {}^B\mathbf{X}_W^F \mathbf{g}_W + \frac{1}{2}\rho \left\| {}^B\mathbf{X}_W \mathbf{w}_W - \mathbf{v} \right\| \left({}^B\mathbf{X}_W \mathbf{w}_W - \mathbf{v} \right) + \boldsymbol{\Omega}\boldsymbol{m}^2
\end{aligned}
$$

$$
\begin{aligned}
&= {}^B\mathbf{X}_W^F \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 9.81 \end{pmatrix} \\
&\quad + \frac{1}{2}\rho \left\| {}^B\mathbf{X}_W \begin{pmatrix} \mathbf{0} \\ \boldsymbol{w}_W \end{pmatrix} - \begin{pmatrix} \mathbf{0} \\ \boldsymbol{v}_l \end{pmatrix} \right\| \left[{}^B\mathbf{X}_W \begin{pmatrix} \mathbf{0} \\ \boldsymbol{w}_W \end{pmatrix} - \begin{pmatrix} \mathbf{0} \\ \boldsymbol{v}_l \end{pmatrix} \right] \qquad (3.61)\\
&\quad + \begin{pmatrix} 0 & db & 0 & -db \\ db & 0 & -db & 0 \\ -k & k & -k & k \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -b & -b & -b & -b \end{pmatrix} \begin{pmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{pmatrix}.
\end{aligned}
$$

It is important to note that almost all terms in Equation (3.61) are time dependent, excluding $\boldsymbol{\Omega}$ and $\mathbf{g}_W$.

The hardware-specific constants b, k, d and ρ must be acquired empirically. The parameters for the AirRobot AR-100 used in this thesis necessary for the computation of Equation (3.61) are compiled through measurements (Section 3.3.4) and experiments (Section 4.2).

Now that the equations for the forces acting on the drone are formulated, we can calculate the robot's acceleration resulting from these forces, following Newton's second law of motion,

$$\mathbf{f} = m\mathbf{a}.$$

Given the spatial equivalent of mass in this equation, the constant inertia matrix, $\mathbf{I}$, the acceleration at time instant t can be calculated according to[30]

$$\dot{\mathbf{v}}(t) = \mathbf{a}(t) = \mathbf{I}^{-1}\left[\mathbf{f}_{res}(t)\right] + \mathbf{n}_a, \qquad (3.62)$$

where $\mathbf{I}$ is given in Equation (3.37) and $\mathbf{n}_a$ is the zero mean Gaussian process noise for acceleration.

Kinematic Model

In our case, the rotor speeds are arriving at a comparably slow rate of about 8Hz. This low rate is not well suited for the rotor-speed-driven dynamic model, where a high temporal resolution is beneficial. The IMU sensor delivers measurements at a rate of about 24Hz and is thus better suited to drive the model of the AR-100. The IMU already measures the linear acceleration caused by the sum of all forces. Therefore, instead of calculating the acceleration using rotor speeds, the IMU readings are employed as parameters for the process model. This disqualifies the use of the IMU during the correction step, since it is now being applied in the time update step.

Velocity

Given the acceleration, the change of velocity over a given time interval can be calculated. Suppose the time interval over which to calculate the change of velocity is defined as $[t_{k-1}, t_k]$, then $\Delta\mathbf{v}\left(t_k\right)$, given as

$$\Delta\mathbf{v}\left(t_k\right) = \int_{t_{k-1}}^{t_k} \dot{\mathbf{v}}\left(x\right) dx \tag{3.63}$$

$$= \int_{t_{k-1}}^{t_k} \mathbf{a}\left(x\right) dx, \tag{3.64}$$

defines that change of velocity. The total velocity at time instant t is given as

$$\mathbf{v}\left(t\right) = \mathbf{v}\left(t_0\right) + \int_{t_0}^{t} \mathbf{a}\left(x\right) dx. \tag{3.65}$$

For sufficiently small time intervals $[t_{k-1}, t_k]$, it can be assumed that the forces are constant during that time[12]. Then $\mathbf{a}\left(t_{k-1}\right) = \mathbf{a}\left(t_k\right)$, simplifying the integration. The change of velocity is given as

$$\Delta\mathbf{v}|_{t_{k-1}}^{t_k} = \mathbf{a}\left(t_{k-1}\right) \cdot \left(t_k - t_{k-1}\right). \tag{3.66}$$

The differential process model equation for evolution of the velocity is then given as the acceleration in Equation (3.62).

Position

The rate of change of the position, $\dot{\mathbf{p}}_B^*$, with respect to body frame coordinates and the orientation in Euler angles instead of quaternions, is given as

$$\begin{aligned}\dot{\mathbf{p}}_B^*\left(t\right) &= \mathbf{v}\left(t\right) \\ &= \mathbf{v}_0 + \int_0^t \mathbf{a}\left(x\right) dx.\end{aligned} \tag{3.67}$$

For a sufficiently small time interval, $[t_{k-1}, t_k]$, this results in the spatial motion vector as follows

$$\begin{aligned}\dot{\mathbf{p}}_B^* (t)|_{t_{k-1}}^{t_k} &= \mathbf{v}(t_{k-1}) + \int_{t_{k-1}}^{t_k} \mathbf{a}(x)\, dx \\ &= \mathbf{v}(t_{k-1}) + \mathbf{a}(t_{k-1}) \cdot (t_k - t_{k-1}) \\ &= \mathbf{v}(t_{k-1}) + \mathbf{a}(t_{k-1}) \cdot \Delta t.\end{aligned}$$

This velocity is still given with respect to the body frame. It has to be transformed using the body to world frame transformation matrix, ${}^W\mathbf{X}_B(t_{k-1})$, calculated from the position component vector, $\boldsymbol{p}(t_{k-1})$ of the system state. The transformed velocity is given as

$$\begin{aligned}\dot{\mathbf{p}}_W^* (t)|_{t_{k-1}}^{t_k} &= {}^W\mathbf{X}_B(t_{k-1})\, \dot{\mathbf{p}}_B^* (t)|_{t_{k-1}}^{t_k} \\ &= \mathbf{v}_W(t_k).\end{aligned}$$

During integration the body frame is regarded as not moving, simplifying the equations. In other words, during integration the non-inertial moving body frame is treated as a non-moving inertial reference frame. The motion trajectories are calculated for the still frame and afterwards transformed to world frame motions with the help of the transformation matrix of the static body frame. After integration, the body frame origin is defined as at the location of the center of mass of the UAV again, rotated correctly.

After conversion, $\dot{\mathbf{p}}_W^* (t)|_{t_{k-1}}^{t_k}$ is given in coordinates with respect to the world frame, but the angular velocity must be converted to quaternion representation. To achieve this, Equation (3.31) is applied, using the calculated world angle velocities from $\dot{\mathbf{p}}_W^* (t)|_{t_{k-1}}^{t_k}$. These angular velocities are the roll, pitch and yaw Euler velocities Φ, Θ and Ψ, given as

$$\mathbf{v}_W = \dot{\mathbf{p}}_W^* (t)|_{t_{k-1}}^{t_k} = \begin{pmatrix}\Phi(t) \\ \Theta(t) \\ \Psi(t) \\ v_N(t) \\ v_E(t) \\ v_D(t)\end{pmatrix}.$$

To convert these velocities into quaternion representation, the time derivative of the orientation quaternion, $\frac{d}{dt}\boldsymbol{q}$, is employed. Representing the rate of change of orientation, it is given as

$$\frac{d}{dt}\boldsymbol{q}(t) = \frac{1}{2}\mathsf{Q}(t)\,\omega(t) \tag{3.68}$$

$$= \frac{1}{2}\begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix} \begin{pmatrix} a\sqrt{\Phi^2 + \Theta^2 + \Psi^2}\,(1 - \|q\|) \\ \Phi \\ \Theta \\ \Psi \end{pmatrix}, \tag{3.69}$$

where $\mathsf{Q}(t)$ is generated from the current orientation at t_{k-1}. The angular velocities Φ, Θ and Ψ are calculated as follows:

$$\begin{pmatrix} \Phi \\ \Theta \\ \Psi \\ v_N \\ v_E \\ v_D \end{pmatrix} = {}^{W}\mathbf{X}_B\left(t_{k-1}\right)\left(\mathbf{v}\left(t_{k-1}\right)\right),$$

because to calculate equation (3.69), the angular speed is taken as constant during the given time frame.

The final seven-dimensional vector modeling the rate of change of the position vector in the system state is given as

$$\dot{\boldsymbol{p}}_W(t) = \begin{pmatrix} \frac{d}{dt}\boldsymbol{q}(t) \\ \boldsymbol{v}_W \end{pmatrix}` + \mathbf{n}_p. \tag{3.70}$$

The noise of the position rate of change, $\mathbf{n}_p$, is modeled as additive zero mean Gaussian noise. Its standard deviation is set to 0.01 for the quaternion component and $0.03\frac{m}{s}$ for the position.

IMU Bias, Motor Speeds, Wind Velocity

IMU sensor biases, the motor speeds and the wind velocity are all assumed to vary by a random walk model with zero mean Gaussian driving terms[37], given as

$$\begin{aligned} \dot{\boldsymbol{b}} &= \mathbf{n}_b, \\ \dot{\boldsymbol{m}} &= \mathbf{n}_m, \\ \dot{\boldsymbol{w}} &= \mathbf{n}_w. \end{aligned} \tag{3.71}$$

Their standard deviations are set according to empirical experience or actual available technical information. The standard deviation, σ, for the IMU biases are three-dimensional vectors with values set as $0.001\frac{rad}{s}$ and $0.01\frac{m}{s^2}$ for the angular and linear components. The standard deviation of the motor speed noise is set to $100RPM$ for each motor and the wind velocity standard deviation is set at a low $0.003\frac{m}{s^2}$. These are the differential process model equations for their corresponding system state components.

Process Noise Covariance Matrix

The process noise covariance matrix, $\mathbf{Q}$, represents the inverse of the accuracy of the process model equations. Or in other words, it denotes the uncertainty that the process model introduces into the state estimate. The process noise covariance matrix can be dynamic over time, but it is also common to define a constant noise covariance matrix to simplify the computations[38].

Some formulations also separate the process noise covariance matrix into two matrices, given as

$$\mathbf{Q} = \mathbf{L}\mathbf{Q}'\mathbf{L}^T, \tag{3.72}$$

where $\mathbf{Q}' \in \mathbb{R}^{n\times n}$ denotes a diagonal positive definite matrix and $\mathbf{L} \in \mathbb{R}^{n\times n}$ represents the corresponding lower triangular positive semi-definite matrix absorbing all non-diagonality of the noise covariance matrix. Accurate values for these matrices must be determined mathematically and empirically[8], but approximations thereof already deliver satisfactory results[26]. The accurate determination of the matrices $\mathbf{Q}'$ and $\mathbf{L}$ for the process noise covariance matrix would go beyond the scope of this thesis, hence the covariance matrix is defined with approximate values[37].

The process noise covariance matrix is defined as a diagonal concatenation of the noise covariance matrices of the system state vector components, given as

$$\begin{aligned}\mathbf{Q} = &\, diag\left[\mathbf{Q}_{n_p}, \mathbf{Q}_{n_v}, \mathbf{Q}_{n_b}, \mathbf{Q}_{n_m}, \mathbf{Q}_{n_w},\right] \\ = &\, diag\,\Big[0.0001, 0.0001, 0.0001, 0.0001, \\ &\quad 0.001, 0.001, 0.01, \left[\frac{m}{s}\right]^2 \\ &\quad 0.001, 0.001, 0.001, \left[\frac{rad}{s}\right]^2 \\ &\quad 0.04, 0.04, 0.04, \left[\frac{m}{s}\right]^2 \\ &\quad 0.001, 0.001, 0.001, \left[\frac{rad}{s}\right]^2 \\ &\quad 0.001, 0.001, 0.001, \left[\frac{m}{s}\right]^2 \\ &\quad 10000, 10000, 10000, 10000, [RPM]^2 \\ &\quad 0.00001, 0.00001, 0.00001 \left[\frac{m}{s}\right]^2\Big].\end{aligned} \tag{3.73}$$

In the IMU driven process model, the IMU noise statistics are used for the angular velocity and linear acceleration.

[8]The interested reader is referred to standard books on the subject, e.g. [38].

Summary

The Equations (3.70), (3.64) and Equations (3.71) above can then be summarized into the differential process model function, $\mathbf{f}$, given as:

$$\mathbf{f}\left(\mathbf{x}, t, \boldsymbol{w}\right) = \frac{d}{dt}\mathbf{x}\left(t\right) \tag{3.74}$$

$$= \frac{d}{dt}\begin{pmatrix} \boldsymbol{p}\left(t\right) \\ \mathbf{v}\left(t\right) \\ \boldsymbol{b}\left(t\right) \\ \boldsymbol{m}\left(t\right) \\ \boldsymbol{w}\left(t\right) \end{pmatrix} \tag{3.75}$$

$$= \begin{pmatrix} \dot{\boldsymbol{p}}_W\left(t\right) \\ \dot{\mathbf{v}}\left(t\right) \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{n}_p \\ \mathbf{n}_v \\ \mathbf{n}_b \\ \mathbf{n}_m \\ \mathbf{n}_w \end{pmatrix}, \tag{3.76}$$

where $\boldsymbol{w}$ is the parameter vector driving the process model. This function is integrated numerically over a given time interval to calculate the actual change of the system state.

3.4.3 Observation Model

The observation model calculates the measurements that are expected from each specific sensor, using information available in the system state. The function representing the observation model, $\mathbf{h}$, is a concatenation of several observation model functions, one for each sensor integrated into the Kalman filter framework. This observation model is defined employing the sequential updates modification to the correction step, described in Section 2.3.4.

Similar to the derivation of the process model, the componental observation models are derived first, after which the concatenated observation model equation is presented. As discussed in Section 2.3.4, these sensor specific models are employed sequentially as required during the correction process. Which model is applied when depends on the order in which the measurements arrive.

Knowledge of the time instant of a measurement's physical taking is important for an accurate correction of the state estimate with that measurement. Therefore the observations must be time stamped as precisely as possible.

The sensors' noise covariance matrices can be dynamic and static. Accurate determination of the noise covariance matrices requires precise knowledge or investigation of the noise characteristics. This process is costly and time-consuming[9]. And with the additional

[9]The interested reader is referred to research on stochastic processes and filter theory, e.g. [22]

handicap of industrial secrecy, the accurate determination of the matrices would have gone beyond the scope of this thesis. For this thesis the noise covariance matrices are approximated by constant matrices containing estimates of the specific sensor's noise statistics. These values are chosen from empirical experience of the sensors' measurements and from papers on state estimation[37, 12].

Measurement Vector

According to the formulation of the sequential updates, the measurement vector $\boldsymbol{z}$ is compartmentalized. It contains the measurements of the sensors defined in this section in an ordered fashion. The measurement vector and the order of its components is defined as

$$\mathbf{z} = \begin{pmatrix} \mathbf{z}_{quat} \\ \mathbf{z}_{imu} \\ \mathbf{z}_{tacho} \\ \mathbf{z}_{cam} \\ \mathbf{z}_{magnet} \\ \mathbf{z}_{baro} \end{pmatrix}. \tag{3.77}$$

The single measurement components are defined below.

Quaternion Unity

The unity of the quaternion must be maintained in order for the rotations to stay pure. This can be achieved through the introduction of a pseudo measurement concerning the quaternion's norm[37]. This pseudo measurement is applied at every correction step. The observation model is given as

$$\begin{aligned} \mathbf{h}_{quat} &= \hat{\mathbf{z}}_{quat}\left(t\right) + n_q \\ &= \|\boldsymbol{q}\left(t\right)\|^2 + n_q \\ &= q_0^2 + q_1^2 + q_2^2 + q_3^2 + n_q. \end{aligned} \tag{3.78}$$

The pseudo observation data received from the pseudo sensor is always 1, i.e.

$$\mathbf{z}_{quat}\left(t\right) = 1.$$

For the noise covariance matrix a very small 0.00001 is chosen.

Inertial Measurement Unit

The inertial measurement unit delivers observations of the angular velocities and linear accelerations of all three dimensions (Section 3.2.4). Additionally for the AR-100, it also reports the current pitch and roll angles. These angles are given with respect to the world frame, contrary to the motion information. The measurement vector is given as

$$\mathbf{z}_{imu}(t) = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \\ a_x \\ a_y \\ a_z \\ \Phi_W \\ \Theta_W \end{pmatrix}. \tag{3.79}$$

Therefore, the IMU observation model, $\mathbf{h}_{IMU}$, calculates the expected angular velocity and linear acceleration from the current state estimate. The linear acceleration is calculated by computing the forces acting on the UAV and deriving the acceleration with the help of Newton's second law of motion, analogous to the calculation of forces in section 3.4.2. The acceleration is given as

$$\mathbf{a}(t) = \mathbf{I}^{-1}\mathbf{f}_{res}(t), \tag{3.80}$$

where the total force, $\mathbf{f}_{res}(t)$, is calculated according to equation (3.61). The linear acceleration component is taken from this spatial motion vector.

The angular velocity, $\hat{\boldsymbol{\omega}}$, is directly available in the system state. It is the angular component of the spatial velocity, $\hat{\mathbf{v}}(t)$, where

$$\hat{\mathbf{v}} = \begin{pmatrix} \hat{\boldsymbol{\omega}} \\ \hat{\boldsymbol{v}} \end{pmatrix}.$$

The angles of the robot's orientation must be converted from the quaternion representation of orientation in the system state to Euler angle representation. Equation (3.30) is employed for this purpose, calculating the estimated roll and pitch angles, $\hat{\Phi}$ and $\hat{\Theta}$, respectively.

These results are then combined with the estimated IMU bias to represent the expected measurement, $\hat{\mathbf{z}}(t)$, given as

$$\begin{aligned} \mathbf{h}_{imu}(\boldsymbol{x}, t) &= \hat{\mathbf{z}}_{imu}(t) \\ &= \begin{pmatrix} \hat{\omega}_x \\ \hat{\omega}_y \\ \hat{\omega}_z \\ \hat{a}_x \\ \hat{a}_y \\ \hat{a}_z \\ \hat{\Phi} \\ \hat{\Theta} \end{pmatrix} + \begin{pmatrix} b_{\omega_x} \\ b_{\omega_y} \\ b_{\omega_z} \\ b_{a_x} \\ b_{a_y} \\ b_{a_z} \\ 0 \\ 0 \end{pmatrix} \end{aligned} \tag{3.81}$$

The noise characteristics of the IMU implemented in teh AR-100 are approximated as

$$\mathbf{R}_{imu} = diag\left[0.0001\left[\frac{rad}{s}\right]^2, 0.0001\left[\frac{rad}{s}\right]^2, 0.0001\left[\frac{rad}{s}\right]^2, \right.$$
$$\left. 0.04\left[\frac{m}{s^2}\right]^2, 0.04\left[\frac{m}{s^2}\right]^2, 0.04\left[\frac{m}{s^2}\right]^2, 0.0007421\,[rad]^2, 0.0007421\,[rad]^2\right]. \tag{3.82}$$

For the IMU driven process model, the measurement vector for the correction step as well as the corresponding noise covariance matrix are cropped to contain only the Euler angle measurements. Thus, the dimension of the measurement vector is 2 instead of 8.

Rotor Speeds

The rotor speeds are contained directly in the system state, $\boldsymbol{x}$, in the component vector $\boldsymbol{m}$. Thus, the observation model is simple, given as

$$\begin{aligned}\mathbf{h}_{tacho}(\boldsymbol{x}, t) &= \hat{\mathbf{z}}_{tacho}(t) \\ &= \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \end{pmatrix}.\end{aligned} \tag{3.83}$$

The noise covariance matrix is given as

$$\mathbf{R}_m = diag\,[10, 10, 10, 10]\,. \tag{3.84}$$

Camera

The camera sensor of the AR-100 is used to estimate its egomotion. The Kanatani egomotion estimation algorithm supplies measurements of the direction of the linear velocity, a unit vector, and the absolute angular velocity. Combined, this is a six-dimensional measurement vector.

As can be seen in Figure 3.8 of the AR-100, the camera is mounted at a -45° angle to the $\boldsymbol{x}$-axis. Thus the measurements have to be rotated around the $\boldsymbol{z}$-axis by $\alpha = -45°$. This is achieved by the constant spatial transformation matrix from camera to body frame, ${}^B\mathbf{X}_C$, given as

$${}^B\mathbf{X}_C = \begin{pmatrix} {}^B\mathbf{R}_C & \mathbf{0} \\ \mathbf{0} & {}^B\mathbf{R}_C \end{pmatrix},$$

where the rotation matrix is given as

$${}^B\mathbf{R}_C = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The inverse is the transpose of the matrix,

$$^{C}\mathbf{X}_{B} = {^{B}\mathbf{X}_{C}^{-1}} = {^{B}\mathbf{X}_{C}^{T}}.$$

The linear velocity contained in the state estimate must be normalized for the expected measurement vector, $\hat{\mathbf{z}}_{cam}(t)$. The spatial motion vector containing the estimated angular velocity and the normalized linear velocity is then transformed to the camera frame. The observation model is given as

$$\begin{aligned}\mathbf{h}_{cam}(\boldsymbol{x}, t) &= {^{C}\mathbf{X}_{B}}\hat{\mathbf{z}}_{cam}(t) \\ &= {^{C}\mathbf{X}_{B}} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \\ |\boldsymbol{v}|^{-1} v_x \\ |\boldsymbol{v}|^{-1} v_y \\ |\boldsymbol{v}|^{-1} v_z \end{pmatrix}. \end{aligned} \tag{3.85}$$

The measurement noise covariance matrix is approximated as

$$\mathbf{R}_{cam} = diag\left[0.01, 0.01, 0.01, 0.1, 0.1, 0.1\right]. \tag{3.86}$$

Magnetometer

The magnetometer observation is the amplitude of the rotational displacement of the body $\boldsymbol{x}$-axis away from world North. It is a two-dimensional vector containing the sine and cosine of the orientation angle. In effect, this describes the yaw angle of the robot attitude. That angle, Ψ, can be calculated from the current estimated orientation using Equation (3.30).

Once Ψ is calculated, the observation values must be obtained with the trigonometric functions, giving the magnetometer observation model of

$$\mathbf{h}_{magnet}(\boldsymbol{x}, t) = \hat{\mathbf{z}}_{magnet}(t) \tag{3.87}$$

$$= \begin{pmatrix} \cos\Psi \\ \sin\Psi \end{pmatrix}. \tag{3.88}$$

Since experiments were done indoors, where other magnetic sources can disturb or even completely overshadow the signal, the observation noise covariance matrix is given as

$$\mathbf{R}_{magnet} = diag\left[0.001, 0.001\right]. \tag{3.89}$$

Barometer

The barometer sensor onboard the AR-100 measures the current air pressure, but is somewhat susceptible to self-induced changes in air pressure due to air pushed by its rotors and possibly changes in temperature (Section 3.3.3). The reported measurement is the height of the drone in meters, but given in opposite direction of the Down-axis of the world NED frame. The height can be taken directly from the system state, only negating its coeffient.

The observation model is defined as

$$\begin{aligned}\mathbf{h}_{baro}\left(\boldsymbol{x},t\right) &= \hat{\mathbf{z}}_{baro}\left(t\right)\\ &= -p_z.\end{aligned} \tag{3.90}$$

Due to the occurrences of erroneous altitude measurements as discussed previously, the noise covariance matrix contains a relatively large value. The observation noise covariance matrix is given as

$$\mathbf{R}_{baro} = diag\left[14\left[m\right]^2\right].$$

Summary

The nonlinear observation model was derived and presented in this section. Due to the nature of sequential updates (Section 2.3.4), each sensor has its own observation model function. It can be seen that this separateness of the models lends itself well to the sensor fusion problem. Each sensor can be treated distinctly during the Kalman measurement update step, sequentially applying all available observations. This is another bonus of this formulation of the observation model and the Kalman filter modification. Their report rates can remain uncoupled, further simplifying the sensor synchronization and data fusion process. Each model function creates a corresponding measurement vector. These vectors can vary in their dimensions absolutely independently between different sensors.

The sequential application strategy of the measurements in whichever order those have become available for the current Kalman correction step generates a flexible observation model. Nonetheless, to summarize the separate sensor observation models, a concatenation of all different observation models presented in this section gives one view on the complete observation model. The concatenated observation model is given as

$$\begin{aligned}\mathbf{h}\left(\mathbf{X}\left(t\right),t\right) &= \begin{pmatrix}\mathbf{h}_{quat}\left(\mathbf{X}\left(t\right),t\right)\\ \mathbf{h}_{imu}\left(\mathbf{X}\left(t\right),t\right)\\ \mathbf{h}_{tacho}\left(\mathbf{X}\left(t\right),t\right)\\ \mathbf{h}_{cam}\left(\mathbf{X}\left(t\right),t\right)\\ \mathbf{h}_{magnet}\left(\mathbf{X}\left(t\right),t\right)\\ \mathbf{h}_{baro}\left(\mathbf{X}\left(t\right),t\right)\end{pmatrix}\\ &= \begin{pmatrix}\hat{\mathbf{z}}_{quat}\\ \hat{\mathbf{z}}_{imu}\\ \hat{\mathbf{z}}_{tacho}\\ \hat{\mathbf{z}}_{cam}\\ \hat{\mathbf{z}}_{magnet}\\ \hat{\mathbf{z}}_{baro}\end{pmatrix}\end{aligned} \tag{3.91}$$

3.4.4 Initialization Phase

At the beginning of the mission, before the UAV moves but while its sensors are already reporting measurements, the system state needs to be initialized to a reasonable configuration. For each state component this means something different. Some are initialized to zero, while other vector elements require complex computations.

The duration of the initialization phase must be chosen in such a way that every sensor required for the initializing computations reports at least as often as necessary for a meaningful average of the observations. In the case of the implementation using the AR-100, a duration of 20 seconds was chosen to provide an ample number of observations.

During the initialization phase, the measurement readings are counted in number and their values accumulated for each sensor separately. This is done in order to calculate the arithmetic means of the measurements. From those means the initial system state vector is calculated at the end of the initialization phase. That calculation at the end of the initialization phase is covered in this section.

Once the initialization phase is passed, the state estimator switches from initialization to estimation mode and treats the initialized system state and measurements according to the common predict-correct cycle of Kalman filters.

Position

The world frame must be aligned to the actual world NED orientation. That is, the NED orientation of the immobile UAV must be calculated employing the magnetometer and IMU measurement means. Given in the magnetometer measurements is the *North* (and *East*) orientation of the robot, and the IMU measurements contain the *Down* direction of the NED reference frame.

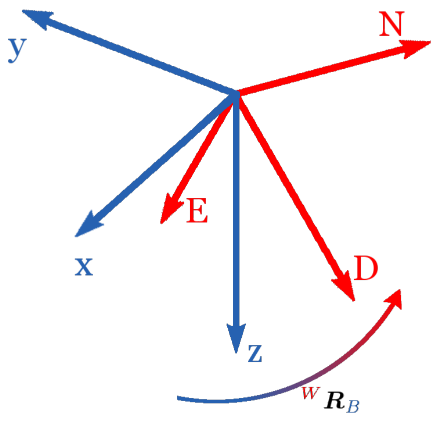


Figure 3.9: Initialization of robot orientation.
The rotation from the body frame to the NED orientation needs to be determined. Its inverse is the orientation of the robot within the world frame.

The idea is to calculate the body to world rotation, $^W\boldsymbol{R}_B$, needed for the current robot orientation, which is $\boldsymbol{q} = (1, 0, 0, 0)^T$, to be completely aligned with the NED frame axes. The inverse of that rotation is then the initial orientation of the robot within the world frame, $\boldsymbol{q}_{init}$. In order to obtain that rotation, the IMU and magnetometer measurement means need to be combined. This is achieved as follows:

1. Rotate $\boldsymbol{z}$-axis of body frame to align with Down-axis of world frame.
2. Rotate around $\boldsymbol{z}$-axis to align the $\boldsymbol{x}$- and $\boldsymbol{y}$-axes with the North- and East-axes of the world frame, respectively.

The rotation to align the $\boldsymbol{z}$-axis with the Down-axis is executed around a normal to the plane spanned by the two axes. It is important to note that the magnetic *North* information is not rotated along. A rotation could invalidate the orientation information, since the magnetometer doesn't compensate the North measurement for the robot's askew orientation out of the world frame North-East-plane. Rather, the North-axis information must be projected onto the $\boldsymbol{x}$-$\boldsymbol{y}$-plane after this first rotation of the $\boldsymbol{z}$-axis to compute the angle for the second rotation. Subsequently, after obtaining the North-direction in the current $\boldsymbol{x}$-$\boldsymbol{y}$-plane, the rotation around the current $\boldsymbol{z}$-axis is performed to align the $\boldsymbol{x}$-axis with the projected North axis. In doing so, the $\boldsymbol{y}$-axis is aligned with the East axis, since its orientation is determined by the orientations of the two other axes.

The normal to the $\boldsymbol{z}$-$\boldsymbol{D}$-plane is given as

$$\boldsymbol{n} = \boldsymbol{z} \times \boldsymbol{D}, \tag{3.92}$$

where $\boldsymbol{D}$ is the normalized vector of the linear acceleration readings, or in other words the three-dimensional direction of the Earth's gravity. It represents the axis of the planned rotation. The angle of the rotation, α, is calculated as follows,

$$\alpha = \arccos\left(\boldsymbol{z} \cdot \boldsymbol{D}\right).$$

Given the axis and angle of the rotation needed to align the $\boldsymbol{z}$-axis to the measured $\boldsymbol{D}$, it is converted into quaternion representation applying Equation (3.25),

$$\boldsymbol{q}_{rot_z} = \begin{pmatrix} \cos\frac{\alpha}{2} \\ n_x \sin\frac{\alpha}{2} \\ n_y \sin\frac{\alpha}{2} \\ n_z \sin\frac{\alpha}{2} \end{pmatrix}.$$

For the next step, the original North orientation, driven by the unit vector $\boldsymbol{N}$, is projected onto the plane spanned by the current $\boldsymbol{x}$- and $\boldsymbol{y}$-axes. The parallel component of the projection is formulated as

$$\boldsymbol{N}_{xy} = \boldsymbol{D} \times \left(\boldsymbol{N} \times \boldsymbol{D}\right).$$

The component of the North direction parallel to the $\boldsymbol{x}$-$\boldsymbol{y}$-plane, $\boldsymbol{N}_{xy}$, is the orientation that the $\boldsymbol{x}$-axis is supposed to be rotated to. $\boldsymbol{D}$ is the normal of that plane, the current $\boldsymbol{z}$-axis. Therefore the rotation axis is the current $\boldsymbol{z}$-axis. The angle for the rotation is given as

$$\beta = \arccos\left(\boldsymbol{x} \cdot \boldsymbol{N}_{xy}\right).$$

This yields a quaternion representation of the second rotation to align the $\boldsymbol{x}$-axis to the projected North direction as follows,

$$\boldsymbol{q}_{rot_x} = \begin{pmatrix} \cos\frac{\beta}{2} \\ D_x \sin\frac{\beta}{2} \\ D_y \sin\frac{\beta}{2} \\ D_z \sin\frac{\beta}{2} \end{pmatrix}.$$

The combined rotation for the body frame to align with the world frame is then given as

$$\boldsymbol{q}_{rot} = \boldsymbol{q}_{rot_x} \cdot \boldsymbol{q}_{rot_z},$$

with respect to the world frame. The initial orientation of the robot is the inverse of that rotation, as presented in section 3.2.3, given as the quaternion's conjugate as

$$\boldsymbol{q}_{init} = \boldsymbol{q}^*_{rot} = \begin{pmatrix} q_{rot,0} \\ -q_{rot,1} \\ -q_{rot,2} \\ -q_{rot,3} \end{pmatrix}.$$

The linear position of the robot is initialized to zero. Combined with the robot's orientation, the initial seven-dimensional position vector of the system state is given as

$$\boldsymbol{p}_{init} = \begin{pmatrix} \boldsymbol{q}_{init} \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3.93}$$

Motor Speeds

The motor speeds and thereby rotor speeds are initialized to the arithmetic mean of the set of measurements, $\mathcal{S}_m$, received during the initialization phase. This should be the approximate idle rotor speed, since the robot must remain immobile during the initialization phase. The initial values are given as

$$\boldsymbol{m}_{init} = \frac{1}{|\mathcal{S}_m|} \sum_{\forall \boldsymbol{m} \in \mathcal{S}_m} \boldsymbol{m}. \tag{3.94}$$

Velocity, IMU Bias and Wind Velocity

The vehicle velocity, IMU bias and wind velocity are all initialized to zero.

$$\mathbf{v}_{init} = \mathbf{0}, \tag{3.95}$$
$$\boldsymbol{b}_{init} = \mathbf{0}, \tag{3.96}$$
$$\boldsymbol{w}_{init} = \mathbf{0}. \tag{3.97}$$

Chapter 4

Experiments and Results

4.1 Chapter Outline

In the course of this chapter, the conducted experiments concerning the robot's developed physical model and its implementation are presented.

The first experiment results to be given are of the experiment investigating the relationship between the unit of the robot's tachometer and the generated rotor speeds. The observed relationship is used to calculate a proportionality constant. That constant can then be used in the process model to convert from the tachometer units to a standard rotation velocity.

The experiments to obtain the necessary physical parameters for the model of the AR-100 are presented next. The generated thrust in relation to a given rotor speed must be predicted. Therefore the proportionality between rotor speed and generated thrust of a single rotor is investigated. The constant describing this relationship is called the rotor coefficient. This coefficient is determined for the rotors implemented in the AR-100.

Another relation is needed for the process model (Section 3.4.2), the proportionality between relative wind velocity and generated drag force. That proportionality is examined to determine a conversion function. The AR-100 air drag characteristics are researched in detail.

After characteristic parameters of the model have been found and determined, the results of experiments scrutinizing the performance of the implementation are presented and discussed.

4.2 Physical Model

Parameters describing the physical model of the AR-100 quadrocopter need to be determined empirically. Additionally, the proportionality constant for the conversion from the I^{2}C values to RPMs was unknown because of it was classified an industrial secret. Therefore it is obtained empirically in this section.

4.2.1 IIC to RPM

Since the tachometer readings of the AR-100 are reported in an unknown unit of the I^{2}C protocol implemented by AirRobot, the conversion function from that unit to RPM had to be determined. A relation of the I^{2}C values to the voltage at the motors is assumed. Because of the voltage to revolution speed characteristics of electric motors, as implemented in the AR-100, a quadratic relationship is expected. The model equation is then of the form

$$y = ax^2 + bx + c, \tag{4.1}$$

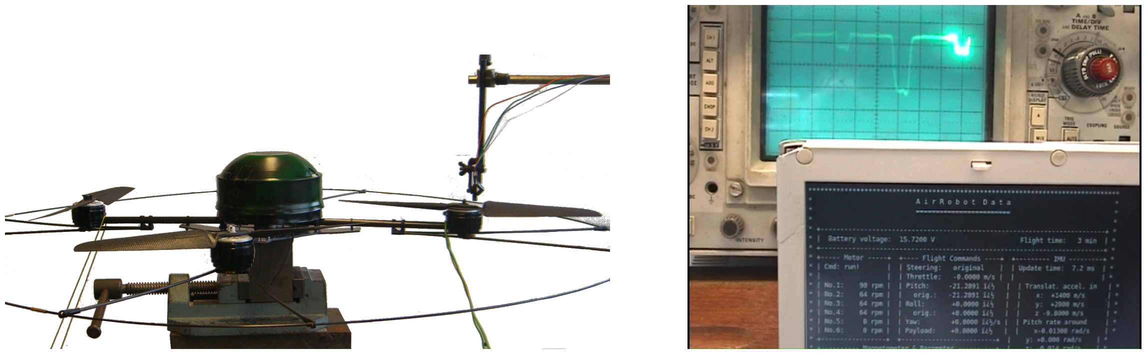
where x is the I^{2}C value and y represents the corresponding rotor speed in [RPM]. a, b and c are scalar constants. This relationship model is needed for the conversion of incoming tachometer measurements into a common physical unit. Once the measurement data are given with respect to a unit, here specifically revolutions per minute, they can be employed in the UAV's process and observation model equations.

Experimentation Setup

In order to measure the relationship between the I^{2}C values and the rotor speed, the drone was immobilized. One rotor and motor pair was chosen as representative for all four rotors, in this case rotor number 1. With a phototransistor positioned close above the rotor, the rotor's running time was measured and displayed by an oscilloscope. A piece of white tape was affixed to one blade of the rotor to improve the signal contrast. The speed of the rotor was adjusted by controlling the I^{2}C value for the motor indirectly via incremental keyboard control commands. The keyboard control has the positive property that the control commands can be set in incremental steps to a constant value (Section 3.2.4). This lets the UAV rotor remain at a more or less constant speed[1], allowing accurate measuring of the running time. The experimentation room was protected against wind. A video camera was setup so that it could record both the oscilloscope display as well as the I^{2}C values received from the quadrocopter, which were displayed on a computer screen. In this way, a simple synchronization of the oscilloscope readings and the reported I^{2}C values was achieved. Figure 4.1 illustrates the experimentation setup of the quadrocopter.

Unfortunately, the control commands sent to the aircraft are not direct rotor speed values but desired attitude angles of the robot. The AR-100 employs an attitude and height stabilization algorithm onboard that includes the IMU sensor readings as well as the barometer measurements. This can cause jumpy variations in the I^{2}C values set at the motors and consequently the rotor speeds, because the quadrocopter compensates for sudden imaginary drops or increases in height. The I^{2}C values displayed on the computer screen nevertheless are the exact values of the tachometer, but lagging behind the current state by the signal

[1]Constant speed insofar, as that the interferences by the stabilization control are not considered. The attitude control can always affect the I^{2}C values unrelated to the control commands.



```
A i r R o b o t  D a t a

| Battery voltage:  15.7200 V                          Flight time:   3 min |

+----- Motor -----+  +---- Flight Commands ----+  +--------- IMU ---------+
| Cmd: run!       |  | Steering:   original     |  | Update time:  7.2 ms  |
|                 |  | Throttle:   -0.0000 m/s  |  |                       |
| No.1:   90 rpm  |  | Pitch:     -21.2891 ï¿½  |  | Translat. accel. in   |
| No.2:   64 rpm  |  |   orig.:   -21.2891 ï¿½  |  |   x:  +1400 m/s       |
| No.3:   64 rpm  |  | Roll:       +0.0000 ï¿½  |  |   y:  +2000 m/s       |
| No.4:   64 rpm  |  |   orig.:    +0.0000 ï¿½  |  |   z -9.0000 m/s       |
| No.5:    0 rpm  |  | Yaw:        +0.0000 ï¿½/s|  | Pitch rate around     |
| No.6:    0 rpm  |  | Payload:    +0.0000 ï¿½  |  |   x-0.01300 rad/s     |
+-----------------+  +--------------------------+  |   y: +0.000 rad/s     |
```

(a) Immobilized UAV for rotor speed measurement. (b) Oscilloscope and PC Monitor (camera view).

Figure 4.1: I^{2}C-RPM Experiment Setup.

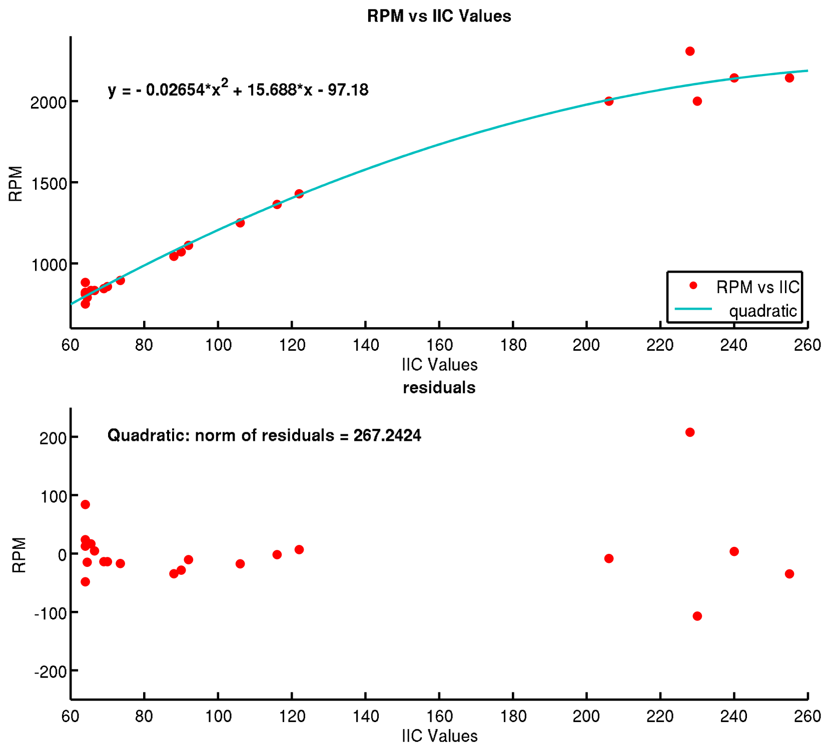
latency inherent in the wireless communication system. The attitude stabilization program of the AR-100 cannot be switched off.

Results and Discussion

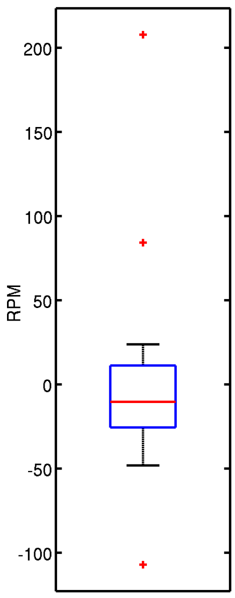
Figure 4.2 shows the results of the first experiment. Its focus was on obtaining a crude overview of the possible values and a preliminary test of the conversion model. The measured speeds of the rotor in relation to the set I^{2}C values show a trend supporting the expectation for the proportionality relationship. The residuals in figure 4.2(b) show no discernible tendency for small I^{2}C values to suggest an incapability of the chosen model to represent the actual system. For higher I^{2}C values, the residuals appear larger. This means, that for higher I^{2}C values, the model seems to represent the actual relationship between I^{2}C values and rotor speeds less accurately. A more detailed analysis of the distribution of the residuals of this preliminary experiment is appropriate.

The boxplot analysis of the residuals (Figure 4.2(b)) shows a slight skewness of the distribution to below the center, indicated by the low positions of the box and the median. The skewness magnitude is roughly -10RPM. Relative to the rotor speed amplitudes, this skewness of the residuals' distribution is small and can be neglected. The box size implies that a high number of sample cases is contained within a dense segment around the fit function. In other words it signifies a high peakedness of the data distribution toward the model function. Few outliers are encountered, but more of them lie above than below the majority of samples. This can be a result of fluctuations in the motor speed based on interferences by the attitude stabilization software. The low number of outliers also depends on the low number of samples. These results support the suggested model but encourage further investigation.

These initial experiment results do not yet represent the full I^{2}C value spectrum, $[64, 256]$, at a satisfying resolution. Hence, in a second experiment the rotor speeds were measured



(a) Plot of I^{2}C values to RPM, with quadratic fit.

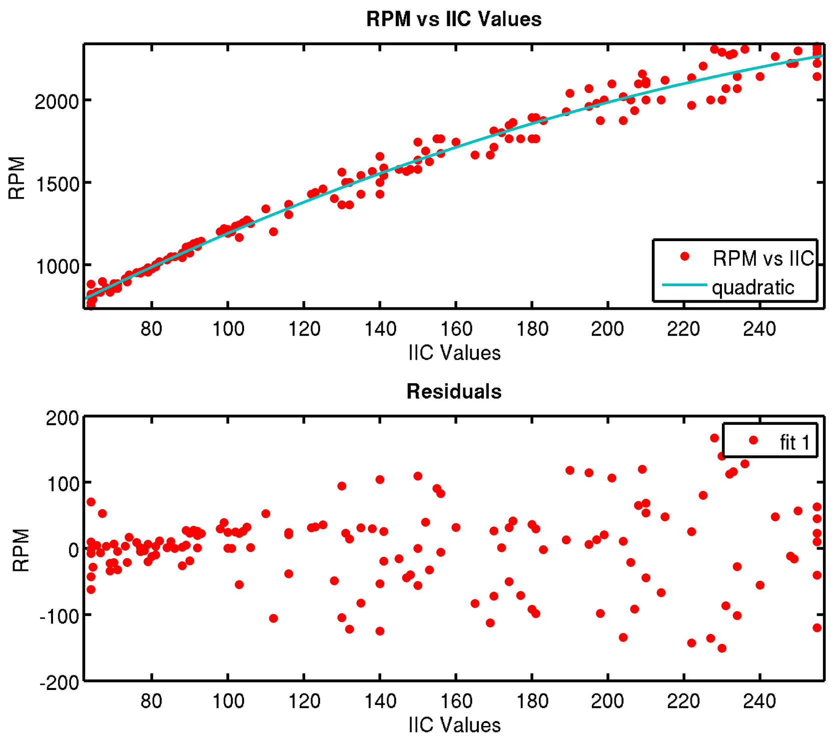


(b) Boxplot analysis.

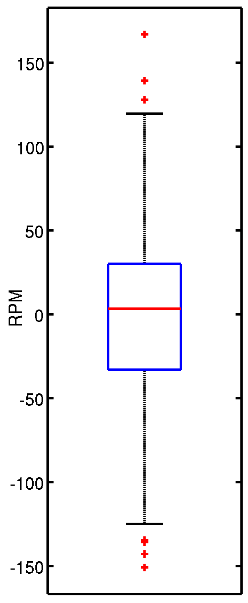
Figure 4.2: Results of preliminary I^{2}C to RPM experiment with quadratic fit.
The samples already show a trend supporting the suggested model. Residuals do not appear skewed, but the boxplot analysis shows a small negative skewness.

at smaller increments of the I^{2}C value scale. Figure 4.3 presents the complete measurement results. The proposed model relationship, which is a quadratic fit, and the corresponding residuals are shown. From the plot, the correlated behavior of the measurements to the expected proportionality relation is discernible. With growing I^{2}C values, this correlation looses its focus and strays further from the expectation. This is more obvious in the plot of the residuals, but the errors to the fit seem scattered without a clear trend. This suggests that the larger residuals at higher rotor speeds can be an effect of a degrading measuring accuracy at those higher speeds. In-depth analysis of the residuals will address this possibility.

Analysis of the residuals via boxplot (Figure 4.3(b)) shows the median of the residuals to be zero. In accordance, the box is well positioned around zero, located in the middle between the whiskers. These two properties reinforce the lack of skewness in the residual distribution. In other words, the expectation is not skewed in some direction with respect to the measurements. The box size, indicating the kurtosis, or peakedness, of the distribution, is compact in relation to the whiskers. This symbolizes a focused distribution of data around



(a) Plot of I^2C values to RPM, with quadratic fit.



(b) Boxplot analysis.

Figure 4.3: Results of rotor speed to I^2C values experiment with quadratic fit. A clear trend of the RPM measurements in relation to the I^2C values can be seen. The median is zero, but the whiskers show a larger spread of the samples. The kurtosis nevertheless reinforces the compactness of the samples around the expectations.

the model, suggesting that the fitted function reflects the actual relation well. Few outliers are discovered, but they remain equally distributed above and below the whiskers. The RMSE is $\epsilon_{RMSE} = 61.42RPM$, which represents a small error of about 2.5%, considering the spectrum of rotor speeds spans up to 2400RPM.

Conclusion

In this section, the proportionality relationship between an AR-100 specific I^2C unit and the rotor speed are investigated. A common proportionality model of an electric motor mapping voltage to motor speed is tested. The results of this experiment do not contradict the suggested model for the relationship between the I^2C values and rotor speeds. They solidify the results of the preliminary test further, offering a relationship of satisfying quality for use in the conversion from I^2C values to rotor speeds. In consequence, the calculated values for the three required conversion parameters a, b and c are used for the conversion of I^2C values to RPM in equation (4.1). These values are given as

$$\begin{aligned} a &= -0.018783505902714, \\ b &= 13.586956245146235, \\ c &= 19.417077590370596. \end{aligned}$$

For future work, more accurate parameters for the conversion could be obtained by controlling the I^2C values directly. Removing or deactivating the onboard attitude stabilization program could improve experiment results further. The wireless transmission of the I^2C signals introduced a certain delay to the displayed values in comparison with the measured rotor speeds on the oscilloscope. Though the measurements were taken during intervals of apparent constant velocity, a direct cable connection would help minimize the latency between I^2C values and measured rotor speeds. Thereby this also minimizes the effect of sudden rotor speed adjustments by the attitude stabilization program, whose I^2C values arrive later than the effect is observable through the phototransistor and oscilloscope.

4.2.2 Rotor Coefficient

In order to calculate the force generated by a spinning rotor, the relationship between rotation speed and generated thrust can be modeled as

$$f = bw^2, \tag{4.2}$$

as discussed in section 3.4.2. To test this model and to determine a value for the proportionality constant b, the induced thrust with respect to the rotor speed was investigated.

Experimentation Setup

In the interest of measuring the thrust force generated by the spinning rotor, the AR-100 was attached rigidly to a real time scale. The scale was tared for the mass of the vehicle and the immobilizing materials combined at zero rotor speed. The fixing had the dual purpose of immobilizing the UAV and connecting it to the measuring scale. A monitor showing the current rotor speeds was positioned beneath the scale display. For synchronization, analogously to the first experiments, a camera was setup to capture the weight and rotor speed readings (Figure 4.4). The generated force was calculated applying Newton's second law of motion, multiplying the measured weight with the acceleration by gravity.

Since the rotors cannot be switched off separately, only the collectively generated thrust can be measured. In order to calculate the thrust of a single rotor accurately, the rotor speeds are incremented concurrently. To increase the sample resolution, rotor speeds are also incremented separately. The separate incrementation introduces the problem that now the rotor speed corresponding to a fourth of the measured collective thrust is not known directly. Nevertheless, that speed can be calculated from the sum of the squares of the individual rotor speeds as follows: Because the measured collective force, $f_r = f_{res}$, could

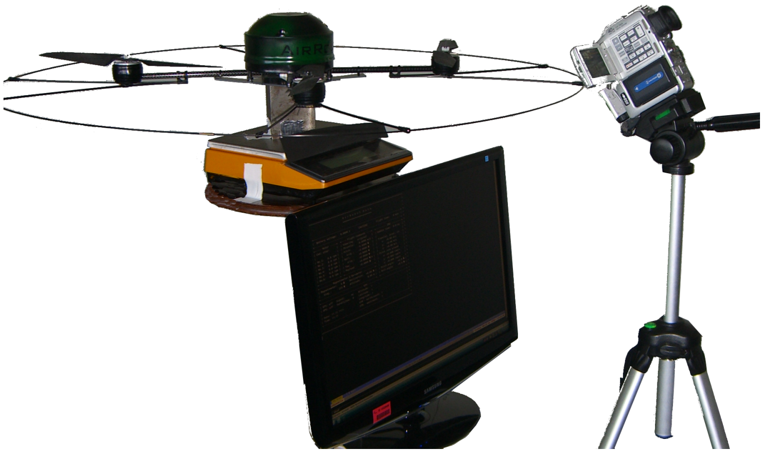


Figure 4.4: Rotor Coefficient Experiment Setup.
The UAV is immobilized and attached to a scale. A camera captures the weight and rotor speed readings.

be generated by 4 individual rotors[2] running at equal speeds, w_r, it follows from Equations (3.51) and (4.2) that

$$f_r = f_{res} \tag{4.3}$$

$$4bw_r^2 = b\left(w_1^2 + w_2^2 + w_3^2 + w_4^2\right) \tag{4.4}$$

$$w_r^2 = \frac{\left(w_1^2 + w_2^2 + w_3^2 + w_4^2\right)}{4} \tag{4.5}$$

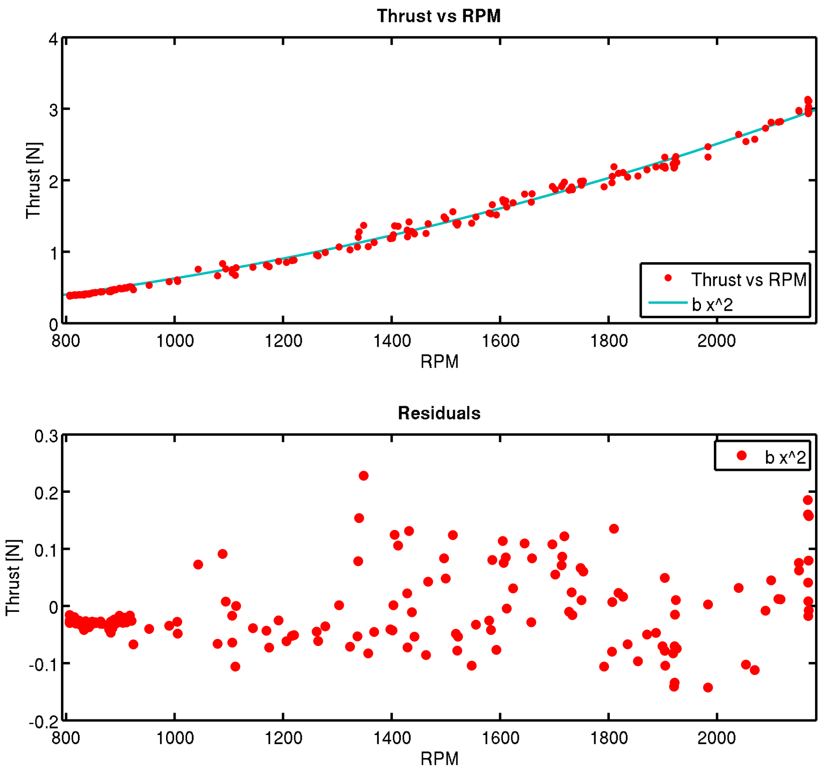
$$w_r = \frac{1}{2}\sqrt{w_1^2 + w_2^2 + w_3^2 + w_4^2}. \tag{4.6}$$

In other words, from the real and possibly distinct speeds of the rotors a virtual rotor speed w_r is calculated, with which the same collective thrust could be generated, if all rotors were running at that speed, w_r. This calculation is independent from the sought-after proportionality constant, b. The rotor speeds given in the results are always computed using Equation (4.6). The plotted thrust forces represent the force component generated by a single such rotor, which is the measured thrust divided by 4.

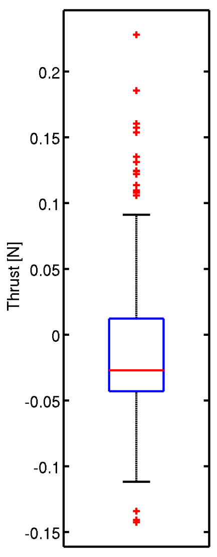
Results and Discussion

Figure 4.5 presents the results of this experiment. The thrust measurements show an obvious trend supporting the suggested proportionality model of Equation (4.2). The measurements stay close to the expected values, even for high rotor speeds. Overall the model seems to predict the generated thrust a little higher than the actual observations. Considering the residuals, this overshoot is reflected more clearly. The majority of the

[2]The rotors must be of the same type, i.e. with the same rotor coefficient as the actual rotors.



(a) Thrust to rotor speed plot, with $f = bw^2$ fit.



(b) Boxplot analysis.

Figure 4.5: Results of rotor coefficient experiment.
The suggested model captures the proportionality between rotor speed and induced thrust, but is skewed to above the actual relationship.

residuals seems to be negative. This skewness of the model to overestimate the thrust magnitudes is especially notable at low rotor speeds.

On closer inspection of the residuals (Figure 4.5(b)), the skewness of the distribution is evident in the median. The distribution is skewed by about 0.025N to the negative of the expectation, which is less than 1% of the approximate maximum generated thrust of the rotor for maximum rotor speeds. The skewness of the distribution can also be noticed by the lowered location of the box between the whiskers. On the other hand, while the skewness is relatively small, the box size is compact. This, an indicator for the kurtosis of the plotted distribution, signifies a high number of samples lie in a small band around the expectation function.

Additionally the box position is located closely around zero for the residuals. This signifies that the samples are spread densely around or just below the expected values. In other words, the model describes the relationship between thrust and rotor speed slightly skewed, but captures the proportionality insofar correct, as that the real measurements are still focused closely around the expectation. More outliers are detected above than below the whiskers. These could be caused by sudden interferences of the onboard stabilization control software in the motor speed regulation.

The root mean squared error, is $\epsilon_{RMSE} = 0.06602N$. This represents about 2% of the maximum thrust, which signifies a good fit of the model to the actual proportionality relationship. At the highest rotor speeds, the residuals are mostly positive. This means that the measurements are higher than the expectations. The robot was observed to be shaking at these high rotor speeds, notwithstanding the immobilizing fixtures. This might have affected the weight readings of the scale, but should result in equally distributed errors. The almust purely positive errors of the measurements can be an effect of the low resolution of the I^{2}C values, treated in the previous experiment. The 192 I^{2}C values from [64,256] map a rotor speed interval of about 800RPM to more than 2400. The maximum I^{2}C value could represent a power value at the motor that is less than the factual maximum power value for the motor. Thus, while the AR-100 tachometer reports one rotor speed, the actual rotor speed could be higher.

Conclusion

This section examines the thrust characteristics of one rotor in relation to its rotation speed. It could be shown that the proposed model for induced thrust with respect to rotor speed reflects the characteristics of the relationship to a satisfying degree. The rotor coefficient, b, is then given as the parameter calculated for Equation (4.2),

$$b = 6.273 \cdot 10^{-7} Ns.$$

For a more accurate determination of the rotor coefficient, a separate experimentation bench can be built. That setup should be for a single rotor using a motor with accurate speed control settings. The motor and rotor can be mounted horizontally on a vertical movable pole so that it lifts upwards. The movable pole is the force meter to measure the generated thrust. At the same time, the pole should be able to rotate. By the force of the rotational motion, it can measure the generated torque by the rotor. Thus the second rotor coefficient for induced torques, k, can be investigated as well.

4.2.3 Air Drag Coefficient

To be able to implement the effect of air drag on the flying robot into the model formulations, the UAV's air drag proportionality constant must be determined. That constant, ρ, is a fusion of multiple arguments of the drag equation. Thus, we obtain the proportionality relation from the one-dimensional drag equation as follows

$$\begin{aligned} f_d &= -\frac{1}{2}\varrho C_d A v_{rel}^2 \\ &= -\frac{1}{2}\rho v_{rel}^2, \end{aligned} \tag{4.7}$$

where ρ combines the drag coefficient, C_d, the air density, ϱ, and the reference area, A, orthogonal to the relative velocity of the object with respect to the air, v_{rel}. This combination is useful, because no statement about the drag coefficient or the reference area of the AR-100 is available. The three-dimensional model for air drag characteristics of the UAV is then obtained from the one-dimensional in equation (4.7). The 3-D formulation is given as

$$\boldsymbol{f}_d = -\frac{1}{2}\boldsymbol{\rho}\left\|\boldsymbol{v}_{rel}\right\|\boldsymbol{v}_{rel}, \tag{4.8}$$

where $\boldsymbol{\rho}$ is the vector of the proportionality constants for each dimension. If the constants of the different dimensions do not differ much, to a certain degree, then $\boldsymbol{\rho}$ can become a scalar constant to simplify the calculation.

Since the robot can be exposed to wind from all directions, its air drag characteristics are investigated by having the wind come from different angles of attack.

Experimentation Setup

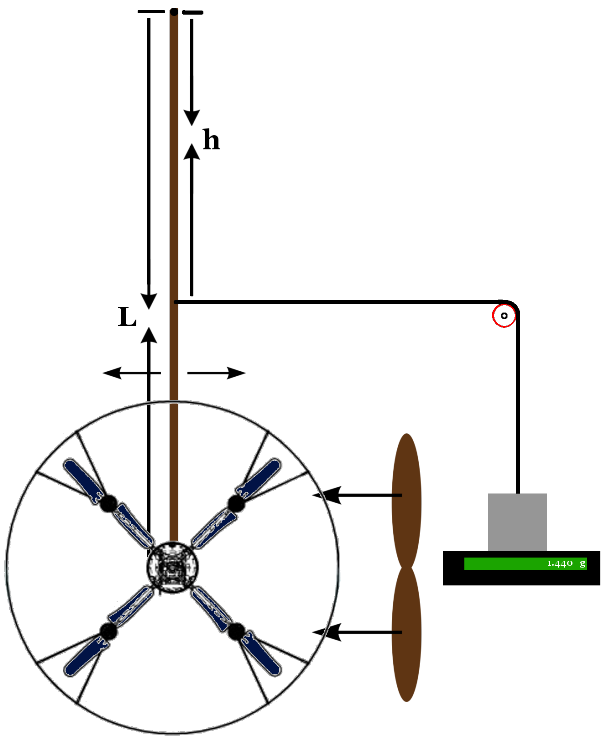


Figure 4.6: Side view of the drag coefficient experimentation setup.
Rigid pendulum can swing only along indicated direction, parallel to the page plane. In the pictured setup the $\boldsymbol{x}$-$\boldsymbol{y}$-plane is examined.

The experiment to examine the air drag properties of the AR-100 was based on the same principle as before. The scale from the previous experiments was reused to measure weight from which to calculate the force causing that change of weight, applying Newton's second law of motion using gravitational acceleration. The distinct difference this time was that the UAV was attached to a vertical, one-dimensional pendulum. The pendulum, when displaced, pulls via a direct 1:1 pulley system vertically on a weight attached to the scale (Figure 4.6). From that change of weight the drag force is calculated as explained above. It is important to note that the drag force acting on the robot is located further down on the pendulum than the pulley line affecting the scale. The pendulum acts as a lever, specifically of the second class. The relation between the two forces can be calculated with the help of the leverage principle. The principle says that the effort force, f_E, times its distance from the fulcrum, l_E, is equal to the load force, f_L, multiplied with its distance from the fulcrum, l_L, in other words

$$f_E l_E = f_L l_L. \tag{4.9}$$

Here, the effort force is the drag force, f_d, at length L from the pendulum axis, and the load force is the force to lift the weight on the scale, denoted as f_s, situated at distance h from the fulcrum (Figure 4.6). For given f_s, the drag force can be obtained by

$$f_d = \frac{h}{L} f_s. \tag{4.10}$$

Regarding the pendulum from the perspective of the wind it acts like a wheelbarrow, with the weight that's on the scale placed inside the wheelbarrow: The pendulum lever makes it easier for the drag force to lift the weight on the scale. The distances for L and h were measured to be

$$\begin{aligned} h &= 1.222m, \\ L &= 2.057m. \end{aligned}$$

The wind source was a 30“ wood propeller for model airplanes mounted on a power drill. The diameter of the rotor spans the diameter of the quadrocopter, producing a wind affecting the whole reference area of the UAV, similar to real wind. The power drill was finely adjustable, allowing a wind velocity spectrum of high resolution. The air drag characteristics were also examined for wind speeds so high that the drone would not be flown in them under normal circumstances. For the experiment with stopped rotors, the wind source was different to the experiments with running rotors. In that case, the wind source was a ventilator with a smaller diameter. The wind velocity was measured with a mobile, calibrated anemometer.

To investigate the components of the air drag properties separately along the $\boldsymbol{x}$-$\boldsymbol{y}$-plane and the $\boldsymbol{z}$-axis, the quadrocopter is mounted as shown in Figure 4.7 for the $\boldsymbol{x}$-$\boldsymbol{y}$-plane, and turned 90° for the $\boldsymbol{z}$-axis. Furthermore, the wind source was placed at different angles for the experiments concerning both the $\boldsymbol{x}$-$\boldsymbol{y}$-plane and the $\boldsymbol{z}$-axis. The wind source angles for the $\boldsymbol{x}$-$\boldsymbol{y}$-plane experiments are 0°, 10°, 20°, 30°, −10°, −20°, −30° and −40°. The $\boldsymbol{z}$-axis characteristics were investigated for wind coming from angles 0°, 120°, 110°, 100° and 180°. Figure 4.7 shows the coordinate system for the angles.

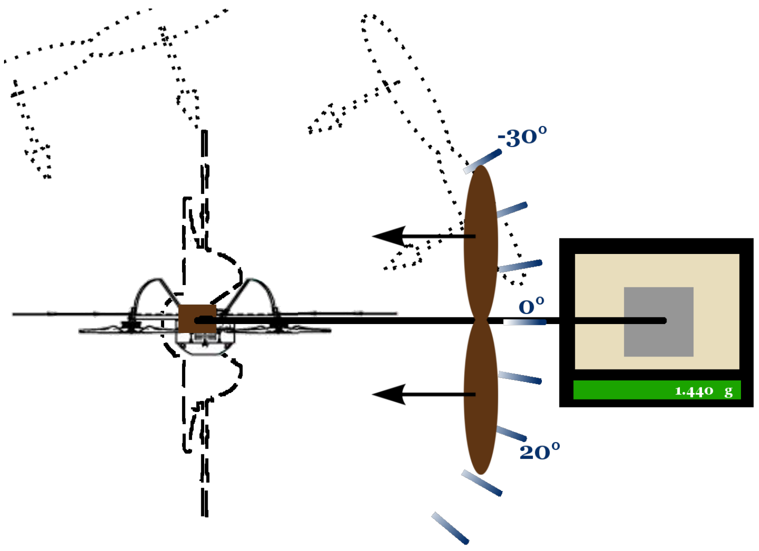


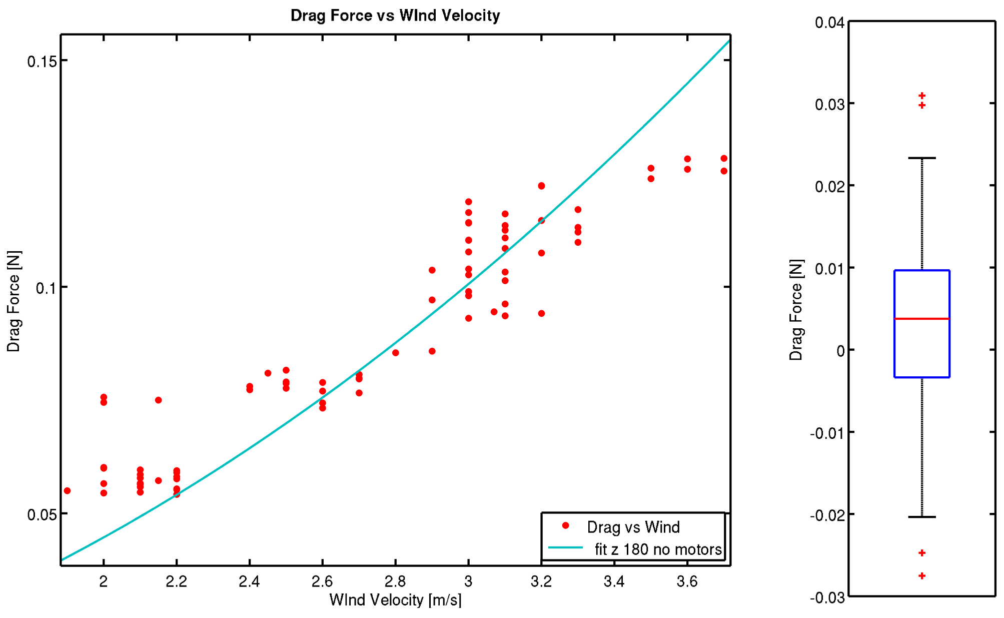
Figure 4.7: Top view of drag coefficient experimentation setup.
The wind source is placed equidistant at different angles around the UAV for different angles of attack. The copter is mounted either for investigation concerning the ***x*-*y***-plane or the ***z***-axis (dotted UAV outline) characteristics.

Results and Discussion

For a preliminary test, the ***z***-axis air drag properties of the UAV at rest were examined. Since the air drag characteristics differ immensely between stopped and running rotors[36], and the effect of air on the robot is for us only interesting during an active mission, the results for passive rotors are discussed only in brief. Afterward, the characteristics of the active UAV, i.e. with running motors, are investigated in more detail.

Stopped Rotors The measurements of this experiment seem at first glance wildly scattered and not too supporting of the suggested model (Figure 4.8). But a look at the scale of the vertical axis (Force) shows that these measurements are plotted at a very high resolution. The boxplot analysis shows the behavior of the measurements in relation to the proposed proportionality model more clearly. The median and the box position indicate a definite skewness of the residuals to above the expectations. In other words, this signifies that the suggested relationship underestimates the actual air drag coefficient. The kurtosis of the residual distribution is nevertheless rather compact, implying a dense distribution of measurement samples around the model function.

One reason the fitted model could be too low is the use of the different wind source. The ventilator employed in this experiment with stopped motors was a little smaller than the diameter of the quadrocopter. This could lead to a smaller effect of the wind on the robot, if the more prominent reference area elements are not affected. This results again in the induction of a smaller drag force than would normally be observed at the measured wind



(a) 180 Degrees along z-axis. Stopped motors. (b) Residuals

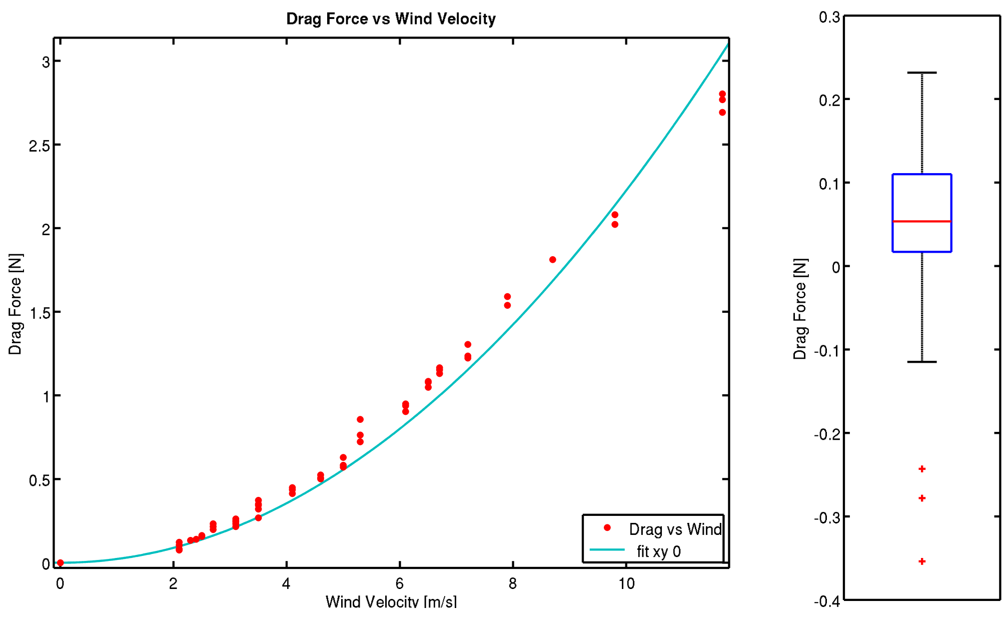
Figure 4.8: Results of air drag experiment, no motors are running.
Forces are plotted at a very high resolution. Slight skewness of residuals is evident from median and box position. A dense peakedness of the residuals' distribution can be seen from the box size.

speed. Thus, the measured drag forces are sometimes lower than the correctly measured ones, affecting the fit of the model to the negative. The RMSE for this fit is given as $\epsilon_{RMSE} = 0.01137N$, which still signifies a mediocre fit at 13% of the median of measured drag forces. The air drag coefficient for Equation (4.8) calculated for this experiment is

$$\rho_{z,180°,stopped} = 0.02236\frac{kg}{m}.$$

Running Rotors The air drag characteristics of the AR-100 with running rotors were investigated in more detail. Results of specific experiments are chosen as representatives for experiment results of several angles. Nevertheless, the results of the remaining experiments are presented and discussed in Appendix B.

The measurements for wind coming from 0° (Figure 4.9) show a clear tendency to behave like the suggested relationship. The fitted function for the most part underestimates the actual drag forces. Only for maximum wind velocities do the actual force measurements fall below the expectations. In the boxplot analysis of the fit residuals, these three measurements are classified as outliers. The readings of these outlying measurements might have been contaminated from the very turbulent flight of the UAV at these high wind speeds. The velocity with which the wind was coming at the UAV was very high, $> 10\frac{m}{s}$. This sometimes



(a) 0° $\boldsymbol{x}$-$\boldsymbol{y}$-plane (b) Residuals

Figure 4.9: Drag coefficient experiment results for 0° $\boldsymbol{x}$-$\boldsymbol{y}$-plane.

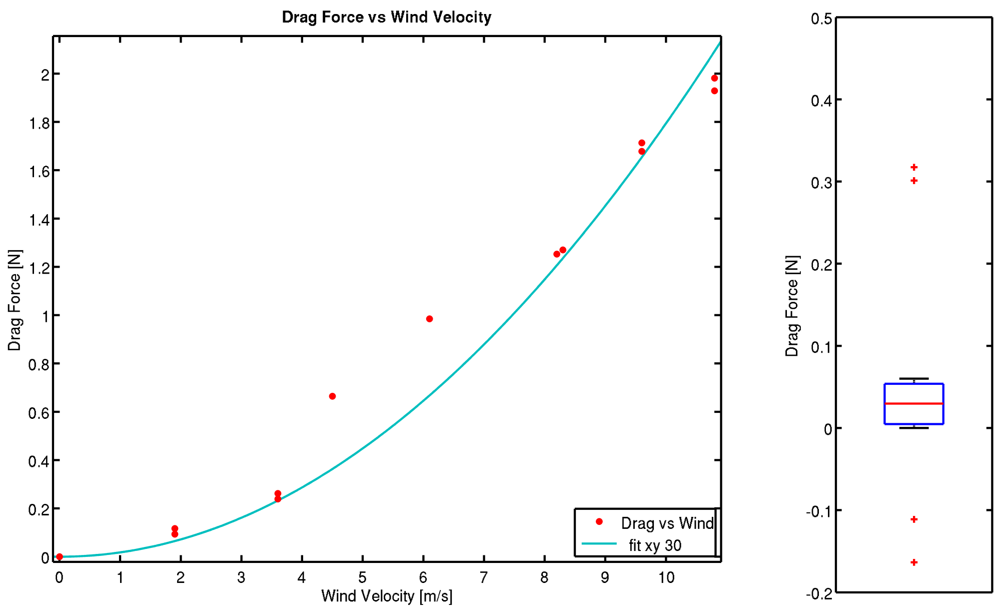
caused the onboard stabilization program to react to conceived changes and steer against the fixing mechanism, leading to some shaking. This shaking was transferred to the scale and could have disturbed the readings somewhat.

From the boxplot analysis (Figure 4.9(b)), it is even clearer that the distribution of residuals is skewed to the positive, but in a very narrow segment above the expectation, thanks to the small kurtosis. Together with an RMSE of $\epsilon_{RMSE} = 0.1164N$, this signifies that the fitted function approximates the actual proportionality relationship well. The calculated coefficient for this angle is

$$\rho_{xy,0^\circ} = 0.044506\frac{kg}{m}.$$

The experiment results for 30° (Figure 4.10) are chosen as representative for the remaining angles of the $\boldsymbol{x}$-$\boldsymbol{y}$-plane. They follow the trend of the first experiment. The fitted model captures the proportionality of the force measurements to the wind velocities, but a slight positive skewness of the samples is evident. The boxplot analysis shows a very compact peakedness for 30°. This results from the fact that at 30° several outliers were classified, being too far from the expectation value.

For the results of the other experiments, not enough measurements were taken to exclude possible outliers, therefore their specific kurtosis are larger, signifying a widely spread distribution of samples around the expectations. This problem could be addressed and the fit



(a) 30° $\boldsymbol{x}$-$\boldsymbol{y}$-plane (b) Residuals

Figure 4.10: Drag coefficient experiment results for 30° $\boldsymbol{x}$-$\boldsymbol{y}$-plane.

improved by taking more measurements. The mentioned outliers could be caused by measurement errors, possibly resulting from interferences by the stabilization software. Thus the model captures the proportionality relationship reasonably well, reflected in an RSME of $\epsilon_{RMSE,30^\circ} = 0.1421N$. The calculated coefficient for Equation (4.8) is given as

$$\rho_{xy,30^\circ} = 0.03588 \frac{kg}{m}.$$

Another trend to notice from this experiment is the decrease of the average and maximum induced drag force. This is an effect of the angle at which the wind blows at the UAV. We keep this in mind and postpone its detailed discussion until all experiments have been presented.

Figure 4.11 summarizes all fits of experiments concerning the $\boldsymbol{x}$-$\boldsymbol{y}$-plane.

For the investigation of the air drag properties of the AR-100 along its $\boldsymbol{z}$-axis, five different angles for the wind to hit the UAV were examined. The angles were 180°, 100°,110°, 120° and 0°, as illustrated in Figure 4.7.

Figure 4.12 presents the results of the first experiment for the $\boldsymbol{z}$-axis, at 180°. It is chosen as a representative for the angles 100°, 110° and 120°. Here the wind blows directly from the top of the AR-100, as seen from the UAV, in direction of the $\boldsymbol{z}$-axis. The samples appear well distributed around the expected values. The fitted model seems to capture the

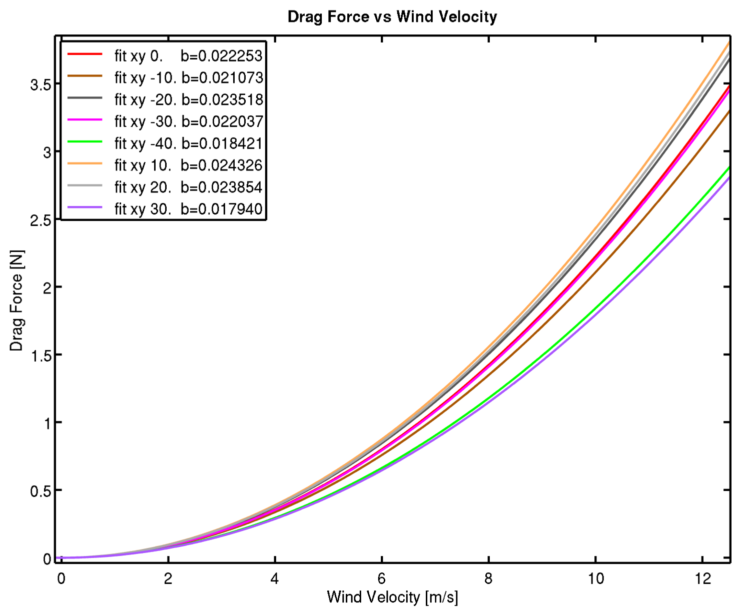
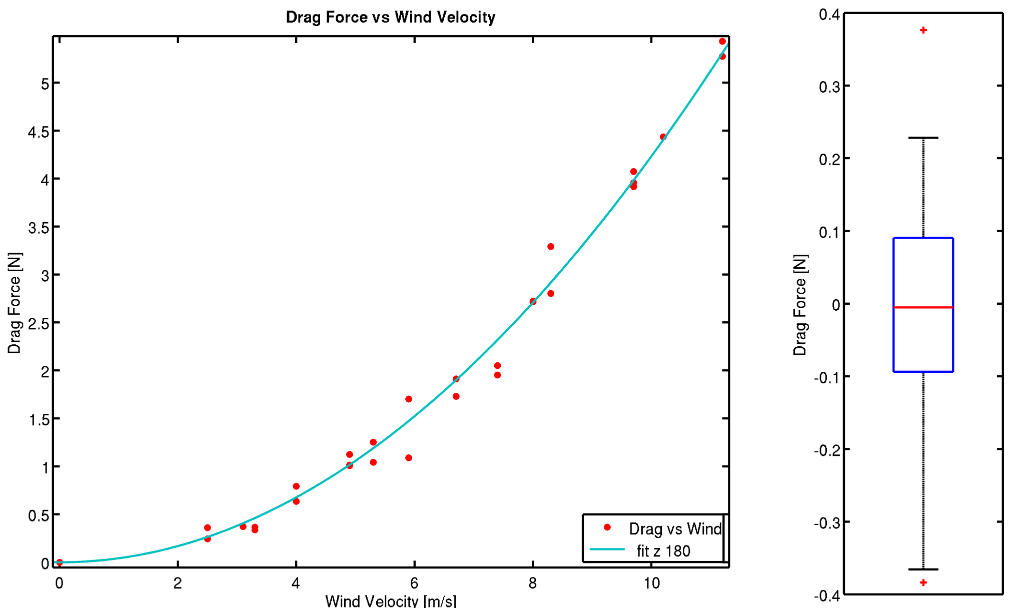


Figure 4.11: All fits of $\boldsymbol{x}$-$\boldsymbol{y}$-plane drag experiments.



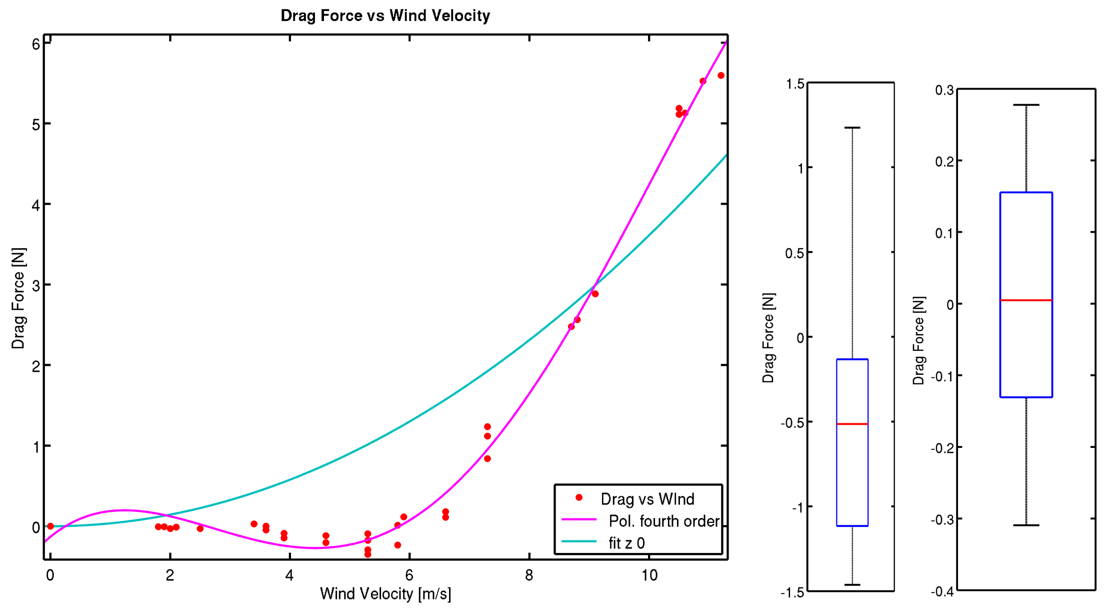
(a) 180° $\boldsymbol{z}$-axis (b) Residuals

Figure 4.12: Drag coefficient exp. results for 180° $\boldsymbol{z}$-axis.

relationship between wind velocity and induced drag force accurately. The boxplot analysis of the residuals (Figure 4.12(b)) shows a skewless distribution, since both the median and the box are positioned at zero. Additionally, the small box size reflects that the samples are densely distributed around the fitted model, validating the suggested relationship model. This band spreads to $\pm 0.1N$ around the proportionality function. The RMSE is given as $\epsilon_{RMSE,z180^\circ} = 0.1576N$. This denotes a good accuracy, considering the induced forces range higher than in the previous experiments. The air drag coefficient was calculated as

$$\rho_{z,180^\circ} = 0.084666\frac{kg}{m}.$$

It should be noted, that the air drag coefficients for the represented $\boldsymbol{z}$-axis experiments in the appendix are smaller. This effect is proposed to result from the angle at which the wind hits the UAV, but its discussion is postponed again.



(a) 0° $\boldsymbol{z}$-axis (b) bw^2 (c) ax^4+bx^3+cx+d

Figure 4.13: Drag coefficient experiment at 0° along the $\boldsymbol{z}$-axis.
Wind comes from below the AR-100, pushing against the wind created by the drone. Complex turbulences are created. The air drag behavior reflects this complexity, appearing more like a polynomial of fourth order than the suggested parabola.

In the last experiment concerning the air drag of the AR-100, the wind came from a 0° angle, i.e. directly below the drone, as seen from the drone. The wind pushes against the wind generated by the rotors. In Figure 4.13 we can see that this results in complex behavior. The drag forces rise slightly at first, then drop below the zero mark after which they rise again to magnitudes closer to the previous experiments for the specific wind velocities. The model from Equation (4.8) does not suffice anymore to describe the prevailing characteristics. The same figure shows a polynomial of fourth order approximating the generated drag forces. Comparing the boxplot analyses, the insufficient fit of the original model is more obvious. It shows a strong skewness to the negative and a large spread of the distribution. The different scales are important to note. On the other hand, the fourth order polynomial residuals display no skewness at all. And the kurtosis shows a very peaked sample distribution around the expected values. Furthermore, the RMSEs for both models are

$$\epsilon_{RMSE,z0^\circ} = 0.9268N,$$
$$\epsilon_{RMSE,z0^\circ,4th} = 0.1711N.$$

The error of the 4th order polynomial is more in line with the errors seen in the previous experiments, suggesting that this model approximates the actual relationship more accurately. To compare with the results of the previous experiments, the calculated air drag coefficient of the second order model is given as

$$\rho_{z,0^\circ} = 0.072244\frac{kg}{m}.$$

To summarize all $\boldsymbol{z}$-axis fits, they are presented together in Figure 4.14.

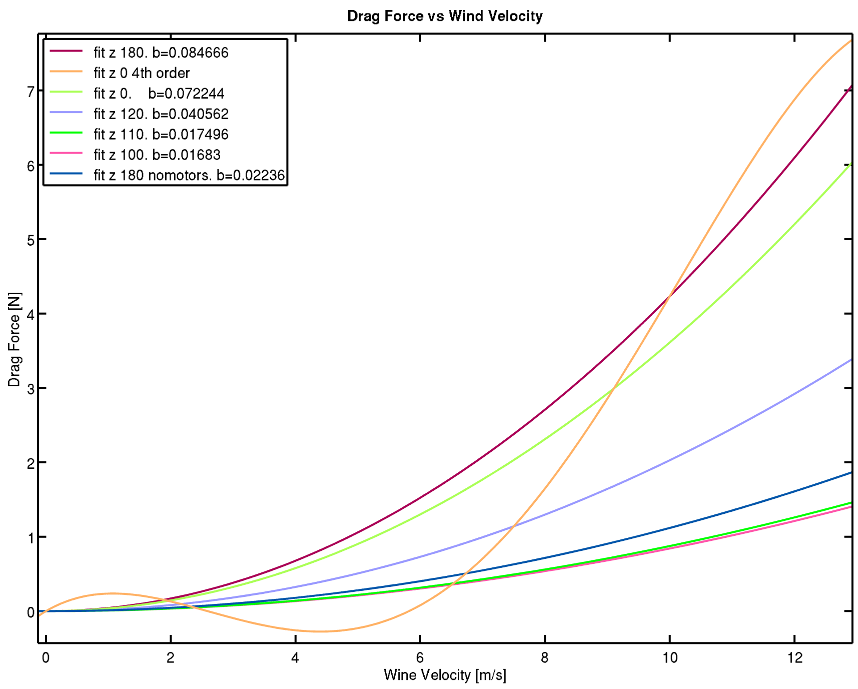


Figure 4.14: All fits of $\boldsymbol{z}$-axis drag coefficient experiment.
The coefficients differ substantially. This depends on the angle at which the wind hits the UAV.

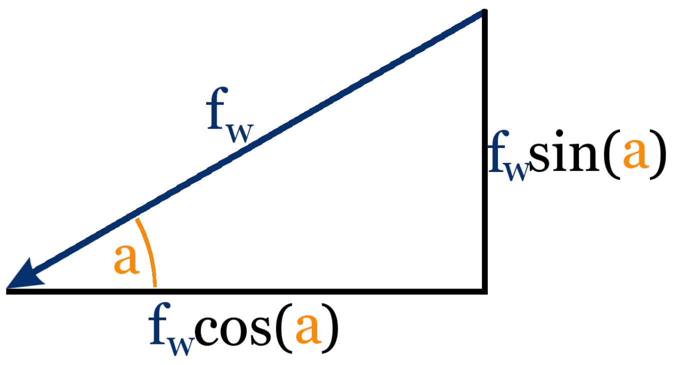


Figure 4.15: Effect of wind angle on drag force.
$\mathbf{f}_w$ is the wind force, **a** is the angle of attack, 30° in this example. The horizontal component is parallel to the swing direction.

Effect of Angle of Attack on induced drag force Comparing the results of the experiments, it becomes clear that the angle of the wind affects the induced drag force. This is because we measure only one dimension of the drag force. This dimension is the one parallel to the swing direction of the pendulum. This means that only the component of the force generated by the wind parallel to the swing direction, $\mathbf{f}_{||}$, is measured. The parallel component of the wind generated force depends on the angle, α, in the following way, as illustrated by Figure 4.15,

$$\begin{aligned}\mathbf{f}_{||} &= \mathbf{f}_w \cos(\alpha) \\ &= \cos(\alpha)\, \boldsymbol{\rho} \left\|\mathbf{v}_w\right\| \mathbf{v}_w,\end{aligned} \tag{4.11}$$

where $\mathbf{v}_w$ is the velocity of the wind. Consequently, we have measured the force component $\mathbf{f}_{||}$ but have assumed the model of equation (4.8), given as

$$\mathbf{f} = \boldsymbol{\rho}_d \left\|\mathbf{v}_w\right\| \mathbf{v}_w.$$

From this follows that the air drag coefficients obtained through the fitted models, ρ_d, are distorted, if the wind was coming from a different direction than exactly head on at 0°. The distorsion can be given as

$$\rho_d = \cos(\alpha)\, \rho.$$

Therefore the correct air drag coefficients can be calculated with

$$\rho = \frac{\rho_d}{\cos(\alpha)}.$$

Table 4.1 lists the distorted and adjusted or corrected air drag coefficients side by side. The calculated coefficients for the $\boldsymbol{x}$-$\boldsymbol{y}$-plane are now very close to each other. The coefficients for the $\boldsymbol{z}$-axis are also somewhat similar. Figure 4.16 shows the air drag expectations with corrected coefficients.

Experiment	Coeff. ρ_d in $\left[\frac{kg}{m}\right]$	Adj. Coeff. $\rho = \frac{\rho_d}{\cos\alpha}$ in $\left[\frac{kg}{m}\right]$
$\boldsymbol{x}$-$\boldsymbol{y}$ $0°$	0.044506	0.044506
$\boldsymbol{x}$-$\boldsymbol{y}$ $10°$	0.048652	0.0494025355215
$\boldsymbol{x}$-$\boldsymbol{y}$ $20°$	0.047708	0.0507697931693
$\boldsymbol{x}$-$\boldsymbol{y}$ $30°$	0.03588	0.041430655317
$\boldsymbol{x}$-$\boldsymbol{y}$ $-10°$	0.042146	0.0427961699845
$\boldsymbol{x}$-$\boldsymbol{y}$ $-20°$	0.047036	0.0500546657062
$\boldsymbol{x}$-$\boldsymbol{y}$ $-30°$	0.044074	0.0508922715285
$\boldsymbol{x}$-$\boldsymbol{y}$ $-40°$	0.036842	0.0480938153536
$\boldsymbol{z}$-axis $180°$, no motors	0.02236	0.02236
$\boldsymbol{z}$-axis $180°$	0.084666	0.084666
$\boldsymbol{z}$-axis $120°$	0.040562	0.081124
$\boldsymbol{z}$-axis $110°$	0.02001	0.0585053260473
$\boldsymbol{z}$-axis $100°$	0.01683	0.0969201072313
$\boldsymbol{z}$-axis $0°$	0.072244	0.072244

Table 4.1: The air drag proportionality coefficients.
The adjusted values are corrected with respect to the angle of the wind.

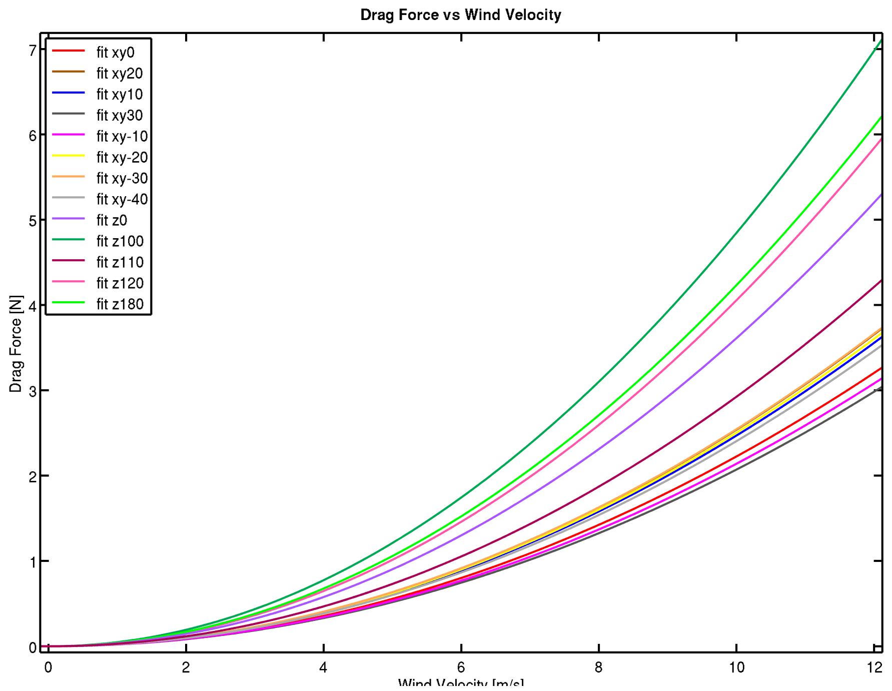


Figure 4.16: All adjusted drag coefficient models.

Conclusion

The results of the conducted experiments can be combined to calculate a final air drag coefficient necessary for the process model. A sensible choice for the coefficient can be based on the information in Table 4.1 and Figure 4.16. It is clear to see that the drag coefficients for the $\boldsymbol{x}$-$\boldsymbol{y}$ plane are significantly smaller than the drag coefficients for the $\boldsymbol{z}$-axis.

The coefficient of the 180° experiment without motors is not included in the calculation of the final drag coefficient, because the experiments confirmed that the air drag characteristics differ significantly between stopped and running rotors. Furthermore, the drag coefficient of the 0° experiment is not included, because the results showed that the proposed model does not describe the complex characteristics sufficiently well. Consequently, a three-dimensional vector containing the arithmetic means of the corresponding coefficients represents an acceptable compromise to describe the air drag characteristics of the AR-100. That vector is given as

$$\boldsymbol{\rho} = \begin{pmatrix} 0.0472432383226 \\ 0.0472432383226 \\ 0.0803038583196 \end{pmatrix} \frac{kg}{m}. \tag{4.12}$$

Future research can improve on the determined air drag coefficients by conducting the experiments within a wind channel specific to such purposes. A more sophisticated measuring platform can deliver more accurate results than the pendulum and scale combination. The used pendulum can introduce errors based on how well its axis rotates. Furthermore, its own weight and air drag can affect the measurements.

4.3 State Estimation

The performance of the developed algorithm in combination with the chosen sensors is investigated in this section. The continuous-discrete square root unscented Kalman filter (Algorithm C.6) with Sequential Updates was implemented in C++. For the QR-decomposition and Cholesky update, *Octave 3.2.3*[3] is employed. For the inversion of matrices the numerically stable and efficient backward-substitution method is used. The square roots of the initial covariance matrix and the model noise covariance matrices are calculated via Cholesky decomposition.

4.3.1 Simulation

For a test of the model and of the estimator algorithm, simulated measurements are generated and delivered to the estimator.

Experimentation Setup

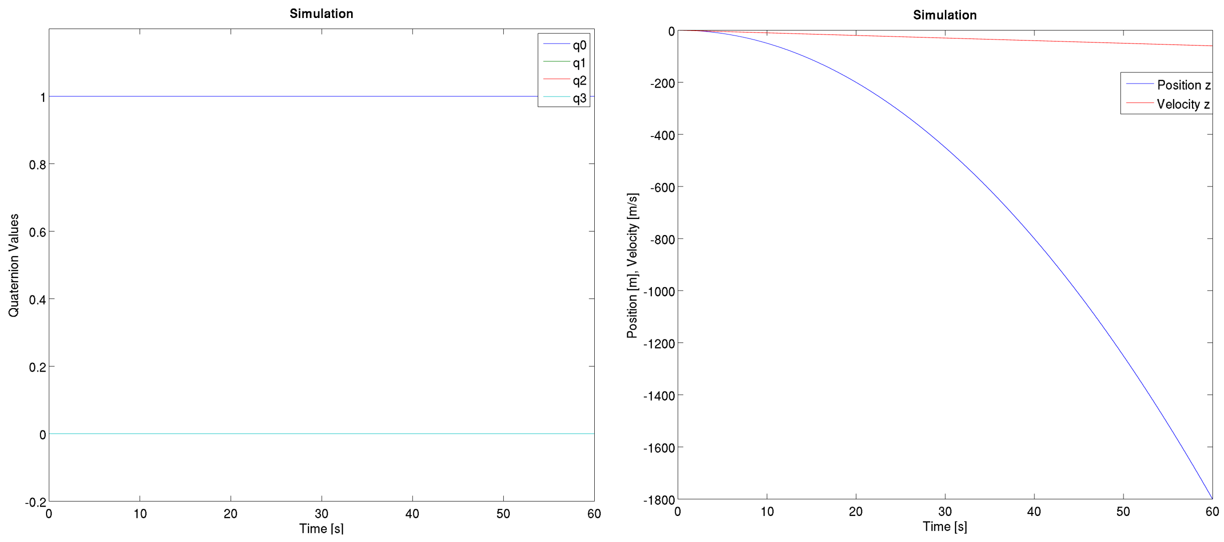
The initial system state is zeroed, only the quaternion's real value is one. The supplied measurements are IMU readings of a linear acceleration of $8.81\frac{m}{s^2}$ in $\boldsymbol{z}$-axis direction. This is an upward acceleration of $1\frac{m}{s^2}$. The second type of supplied measurements are tachometer measurements of 1872.772RPM for each motor. And the last measurement

[3]`www.octave.org` in conjunction with its *qrupdate* library version 1.0.1.

type is the quaternion pseudo measurement. The measurements are delivered at a rate of 20Hz for 60 seconds.

Results and Discussion

Figure 4.17 shows the results of the simulation experiment. The quaternion re-



(a) Quaternion Values (b) Position and Linear Velocity

Figure 4.17: State Estimation upward simulation experiment results.

mains at its initial values, correctly since all angular velocities are zero. All the parameters not shown are zero throughout the whole simulation. The linear velocity in z-axis grows at the intended $1\frac{m}{s^2}$, to reach $60\frac{m}{s}$ after 60 seconds. The estimated position is -1800m at 60 seconds in NED-frame, which corresponds to the calculated true position,

$$p = \frac{1}{2}a \cdot t^2 = -\frac{1}{2} \cdot 1 \cdot 60^2 m = -1800m.$$

The model and the estimation algorithm function correctly.

A second simulation (Figure 4.18) for a rotation around the z-axis at an angular velocity of $1\frac{\circ}{s}$. The linear acceleration the IMU reports is $9.81\frac{m}{s^2}$ in z-axis direction, i.e. gravity. The quaternion values can be converted to Euler angles. The final yaw angle is obtained as 59.9996°, compared to an actual 60°. This error of $4 \cdot 10^{-4\circ}$ is reasonably small to say that the model represents the actual dynamics correctly. The error could also come from precision errors in the conversion calculation from the small quaternion values to degrees.

Conclusion

In this section, the performance of the state estimation and the model were examined. From the results of these simulation experiments, it can be deduced that the

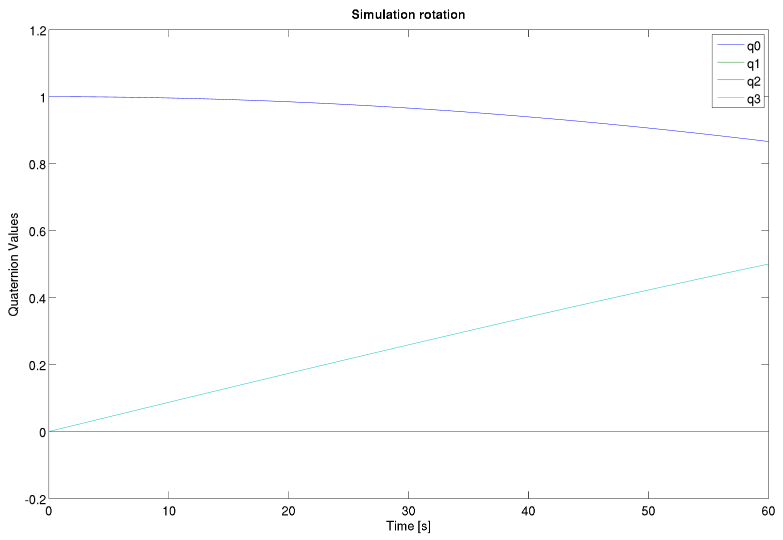


Figure 4.18: Yaw rotation simulation experiment results. Quaternion values are displayed.

model and the state estimation algorithm function correctly. With this knowledge, real world experiments can be run. More in-depth simulation experiments can include whole flight courses within a simulation environment. This environment can then also implement random noise characteristics and support more sensor types, simulating the real world circumstances concerning quadrocopter flight.

4.3.2 Yaw Rotation

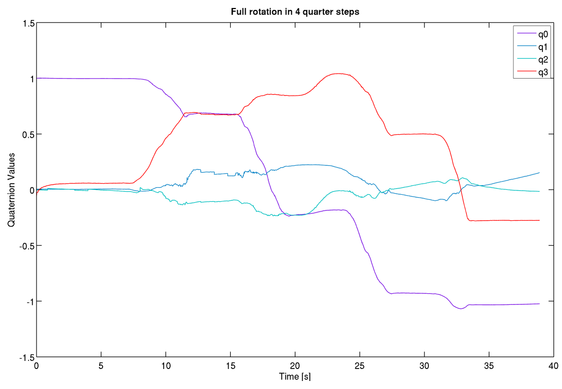
The first real world experiment is a yaw rotation to investigate the estimator's performance. The AR-100 is flown in a pure full rotation around the $\boldsymbol{z}$-axis in 4 steps of about 90°. All available sensors are employed and integrated in the correction step.

Experimentation Setup

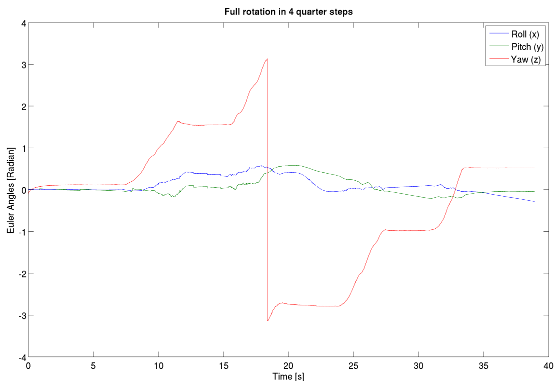
The system state is initialized by the initialization procedure (Section 3.4.4). In order to assist the egomotion estimator, the UAV is positioned above and surrounded by a contrast-rich area, i.e. the floor is white with equidistant black dots with a diameter of about 2cm. The kanatani egomotion estimator delivers its measurements at about 1Hz.

Results and Discussion

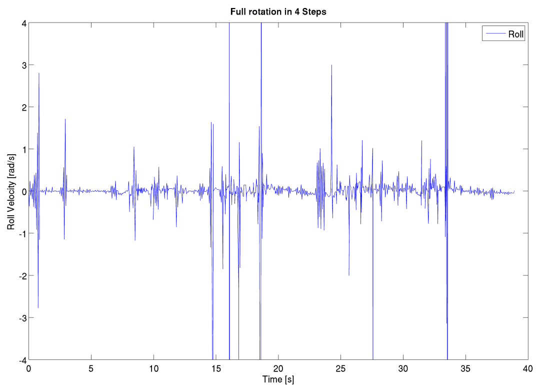
The results of the rotation in four steps experiment are presented in Figure 4.19. Looking at the quaternion values and their converted Euler angle counterparts, (Figures 4.19(a) & (b)), the four steps of the rotation are clearly visible. As can be seen from the initial attitude, the UAV was positioned almost aligned with the world NED axes. The $\boldsymbol{x}$-axis was slightly off North, visible in the small positive yaw value of the third complex



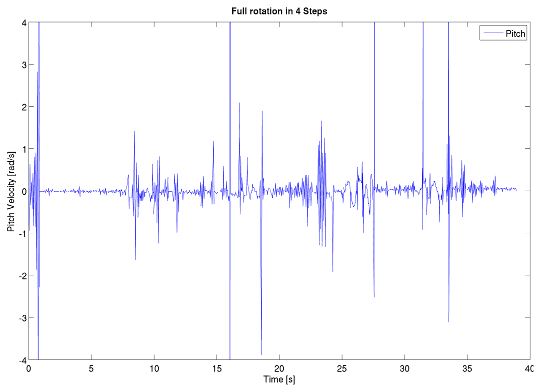
(a) Quaternion Values



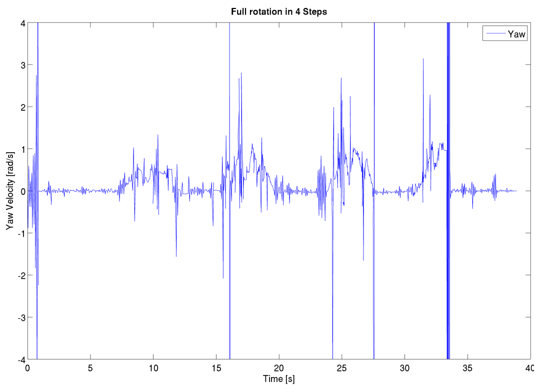
(b) Euler Angles



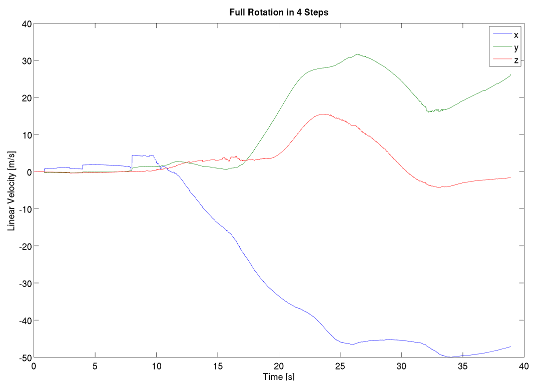
(c) Angular Velocity Roll



(d) Angular Velocity Pitch



(e) Angular Velocity Yaw



(f) Linear Velocity

Figure 4.19: State Estimation rotation in 4 steps experiment results.

quaternion component. The UAV is rotating in positive direction around the z-axis whenever the lines have a slope different from zero. From the quaternion values, it would seem that around 22 seconds, another rotation step was executed. Looking at the Euler angles at the same time instant, no rotation is visible. This also rules out a correction by a magnetometer measurement, since in this case a rotation would also be visible in the Euler angle representation. Thus this bump in the quaternion values is the effect of the pseudo quaternion measurement to keep the quaternion's unity requirement in check.

The Euler angles show slight offsets from the actual 90° steps after each step. While this could be an estimation error, another possible explanation exists. During in-flight rotations, the rotation angle is difficult to hit exactly, since the UAV is controlled manually. Thus, small over- or undershoots are possible and visible in the angles. Upon landing, the total excess rotation angle can be measured at 30°. This corresponds within measuring precision to the final estimated yaw angle of 29.74°. The final roll and pitch angle estimations are -8° and -2.6°, respectively. This represents an error at the final time instant of 4.4% for roll and 1.4% for pitch angle estimation, considering that an error of 100% would represent the opposite direction.

Figures 4.19(c) to (e) display the angular velocities, which are the IMU driven measurements. The sudden high peaks are measurement errors, but the estimation algorithm retains a stable system state estimate. During yaw rotation, a certain noisyness in the roll and pitch velocities is captured in these measurements.

Finally, the estimated linear velocities are presented (Figure 4.19(f)). While they should be close to zero, the estimates for the $\boldsymbol{x}$- and $\boldsymbol{y}$-directions show large errors with a tendency to grow larger. The $\boldsymbol{z}$-axis velocity estimate catches itself after a large error. The inability of this Kalman filter to estimate correct linear velocities is a consequence of the fact, that no measurements of the absolute velocity or an indirect effect thereof are employed in the correction process. The only measurement used to correct the linear velocity and the IMU acceleration biases is the egomotion estimation received from the optical sensor. It is important to note that the linear component of that estimate is a normalized vector, since only relative translative velocity can be calculated by the egomotion estimation algorithm. This unit vector is obviously not sufficient for a satisfactory correction of the linear velocity estimates. Consequently, the location estimates are erroneous as well.

4.3.3 Pitch Rotation

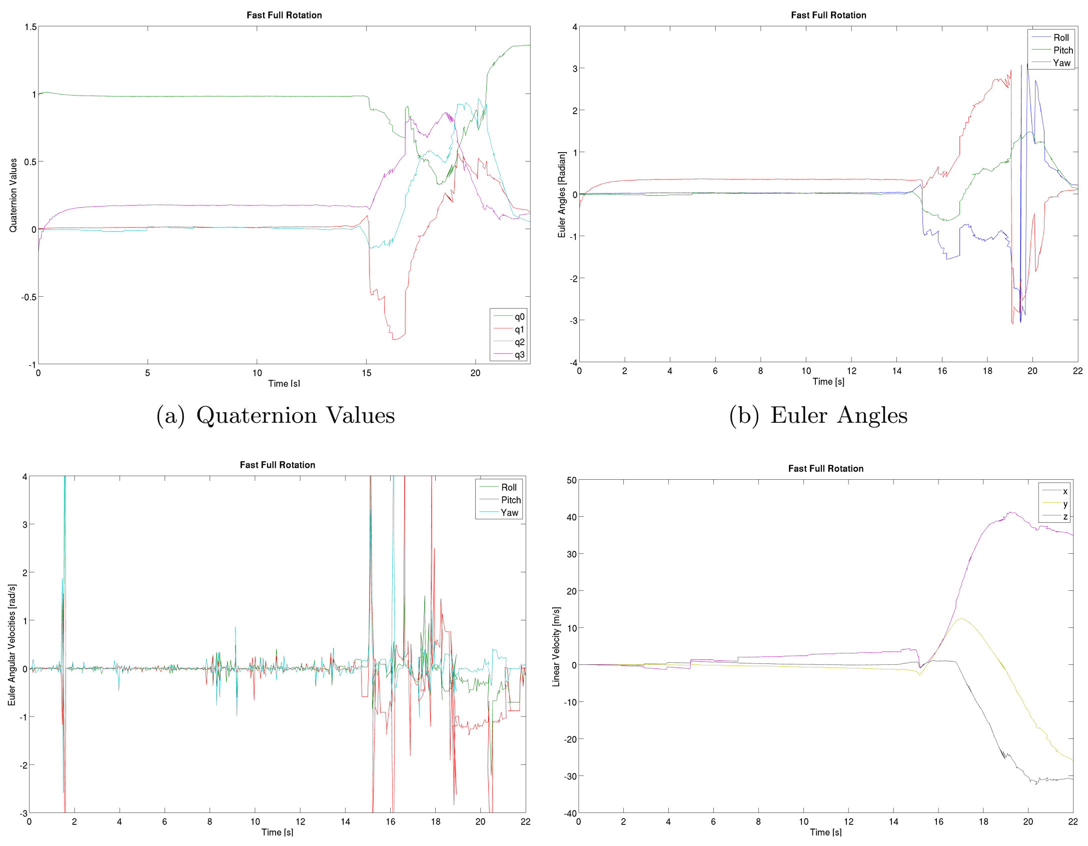
The second experiment investigates a forced and controlled rotation about the $\boldsymbol{y}$-axis.

Experimentation Setup

The experiment setup is almost identical to the previous, with the exception that the UAV is rotated around its $\boldsymbol{y}$-axis manually, not by its rotors. The experimentator secures the AR-100 and turns it one full rotation. This motion would lead to a crash of the quadrocopter, but the desired data to observe is only the performance of the state estimator, not the viability of the motion during actual flight missions.

Results and Discussion

The results of the pitch rotation experiment are presented in Figure 4.20. The



(a) Quaternion Values (b) Euler Angles

(c) Angular Velocity (d) Linear Velocity

Figure 4.20: State Estimation pitch rotation experiment results.

quaternion values during the rotation seem rather wild, but less smooth results were to be expected, since the pitch rotation by hand is not very pure and introduces noise rotations about the other axes. Another noise factor of the hand-fixed rotation of the UAV is that the motion is not very constant or steady, as at one point the experimentator needs to change his hold on the UAV. Nevertheless, after the rotation, the quaternion values reflect a good approximation of the result of a 360° rotation: the initial configuration. The final q_0 value is a little higher than the initial value, but looking at the Euler angles in Figure 4.20(b), it can be seen that they are almost at their initial values. Thus the quaternion could be slightly out of unity, which would be corrected in the near future after the rotation.

The angular velocities reflect the moment of the switching of the hand-hold clearly. The pitch angular velocity drops during the rotation, even turns a bit backward, before continuing the begun pitch rotation until one full rotation is finished. The linear velocity in Figure

4.20(d) shows again the inability of this combination of sensors to contain the bias of the linear acceleration sensors of the IMU. The linear velocity grows, but stops and stays at its value once the robot stops moving.

4.3.4 Translation

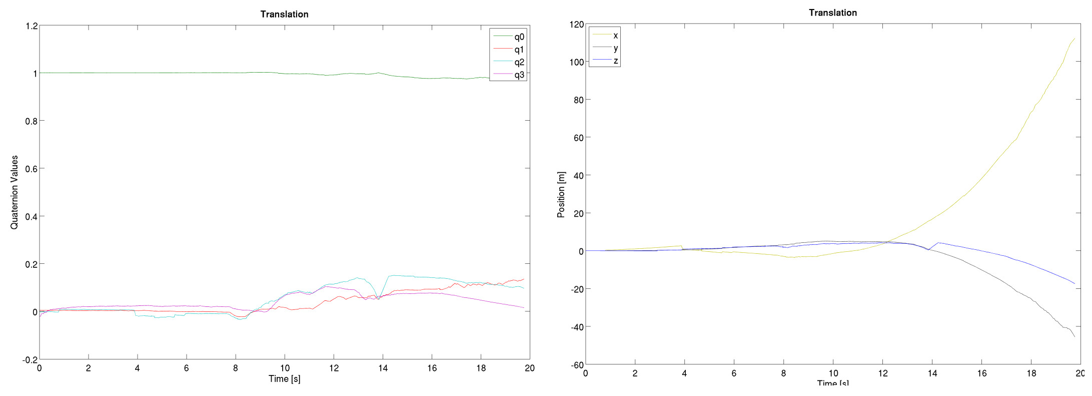
The last experiment examines the state estimation for a translational motion.

Experimentation Setup

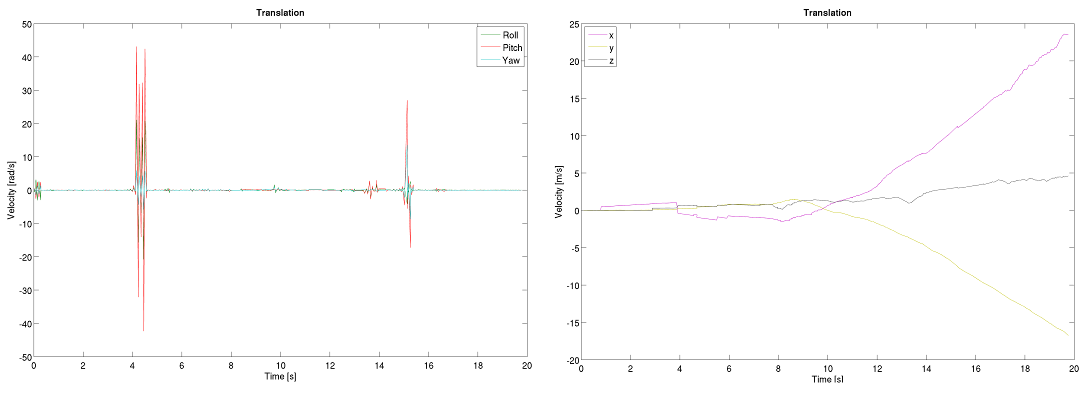
In this experiment, the drone starts again within the contrast-rich environment. Instead of flying a rotation as in the precious experiments, the UAV flies straight forward along the $\boldsymbol{x}$-axis direction without rotating. It flies a straight distance of 2m, where it lands. All available sensors, as defined in Section 3.4.3, are employed in the correction process.

Results and Discussion

Figure 4.21 summarizes the results of the translation experiment. Similar to the



(a) Quaternion Values (b) Position



(c) Angular Velocity (d) Linear Velocity

Figure 4.21: State Estimation translation experiment results.

previous experiments, the results show a good preformance of the state estimation for the attitude of the AR-100. The quaternion values stay close to constant during the rotationless forward-motion. The angular velocities are also kept at zero, with measurement error spikes at the start and end of the translation.

The linear values fare worse than the angular. The linear velocities grow in a linear way, indicating the IMU bias to be responsible, since that is integrated to obtain the linear velocity. From the too high linear velocity follows the fast increase in the position vector. But if the optical sensor measurement would have absolutely no effect on the linear acceleration during the correction step, then, if the biases are all about the same magnitude, the velocities would grow at the same rates. That is not the case, here, where the $\boldsymbol{x}$-axis component grows far more rapidly. The reason for this can lie in two explanations. Either the bias of the $\boldsymbol{x}$-axis acceleration component is bigger than the $\boldsymbol{y}$- and $\boldsymbol{z}$-axis counterparts, or the optical sensor measurement, reporting a definitive translational direction, affects the linear velocity in such a way that it supports its main direction and suppresses the other directions in accordance with their relative magnitudes. Unfortunately, a definitive answer to that question cannot be deduced from the compiled data.

4.3.5 Conclusion

This section presents and discusses the results of the state estimation experiments. Different motions were examined, to investigate the performance of the state estimator on angular and linear components separately. The results show successful estimation of the angular components. The linear components are not estimated correctly, as the combination of sensor cooperating in the correction process do not offer absolute values for effeects of the linear state components. The barometer altitude sensor of the AR-100 seems too noisy to be of satisfying use in the correction.

From these results, the conclusion can be drawn that the anguler velocity meaasurements from the optical egomotion estimation combined with magnetometer measurements suffice for a reasonable estimation of the angular velocity and attitude of the UAV. However, the linear direction information of the optical flow egomotion analysis is not sufficient for absolute estimation of the linear velocity and position of the UAV. The effect on the direction of the velocity estimate begs further investigation.

Recommended future work addressing this issue could fly in one direction yet have the camera grab pictures of motion into a different direction, either by mounting teh camera differently on the drone or by flying infront of a large screen. A large projection on the screen could then be the optical input to the drone's camera sensor. In this way, the effect of different directed optical measurements on the state estimate can be examined.

Chapter 5

Conclusions and Future Work

This thesis discusses the development of a dead reckoning navigation system applying a recursive Bayesian estimation framework. The specific state estimation algorithm developed for this system is the continuous-discrete square root Unscented Kalman filter with sequential updates. This algorithm is derived from multiple modifications of the original Kalman filter that have already proven themselves separately. This fusion of algorithms was implemented for analysis of the estimation performance. The results show a good estimation of the angular navigational components, but the linear components could not be estimated to a satisfactory degree. The cause for that inability is the absence of good absolute measurements of indirect effects of the linear components. An effect of the linear directional information contained within the visual egomotion estimation is possible, but the obtained results do not allow a definitive answer. Further research is required, focusing on the effect of different direction measurements relative to the actual motion. Future work can either add new sensors delivering the required measurements or incorporate the implemented GPS sensor on the AR-100. This addition is easily accomplished due to the sequential updates modification of the estimation algorithm.

Mathematical models describing the physical characteristics of the unmanned aerial vehicle AR-100 and the mappings to its sensor measurements are developed and discussed in detail. The parameters of these dynamic nonlinear models are determined in experiments and shown to support the expected relationship models with good accuracy. In the context of the air drag experiments, complex behavior was detected when wind was blowing at the drone from below, in opposite direction to the wind pushed by the rotors. Though a fourth order polynomial could describe the characteristics with the same accuracy as quadratic equations describe the air drag characteristics of other directions, this relationship warrants further investigation, optimally employing a wind channel for researching the quadrocopter flight dynamics.

Future work can address the shortcomings of the conducted experiments. Because of lack of the proper equipment, no ground truth could be obtained during flight. For having a ground truth to compare the estimated state information to in order to evaluate the quality of the models and estimation algorithms more accurately, experiments can be conducted with the help of tracking devices. Infrared cameras can track the location and orientation of the drone in six degrees of freedom, providing the necessary base truth.

With the base truth established, the performance of the filter framework can be investigated for more complex flight maneuvers. The effect of flight in more difficult terrain for the sensors to measure can be examined as well, to determine the limits of the system.

Furthermore, the effects of the different sensor measurements on state estimation through the correction step can be investigated. Different combinations of sensor fusions can be evaluated and a conclusion can be drawn, as to which sensors work better in which combination.

Bibliography

[1] Forlizzi, Jodi ; DiSalvo, Carl F.: Service robots in the domestic environment: a study of the roomba vacuum in the home. In: Goodrich, Michael A. (Hrsg.) ; Schultz, Alan C. (Hrsg.) ; Bruemmer, David J. (Hrsg.): *HRI*, ACM, 2006. – ISBN 1–59593–294–1, 258-265 1

[2] Cardoso, Jorge ; Ferreira, Manuel ; Santos, Cristina: LegOSC: Mindstorms NXT robotics programming for artists. In: Filipe, Joaquim (Hrsg.) ; Cetto, Juan A. (Hrsg.) ; Ferrier, Jean-Louis (Hrsg.): *Proceedings of the Fifth International Conference on Informatics* Bd. **RA-1**. Funchal, Madeira, Portugal, Mai 2008, 177-182 1

[3] Singer, P. W.: *Wired for War*. Penguin Press HC, The, 2009 1

[4] Bodner, J. ; Wykypiel, H. ; Wetscher, G. ; Schmid, T.: First experiences with the da Vinci operating robot in thoracic surgery. In: *European Journal of Cardio-Thoracic Surgery* **25** (2004), Nr. 5, S. 844 1

[5] Hagn, Ulrich ; Nickl, Matthias ; Jörg, Stephan ; Tobergte, Andreas ; Kübler, Bernhard ; Passig, Georg ; Grüger, Martin ; Fröhlich, Florian ; Seibold, Ulrich ; Konietschke, Rainer ; Le-Tien, Luc ; Albu-Schäffer, Alin ; Grebenstein, Markus ; Ortmaier, Tobias ; Hirzinger, Gerd: DLR MiroSurge – towards versatility in surgical robotics. In: *7. Jahrestagung der Deutschen Gesellschaft für Computer- und Roboterassistierte Chirurgie e.V.*, 2008. – ISBN 978–3–00–025798–8, 143 - 146 1

[6] Kang, Sang-Wook ; Jeong, Jong J. ; Nam, Kee-Hyun ; Chang, Hang S. ; Chung, Woong Y. ; Park, Cheong S.: Robot-Assisted Endoscopic Thyroidectomy for Thyroid Malignancies Using a Gasless Transaxillary Approach. In: *Journal of the American College of Surgeons* 209 (2009), August, Nr. 2, e1–e7. `http://dx.doi.org/10.1016/j.jamcollsurg.2009.05.003`. – DOI 10.1016/j.jamcollsurg.2009.05.003. – ISSN 10727515 1

[7] Withrow, Thomas J. ; Shen, Xiangrong ; Mitchell, Jason E. ; Goldfarb, Michael: A forearm actuation unit for an upper extremity prosthesis. In: *ICRA*, IEEE, 2008, 2459-2464 1

[8] Hill, Ciaran ; Amodeo, Antonio ; Joseph, Jean V. ; Patel, Hitendra R.: Nano- and microrobotics: how far is the reality? In: *Expert Review of Anticancer Therapy* **8** (2008), Nr. 12, S. 1891–1897 1

[9] Estlin, Tara A. ; Gaines, Daniel M. ; Chouinard, Caroline ; CastaÃśo, Rebecca ; Bornstein, Benjamin ; Judd, Michele ; Nesnas, Issa A. D. ; Anderson, Robert C.: Increased Mars Rover Autonomy using AI Planning, Scheduling and Execution. In: *ICRA*, IEEE, 2007, 4911-4918 1

[10] Squyres, Steve: *Roving Mars: Spirit, Opportunity, and the Exploration of the Red Planet*. New York, NY, USA : Hyperion, 2005 1

[11] Upcroft, Ben ; Moser, Michael ; Makarenko, Alex ; Johnson, David ; Donikian, Ashod ; Alempijevic, Alen ; Fitch, Robert ; Uther, Will ; GrÃŷtli, Esten I. ; Biermeyer, Jan ; Gonzalez, Humberto ; Templeton, Todd ; srini, Vason P. ; Sprinkle, Jonathan: DARPA Urban Challenge Technical Paper: Sydney-Berkeley Driving Team / University of Sydney; University of Technology, Sydney; University of California, Berkeley. Version: June 2007. `http://chess.eecs.berkeley.edu/pubs/379.html`. 2007. – Forschungsbericht 1

[12] Van Der Merwe, Rudolph: *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*, Diss., 2004. `http://portal.acm.org/citation.cfm?id=1037398` 10, 16, 19, 22, 28, 29, 40, 79, 84, 126, 128

[13] Welch, Greg ; Bishop, Gary: An Introduction to the Kalman Filter. Version: 1995. `http://portal.acm.org/citation.cfm?id=897831`. Chapel Hill, NC, USA : University of North Carolina at Chapel Hill, 1995. – Forschungsbericht 11, 16

[14] Weisstein, Eric W.: *Delta Function*. `http://mathworld.wolfram.com/DeltaFunction.html` 14

[15] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems. In: *Transactions of the ASME - Journal of Basic Engineering* (1960), Nr. 82 (Series D), 35–45. `http://www.cs.unc.edu/~welch/kalman/media/pdf/Kalman1960.pdf` 15, 17, 19

[16] Maybeck, P. S. ; Press, Academic (Hrsg.): *Stochastic models, estimation and control. Volume I*. 1979 `http://www.cs.unc.edu/~welch/media/pdf/maybeck_ch1.pdf` 18

[17] Raymond, Xavier S.: *Elementary Introduction To The Theory Of Pseudodifferential Operators*. CRC Press, 1991. – ISBN 0849371589 21

[18] Julier, S. ; Uhlmann, J.: *A General Method for Approximating Nonlinear Transformations of Probability Distributions*. `http://citeseer.ist.psu.edu/julier96general.html`. Version: 1996 23, 24

[19] Julier, Simon J. ; Uhlmann, Jeffrey K.: New extension of the Kalman filter to nonlinear systems. In: Kadar, Ivan (Hrsg.): *Signal Processing, Sensor Fusion, and Target Recognition VI* Bd. 3068, SPIE, 1997, 182–193 24

[20] Julier, Simon J.: The Scaled Unscented Transformation. In: *Proceedings of the 2002 American Control Conference* Bd. 6, 2002, 4555–4559 27, 28

[21] Bucy, Peter D. Richard S. ; Joseph J. Richard S. ; Joseph: *Filtering for stochastic processes with applications to guidance*. 2. ed., repr. Providence, RI : AMS Chelsea Publ., 2005. – ISBN 0–8218–3782–6 ; 978–0–8218–3782–5. – Includes bibliographical references and index. - Originally published: 2nd ed. New York, N.Y. : Chelsea Pub., c1987 30

[22] Jazwinski, Andrew H.: *Stochastic Processes and Filtering Theory*. Academic Press, 1970 `http://books.google.com/books?vid=ISBN0123815509`. – ISBN 0123815509 30, 83

[23] Särkkä, Simo: On Unscented Kalman Filtering for State Estimation of Continuous-Time Nonlinear Systems. In: *Automatic Control, IEEE Transactions on* 52 (2007), Nr. 9, 1631–1641. `http://dx.doi.org/10.1109/TAC.2007.904453`. – DOI 10.1109/TAC.2007.904453 30, 31, 32

[24] Kailath, T.: An innovations approach to least-squares estimation–Part I: Linear filtering in additive white noise. In: *Automatic Control, IEEE Transactions on* 13 (1968), Nr. 6, 646–655. `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1099025` 31

[25] Grewal, Mohinder S. ; Andrews, Angus P.: *Kalman Filtering: Theory and Practice Using MATLAB*. 3. Wiley-IEEE Press, 2008 `http://books.google.com/books?vid=ISBN0470173661`. – ISBN 0470173661 32

[26] Särkkä, Simo: *Recursive Bayesian Estimation on Stochastic Differential Equations*, Helsinki University of Technology, Diss., April 2006 33, 82

[27] Merwe, Rudolph van d. ; Wan, Eric A.: The square-root unscented Kalman filter for state and parameter-estimation. In: *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)* Bd. 6, 2001, 3461–3464 34

[28] Anderson, Brian D. O. ; Moore, John B.: *Optimal Filtering*. Englewood Cliffs, NJ : Prentice-Hall, 2005 `http://books.google.com/books?vid=ISBN0486439380`. – ISBN 0486439380 36

[29] Haykin, Simon: *Adaptive Filter Theory (4th Edition)*. 4. Prentice Hall, 2001 `http://books.google.com/books?vid=ISBN0130901261`. – ISBN 0130901261 41

[30] Featherstone, Roy: *Rigid Body Dynamics Algorithms*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2007. – ISBN 0387743146 47, 54, 62, 63, 78

[31] Siciliano, Bruno (Hrsg.) ; Khatib, Oussama (Hrsg.): *Springer Handbook of Robotics*. Berlin, Heidelberg : Springer, 2008 `http://dx.doi.org/10.1007/978-3-540-30301-5`. – ISBN 978–3–540–23957–4 52, 58

[32] Hoag, David: Apollo Guidance and Navigation: Considerations of Apollo IMU Gimbal Lock:E 1344. In: *Apollo Lunar Surface Journal* (1963), April. `http://www.hq.nasa.gov/alsj/e-1344.htm` 59

[33] BAHADIR, Deniz: *Verbesserte Eigenbewegungsschätzung für Flugroboter.* 2009 65

[34] GUENARD, Nicolas: *Optimisation et implémentation de lois de commande embarquées pour la téléopération de micro drones aériens X4-ïňĆyer*, Laboratoire Télérobotique et Cobotique - CEA, Diss., octobre 2007 74

[35] HAMEL, Tarek ; MAHONY, Robert ; LOZANO, Rogelio ; OSTROWSKI, James: Dynamic modelling and configuration stabilization for an X4-flyer. In: *15th Triennial World Congress, Barcelona, Spain* (2002) 74, 75

[36] PROUTY, Raymond W.: *Helicopter Performance, Stability, and Control.* Krieger Pub Co, 2001 `http://books.google.com/books?vid=ISBN1575242095`. – ISBN 1575242095 75, 103

[37] OH, Seung-Min: *Nonlinear Estimation for Vision-Based Air-to-Air Tracking*, Georgia Institute of Technology, Diss., Dezember 2007 81, 82, 84

[38] GELB, Arthur: *Applied Optimal Estimation.* The MIT Press, 1974 `http://books.google.de/books?id=KlFrn8lpPP0C` 81, 82

[39] PEEBLES, Peyton: *Probability, Random Variables, and Random Signal Principles.* McGraw Hill Higher Education, 2000 `http://books.google.com/books?vid=ISBN0071181814`. – ISBN 0071181814 127

Appendix A

EKF Inaccuracies

Let's examine the EKF's shortcomings in an example (from [12]): A scalar random variable x is propagated through a simple nonlinear function g,

$$y = g\left(x\right) = x^2,$$

with $x \sim \mathcal{N}\left(\bar{x}, \sigma_x^2\right)$ taken from a Gaussian distribution with mean $\bar{x}$ and covariance σ_x^2. First, the mean $\bar{y}$ and covariance σ_y^2 of y are calculated analytically, before calculating them in the way of the EKF as comparison. The mean is given as

$$\begin{aligned}
\bar{y} &= g\left(\bar{x}\right) + E\left[\left(x-\bar{x}\right)\frac{d\mathbf{g}\left(\bar{x}\right)}{d\mathbf{x}} + \frac{1}{2}\left(x-\bar{x}\right)^2\frac{d^2\mathbf{g}\left(\bar{x}\right)}{d^2\mathbf{x}} + \cdots\right] && \text{(A.1)}\\
&= \bar{x}^2 + E\left[2\left(x-\bar{x}\right)\bar{x} + \left(x-\bar{x}\right)^2\right] && \text{(A.2)}\\
&= \bar{x}^2 + E\left[2\bar{x}x - 2\bar{x}^2 + \left(x-\bar{x}\right)^2\right]\\
&= \bar{x}^2 + 2\bar{x}^2 - 2\bar{x}^2 + E\left[\left(x-\bar{x}\right)^2\right] && \text{(A.3)}\\
&= \bar{x}^2 + \sigma_x^2,
\end{aligned}$$

where the fact that the derivatives greater than two of $g\left(x\right) = x^2$ are zero was applied in Equation (A.2).

The covariance is given by

$$\begin{aligned}
\sigma_y^2 &= E\left[(y-\bar{y})^2\right] && \text{(A.4)}\\
&= \left(\frac{d\mathbf{g}(\bar{x})}{d\mathbf{x}}\right)\sigma_x^2\left(\frac{d\mathbf{g}(\bar{x})}{d\mathbf{x}}\right) - \frac{1}{4}E\left[(x-\bar{x})^2\frac{d^2\mathbf{g}(\bar{x})}{d\mathbf{x}^2}\right]^2 \\
&\quad + E\left[\frac{1}{2}(x-\bar{x})\frac{d\mathbf{g}(\bar{x})}{d\mathbf{x}}(x-\bar{x})^2\frac{d^2\mathbf{g}(\bar{x})}{d\mathbf{x}^2}\right.\\
&\quad + \frac{1}{2}(x-\bar{x})^2\frac{d^2\mathbf{g}(\bar{x})}{d\mathbf{x}^2}(x-\bar{x})\frac{d\mathbf{g}(\bar{x})}{d\mathbf{x}} && \text{(A.5)}\\
&\quad \left.+ \frac{1}{4}(x-\bar{x})^2\frac{d\mathbf{g}(\bar{x})}{d\mathbf{x}}(x-\bar{x})^2\frac{d^2\mathbf{g}(\bar{x})}{d\mathbf{x}^2}\right]\\
&= (2\bar{x})^2\sigma_x^2 - \frac{1}{4}E\left[2(x-\bar{x})^2\right]^2 + E\left[4\bar{x}(x-\bar{x})^3 + (x-\bar{x})^4\right]\\
&= 4\bar{x}^2\sigma_x^2 - \left(\sigma_x^2\right)^2 + 4\bar{x}E\left[(x-\bar{x})^3\right] + E\left[(x-\bar{x})^4\right]. && \text{(A.6)}
\end{aligned}$$

Equation (A.6) for the covariance σ_y^2 can be reduced further given general knowledge about Gaussian random variables. The second part of the third term in Equation (A.6) , i.e. $E\left[(x-\bar{x})^3\right]$, is called the *skewness* of x and for Gaussian random variables this moment is zero [39]. Thereby, the whole third term is zero, resulting in

$$\sigma_y^2 = 4\bar{x}^2\sigma_x^2 - \left(\sigma_x^2\right)^2 + E\left[(x-\bar{x})^4\right]. \tag{A.7}$$

The last term in Equation (A.7) is known as the *kurtosis* of x. For a Gaussian random variable, this can be described using the covariance as follows:

$$E\left[(x-\bar{x})^4\right] = 3\left(\sigma_x^2\right)^2, \tag{A.8}$$

allowing a final reduction of the covariance of y, given as

$$\begin{aligned}
\sigma_y^2 &= 4\bar{x}^2\sigma_x^2 - \left(\sigma_x^2\right)^2 + 3\left(\sigma_x^2\right)^2\\
&= 4\bar{x}^2\sigma_x^2 + 2\left(\sigma_x^2\right)^2.
\end{aligned} \tag{A.9}$$

To see what the EKF computes for the posterior mean, $\bar{y}$, and covariance, σ_x^2, they are calculated using the EKF linearization technique, the first order Taylor Expansion, giving

$$\bar{y}^{lin} = g(\bar{x}) = \bar{x}^2 \tag{A.10}$$

$$\begin{aligned}
\left(\sigma_y^2\right)^{lin} &= \left(\frac{d\mathbf{g}(\bar{x})}{d\mathbf{x}}\right)\sigma_x^2\left(\frac{d\mathbf{g}(\bar{x})}{d\mathbf{x}}\right)\\
&= 4\bar{x}^2\sigma_x^2.
\end{aligned} \tag{A.11}$$

The approximation cut away the higher order terms, leaving only the first components. Comparing Equation (A.3) to Equation (A.10) (mean) and Equation (A.9) to Equation

(A.11) (covariance), the errors of the EKF approximations can be calculated as

$$\begin{aligned} e_{\hat{y}} &= \bar{y} - \bar{y}^{lin} = \left(\bar{x}^2 + \sigma_x^2\right) - \bar{x}^2 = \sigma_x^2, \\ e_{\sigma_y^2} &= \sigma_y^2 - \left(\sigma_y^2\right)^{lin} = \left[4\bar{x}^2\sigma_x^2 + 2\left(\sigma_x^2\right)^2\right] - 4\bar{x}^2\sigma_x^2 = 2\left(\sigma_x^2\right)^2. \end{aligned}$$

It is obvious that the errors of the mean and the covariance are proportional to the covariance of the prior random variable x. In other words, the more the prior random variable x is focused at its mean, the higher the accuracy of the EKF linearization. Consequently, for any probability distribution of x significantly different from a Gaussian probability distribution, the posterior mean $\bar{y}^{lin}$ is biased and the actual posterior covariance σ_y^2 will be underestimated.
Further detailed mathematical and experimental discussion of the flaws of the extended Kalman filter can be found in [12].

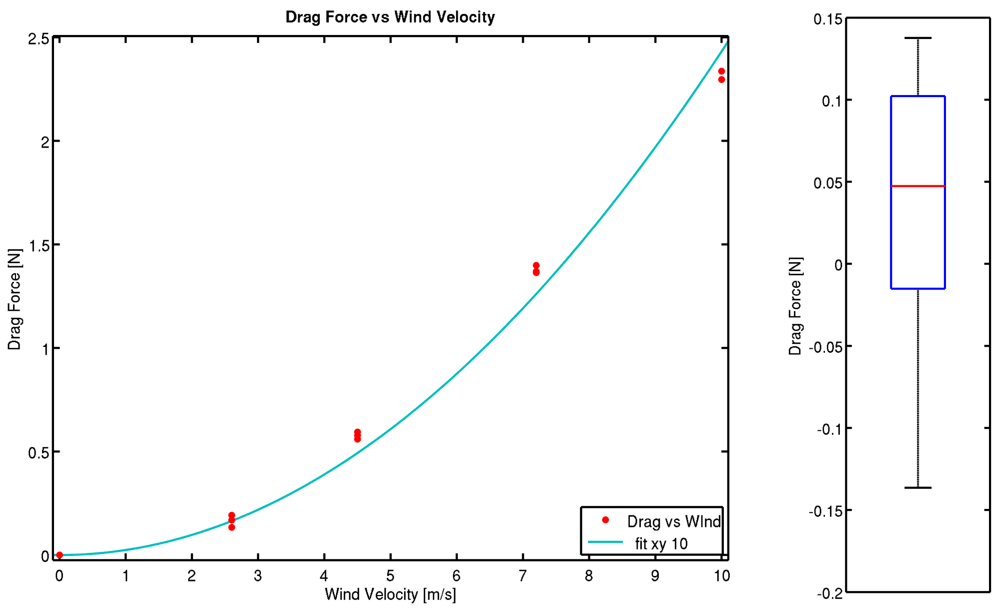
Appendix B

Air Drag Experiments

The additional air drag experiments concerning the coefficients for the $\boldsymbol{x}$-$\boldsymbol{y}$-plane as well as the $\boldsymbol{z}$-axis that were cut from section 4.2.3, are presented here in more detail. The angles discussed here for the $\boldsymbol{x}$-$\boldsymbol{y}$-plane are $10°$, $20°$, $-10°$, $-20°$, $-30°$ and $-40°$. With respect to the $\boldsymbol{z}$-axis the experiments presented in this chapter are for angles of $100°$, $110°$ and $120°$.

B.1 X-Y-plane

B.1.1 10 Degrees



(a) $10°$ $\boldsymbol{x}$-$\boldsymbol{y}$-plane (b) Residuals

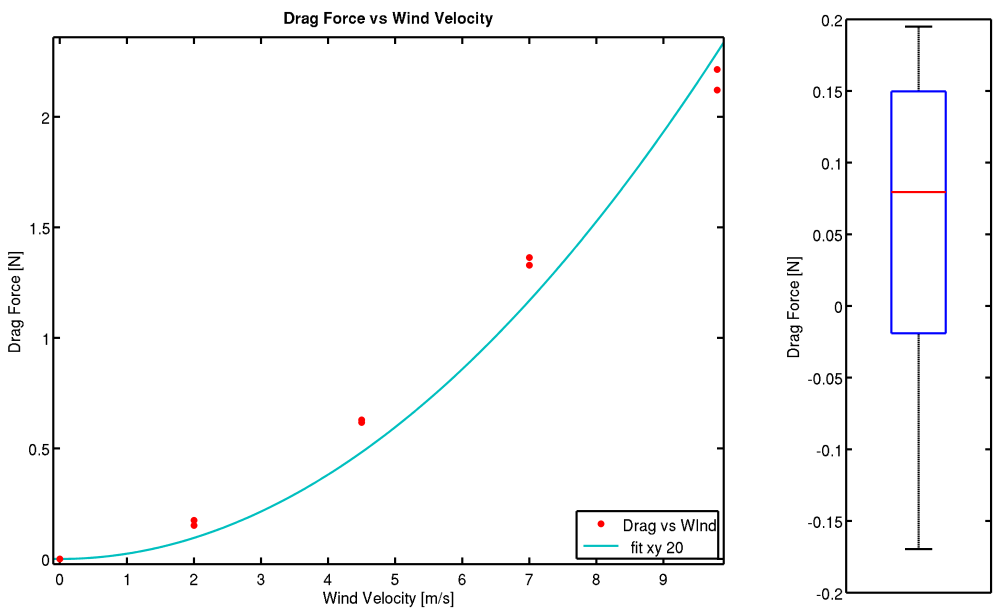
Figure B.1: Drag coefficient experiment results for $10°$ $\boldsymbol{x}$-$\boldsymbol{y}$-plane.

The results of the 10° experiment (Figure B.1) mirror the findings of the first experiment. While most samples fall above the fitted expectation, the drag forces at maximum wind velocities lie underneath their expectations. The samples follow the suggested model, which can be seen in the boxplot analysis of the residuals. There the positions of the median and the box reflect the skewness to above the expectation.

The skew is only $0.05N$, and the RMSE is $\epsilon_{RMSE} = 0.09179N$, which are small in comparison to the actual generated forces. Thus the proposed model approximates the behavior well for this angle of attack. The calculated air drag coefficient is

$$\rho_{xy,10^\circ} = 0.048652\frac{kg}{m}.$$

B.1.2 20 Degrees



(a) 20° $\boldsymbol{x}$-$\boldsymbol{y}$-plane (b) Residuals

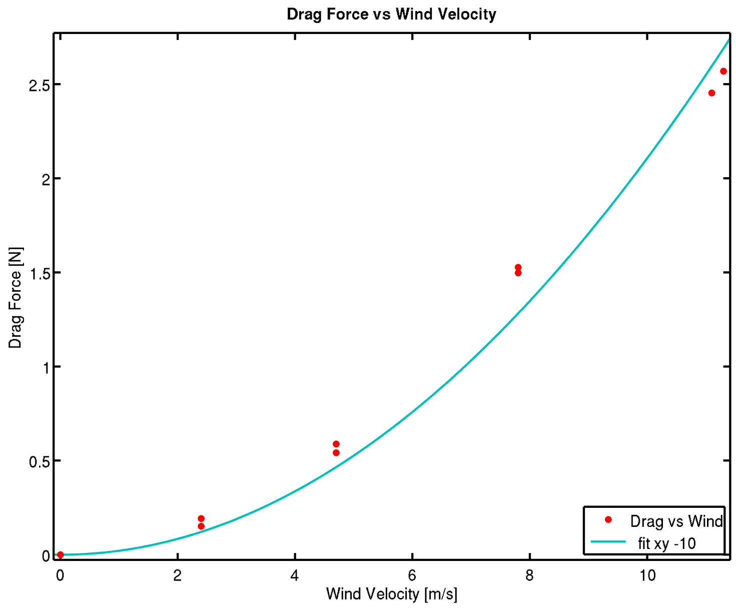
Figure B.2: Drag coefficient experiment results for 20° $\boldsymbol{x}$-$\boldsymbol{y}$-plane.

The experiment results for 20° (Figure B.2) follow the trend of the first experiments for the $\boldsymbol{x}$-$\boldsymbol{y}$-plane. The fitted model captures the proportionality of the force measurements to the wind velocities, but a positive skewness of the samples is evident in the positions of the median and the box. For the results of the 20° experiment, not enough measurements were taken to exclude possible outliers. This is reflected by a large kurtosis, signifying a widely spread distribution of samples around the expectations. A simple way to address this issue and improve the fit is to take more measurements.

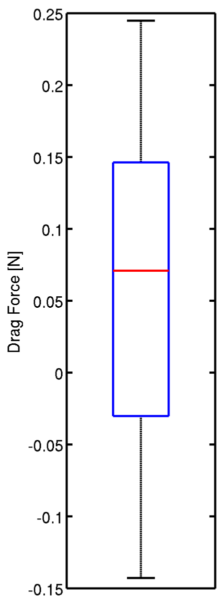
The large residuals could be caused by measurement errors, possibly resulting from interferences by the stabilization software. Thus the model captures the proportionality relationship not as well as before, resulting in an RSME of $\epsilon_{RMSE,20^\circ} = 0.1356N$, which is still a small error considering the magnitudes of induced forces. The calculated coefficient for equation (4.8) is given as

$$\rho_{xy,20^\circ} = 0.047708 \frac{kg}{m}$$

B.1.3 -10 Degrees to -40 Degrees



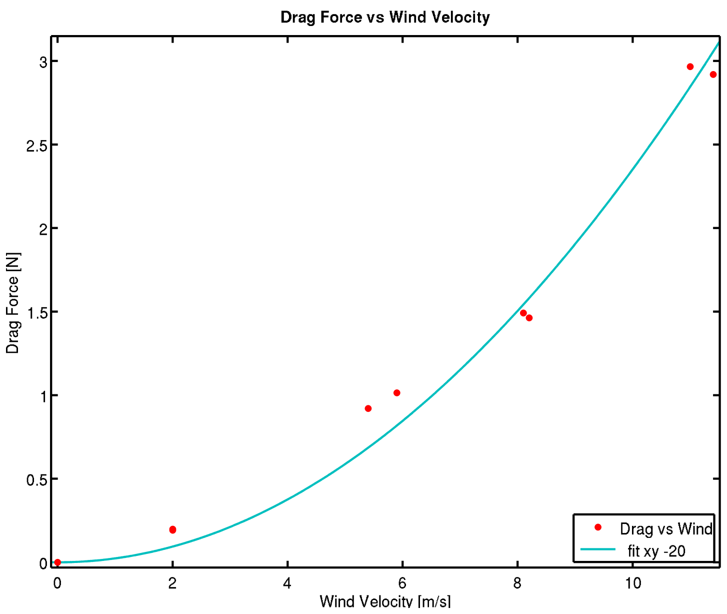
(a) -10° $\boldsymbol{x}$-$\boldsymbol{y}$-plane



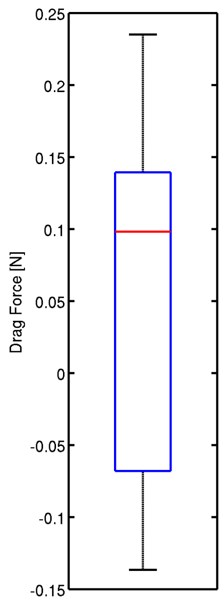
(b) Residuals

Figure B.3: Drag coefficient experiment results for -10° $\boldsymbol{x}$-$\boldsymbol{y}$-plane.

Figures B.3, B.4, B.5 and B.6 present the results of the experiments for the remaining angles concerning the air drag properties of the $\boldsymbol{x}$-$\boldsymbol{y}$-plane of the UAV. These four experiments, angles -10° to -40°, can be discussed jointly, as they all reflect very similar characteristics. The proposed relationship stays close to the actual samples. This is a good indication that the fitted model shows the same properties as the underlying system. While the samples seem larger in majority than the expectation function, analysis via boxplot of the residuals enlightens the distribution of the samples with respect to the expectations. The positions of the median and the box in the boxplots of all four experiments are slightly skewed to the positive. This is a common theme of the proposed model for all the experiments concerning air drag presented here. The box sizes are all comparatively large. As before, this can be an effect of the low number of samples, where large measurement errors

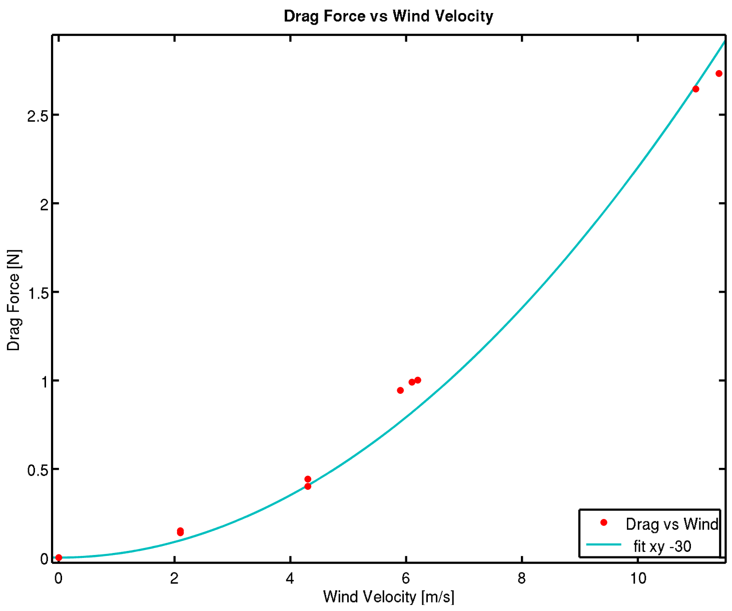


(a) $-20°$ $\boldsymbol{x}$-$\boldsymbol{y}$-plane

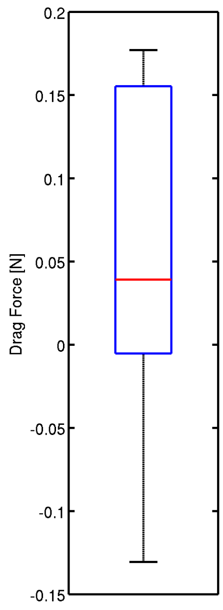


(b) Residuals

Figure B.4: Drag coefficient experiment results for $-20°$ $\boldsymbol{x}$-$\boldsymbol{y}$-plane.

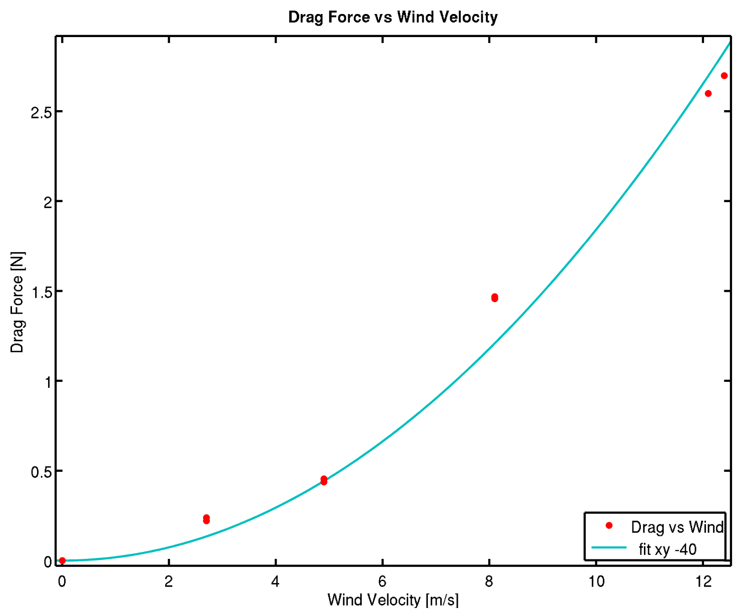


(a) $-30°$ $\boldsymbol{x}$-$\boldsymbol{y}$-plane

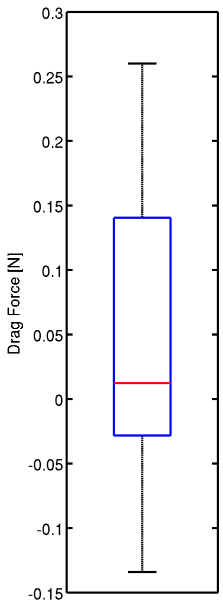


(b) Residuals

Figure B.5: Drag coefficient exp. results for $-30°$ $\boldsymbol{x}$-$\boldsymbol{y}$-plane.



(a) -40° $\boldsymbol{x}$-$\boldsymbol{y}$-plane



(b) Residuals

Figure B.6: Drag coefficient exp. results for -40° $\boldsymbol{x}$-$\boldsymbol{y}$-plane.

are not detected as outliers but count as correct and distort the image. Nevertheless, the box sizes indicate spread out distributions of samples around the fitted model equations. From the box size can be seen that the majority of samples lie within a band of about $0.15N$ around and mainly above the expectations. The root mean square errors for these experiments are given as

$$\begin{aligned}
\epsilon_{RMSE,-10^\circ} &= 0.1451N, \\
\epsilon_{RMSE,-20^\circ} &= 0.1431N, \\
\epsilon_{RMSE,-30^\circ} &= 0.1096N, \\
\epsilon_{RMSE,-40^\circ} &= 0.1481N.
\end{aligned}$$

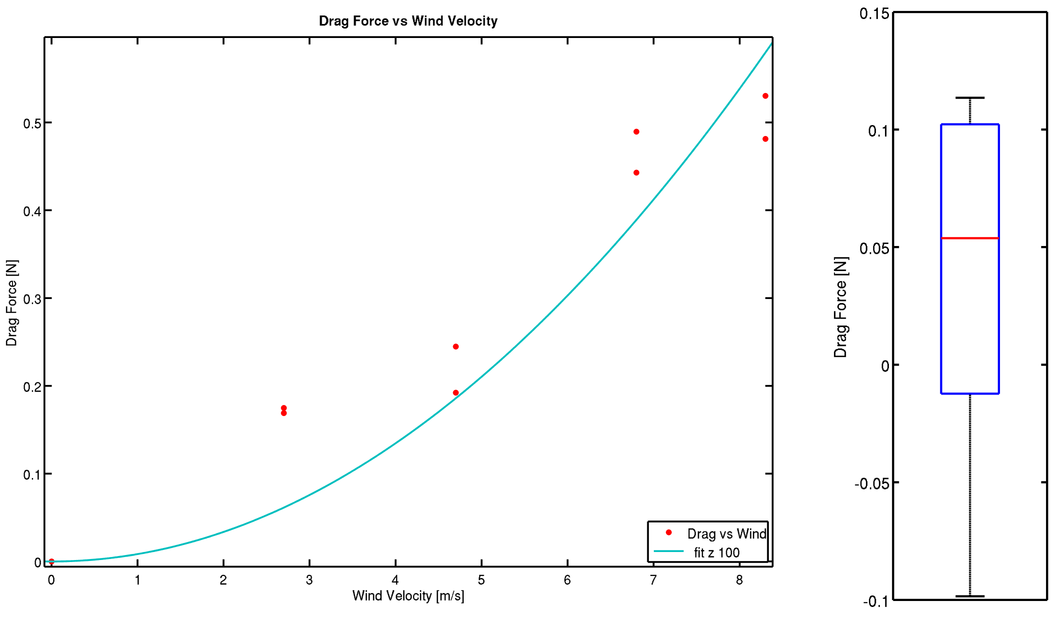
It is clear to see, that the proposed model approximates the actual relationship equally well in all four experiments. The air drag coefficients were calculated as

$$\begin{aligned}
\rho_{xy,-10^\circ} &= 0.042146\frac{kg}{m}, \\
\rho_{xy,-20^\circ} &= 0.047036\frac{kg}{m}, \\
\rho_{xy,-30^\circ} &= 0.044074\frac{kg}{m}, \\
\rho_{xy,-40^\circ} &= 0.036842\frac{kg}{m}.
\end{aligned}$$

Again, the similarity between the values is evident. Only the coefficient for the furthest displaced wind source is too low. The cause for this is supposedly the large angle of the wind. This is covered in the detailed discussion on the effect of the angles on the computed air drag coefficients in section 4.2.3.

B.2 Z-axis

B.2.1 100 Degrees



(a) 100° $\boldsymbol{z}$-axis (b) Residuals

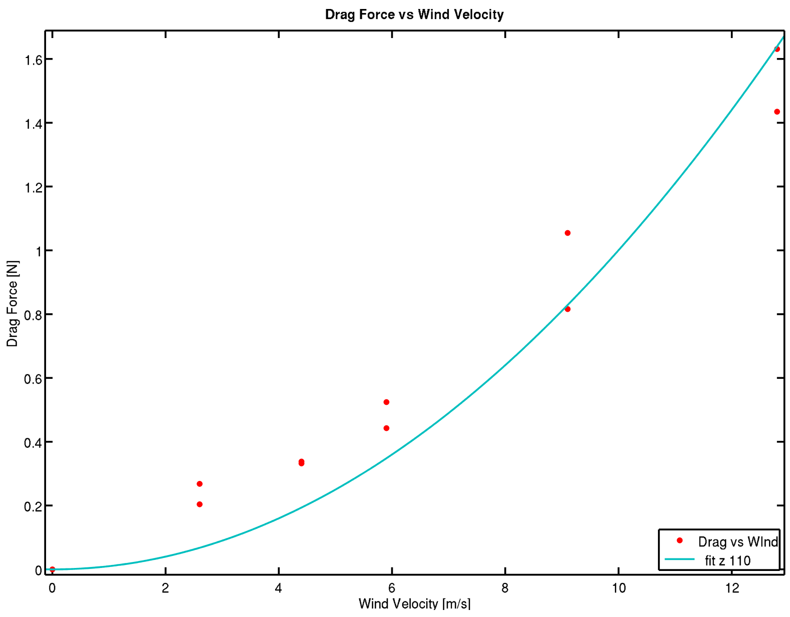
Figure B.7: Drag coefficient exp. results for 100° $\boldsymbol{z}$-axis.
Samples are skewed above the expectations. Kurtosis is large in relation to the scale of induced forces.

The experiment results for 100° direction, which is almost orthogonal to the $\boldsymbol{z}$-axis, are shown in Figure B.7. The first noticeable difference to the former 180° experiment is the scale of the induced drag force. It is only about a tenth of the scale of Figure 4.12. This is an effect of the attack angle of the wind, but the discussion thereof is postponed until all experiments are presented. Considering the distribution of the samples, it is obvious that the majority is above the fitted expectation function. The boxplot clarifies the assumed skewness by the dislocated positions of the median and box. They are positioned in the positive, but indicate a skew of only $0.05N$. The skew represents about 10% of the measured maximum force value, meaning a strong skew of the residuals. The large box in the boxplot also reflects the mediocre fit of the expectation function. The peakedness is not too high, denoting a distribution of samples spread further away around the suggested model. In comparison to the other experiment results, this peakedness is still good, but taking only the measurements from this experiment into account, the model does not fit too great. The

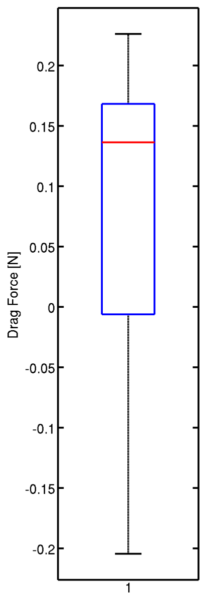
RMSE is $\epsilon_{RMSE,z100^\circ} = 0.08146N$, which is about 16% of the maximum induced drag force in the plot, reflecting the poor accuracy of the fitted model at this resolution. Compared to the RMSE of the 180° experiment, this fit is far better. The calculated air drag coefficient is

$$\rho_{z,100^\circ} = 0.01683\frac{kg}{m}.$$

B.2.2 110 Degrees



(a) 110° z-axis



(b) Residuals

Figure B.8: Drag coefficient exp. results for 110° z-axis.

The results for the 110° experiment (Figure B.8) show that the samples fall all above the expectations, analogous to the previous experiment. Only the drag forces generated at the highest wind speeds are measured to be far below the expected model. But as discussed before, at those wind speeds the measurement readings could have been especially noisy because of the turbulent flight properties. Accordingly, in the boxplot analysis we see that those measurements draw the lower whisker far down, possibly almost classifying as an outlier. The median and the box position both reflect the strong skew of the samples to stay above the fitted model. A larger number of samples could assist in detecting measurement errors and disqualifying them as outliers. The box size is also relatively large. This indicates a distribution with a large kurtosis, i.e. the samples are spread further away from the expectations. How bad this affects the accuracy can be seen in the RMSE. It is $\epsilon_{RMSE,z110^\circ} = 0.1523N$, in the same range of accuracy as the 180° experiment. The air drag coefficient was calculated as

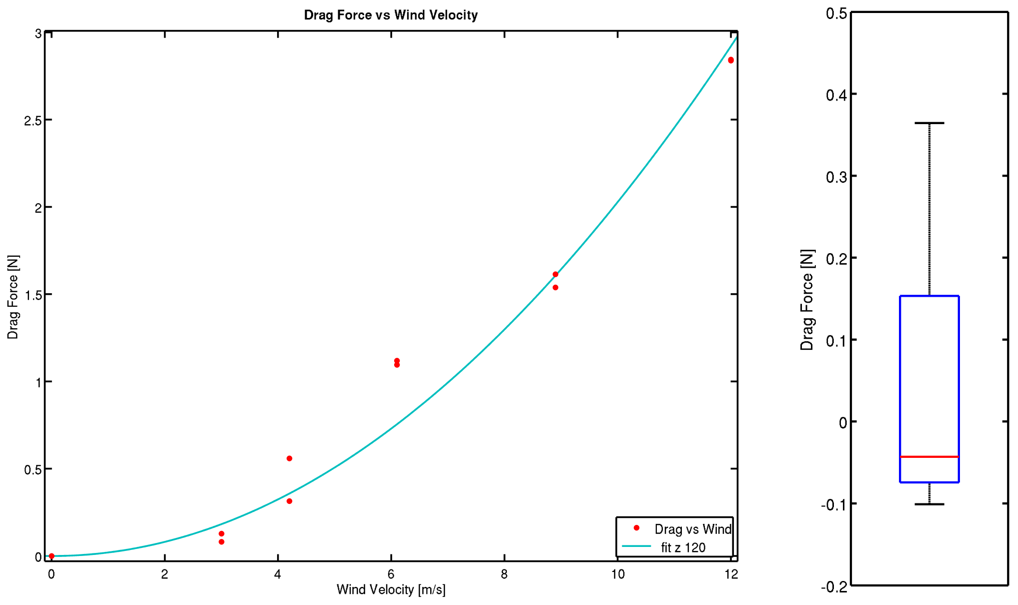
$$\rho_{z,110^\circ} = 0.02001 \frac{kg}{m}.$$

B.2.3 120 Degrees

Figure B.9 show the results for an angle of 120°. Those results show a better fit of the measurements to the fitted model. The samples are located both below and above the expectation function, indicating a distribution with little skewness. The boxplot analysis shows, that the skewness is in fact a little to below the expectation. The box size shows a not too dense distribution of the samples around the expectation. This could be a result of having too few samples, therefore the noisy measurements lying far above the expectations affect the statistics relative to the remaining measurements too strongly. The RMSE is $\epsilon_{RMSE,z120^\circ} = 0.1794N$, so the error is a little bigger than for the other $\boldsymbol{z}$-axis experiments. The air drag coefficient was computed as

$$\rho_{z,120^\circ} = 0.040562 \frac{kg}{m},$$

which follows the trend observed at the experiments concerning the $\boldsymbol{x}$-$\boldsymbol{y}$-plane. The bigger the angle between the wind direction and the measured induced force direction, the smaller the induced force. More is discussed in the paragraph on the effect of these angles.



(a) 120° $\boldsymbol{z}$-axis (b) Residuals

Figure B.9: Drag Coefficient Exp. Results for 120° $\boldsymbol{z}$-axis.

Appendix C

Algorithms

Algorithm C.1: The Kalman Filter

```
Input: $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$, $\mathbf{P}_0 = E\left[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T\right]$
Result: $\hat{\mathbf{x}}_k$ is optimal estimate of $\mathbf{x}_k$, $\mathbf{P}_k$ its error covariance.
Loop as desired.
begin
    Time Update
    1) Forward $\mathbf{x}_{k-1}$ in time
        $\hat{\mathbf{x}}_k^- = \mathbf{F}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k$
    2) Update a priori estimate error covariance
        $\mathbf{P}_k^- = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}$
    Measurement Update
    1) Calculate Kalman Gain
        $\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T\left(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}\right)^{-1}$
    2) Correct $\hat{\mathbf{x}}_k^-$ with measurement $\mathbf{z}_k$
        $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-\right)$
    3) Update a posteriori estimate error covariance
        $\mathbf{P}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-$
end
```

Algorithm C.2: The Extended Kalman Filter

```
Input: $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$, $\mathbf{P}_0 = E\left[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T\right]$
Result: $\hat{\mathbf{x}}_k$ is stimate of $\mathbf{x}_k$, $\mathbf{P}_k$ its error covariance.

Loop as desired.
begin
    Time Update
    1) Forward $\mathbf{x}_{k-1}$ in time
        $\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k, 0)$
    2) Calculate Jacobians $\mathbf{F}_k$ and $\mathbf{V}_k$ at point $\hat{\mathbf{x}}_k^-$
    3) Update a priori estimate error covariance
        $\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{V}_k \mathbf{Q}_k \mathbf{V}_k^T$
    Measurement Update
    1) Calculate Jacobians $\mathbf{H}_k$ and $\mathbf{L}_k$ at point $\hat{\mathbf{x}}_k^-$
    2) Calculate Kalman Gain
        $\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{L}_k \mathbf{R}_k \mathbf{L}_k^T\right)^{-1}$
    3) Correct $\hat{\mathbf{x}}_k^-$ with measurement $\mathbf{z}_k$
        $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \left(\mathbf{z}_k - \mathbf{h}\left(\hat{\mathbf{x}}_k^-, 0\right)\right)$
    4) Update a posteriori estimate error covariance
        $\mathbf{P}_k = (\mathbf{1} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-$
end
```

Algorithm C.3: The Scaled Unscented Transformation

```
Input: Initial mean $\bar{\mathbf{x}} \in \mathbb{R}^N$, covariance $\mathbf{P}_x$, parameters $\alpha$, $\beta$, $\kappa$;
Result: Accurate estimates of posterior mean, covariance and
        cross-covariance.
Set $\lambda = \alpha^2 (N + \kappa) - N$, $\gamma = \sqrt{N + \lambda}$.
begin
    1) Selection of Sigma Points:
        $\mathbf{X}_0 = \bar{\mathbf{x}} \quad , w_0^m = \frac{\lambda}{N+\lambda} \quad (i = 0)$
        $\mathbf{X}_i = \bar{\mathbf{x}} + \left(\sqrt{(N+\lambda)\mathbf{P}_x}\right)_i \quad (0 < i \leq N), w_0^c = \frac{\lambda}{N+\lambda} + \left(1 - \alpha^2 + \beta\right) \quad (i = 0)$
        $\mathbf{X}_i = \bar{\mathbf{x}} - \left(\sqrt{(N+\lambda)\mathbf{P}_x}\right)_{i-N} \quad (N < i \leq 2N), w_i^m = \quad w_i^c = \frac{1}{2(N+\lambda)} \quad (0 < i \leq 2N)$
        // auxiliary operator $(\cdot)_j$ returns the $j$-th column of the
        argument matrix
    2) Propagate each Sigma Point through the nonlinear transformation $\mathbf{g}$:
        $\mathbf{y}_i = \mathbf{g}(\mathbf{X}_i) \qquad \forall \quad i \in \{0, \ldots, 2n\}$
    3) Compute posterior mean, covariance and cross-covariance:
        $\bar{\mathbf{y}} \approx \sum_{i=0}^{2N} w_i \mathbf{y}_i$
        $\mathbf{P}_y \approx \sum_{i=0}^{2N} w_i (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^T$
        $\mathbf{P}_{xy} \approx \sum_{i=0}^{2N} w_i (\mathbf{X}_i - \bar{\mathbf{x}})(\mathbf{y}_i - \bar{\mathbf{y}})^T$
end
```

Algorithm C.4: The Unscented Kalman Filter

```
Input: Initial mean $\bar{\mathbf{x}}_\mathbf{0} \in \mathbb{R}^N$, covariance $\mathbf{P}_{x,0}$, parameters $\alpha$, $\beta$, $\kappa$;
Output: $m_k$ is optimal estimate of $\mathbf{x}_k$ with covariance $\mathbf{P}_{x,k}$
  Set $\lambda = \alpha^2 (N+\kappa) - N$, $\gamma = \sqrt{N+\lambda}$. Calculate Weights:
    $w_0^{(m)} = \frac{\lambda}{N+\lambda}$ , $w_0^{(c)} = \frac{\lambda}{N+\lambda} + \left(1 - \alpha^2 + \beta\right)$
    $w_i^{(m)} = w_i^{(c)} = \frac{1}{2(N+\lambda)}$   $(0 < i \leq 2N)$
    $W = (\mathbf{1} - [w_m \quad w_m \quad \cdots \quad w_m])\, diag\,(w_c)\,(\mathbf{1} - [w_m \quad w_m \quad \cdots \quad w_m])^T$
begin
    Time Update
    1) Calculate Sigma Points
        $\hat{\mathbf{X}}_{k-1} = [m_{k-1} \quad m_{k-1} \quad \cdots \quad m_{k-1}] + \gamma \left[0 \quad \sqrt{\mathbf{P}_{k-1}} \quad -\sqrt{\mathbf{P}_{k-1}}\right]$
    2) Propagate $\hat{\mathbf{X}}_{k-1}$ through nonlinear state transition function, then
    update a priori mean and covariance:
        $\hat{\mathbf{X}}'_k = \mathbf{f}\left(\hat{\mathbf{X}}_{k-1}, t_k\right)$
        $m_k^- = \hat{\mathbf{X}}'_k w_m$
        $\mathbf{P}_k^- = \hat{\mathbf{X}}'_k W \left(\hat{\mathbf{X}}'_k\right)^T + \mathbf{Q}_k$
    Measurement Update
    1) Calculate Sigma Points
        $\hat{\mathbf{X}}_k^- = \left[m_k^- \quad m_k^- \quad \cdots \quad m_k^-\right] + \gamma \left[0 \quad \sqrt{\mathbf{P}_k^-} \quad -\sqrt{\mathbf{P}_k^-}\right]$
    2) Propagate $\hat{\mathbf{X}}_k^-$ through nonlinear observation model function, then
    calculate observation mean, covariance and cross-covariance:
        $\bar{\mathbf{Z}}_k = \mathbf{h}\left(\hat{\mathbf{X}}_k^-, k\right) w_m = \mathbf{Z}_k^- w_m$
        $\mathbf{P}_{y,k} = \mathbf{Z}_k^- W \left(\mathbf{Z}_k^-\right)^T + \mathbf{R}_k$
        $\mathbf{P}_{xy,k} = \mathbf{X}_k^- W \left(\mathbf{Z}_k^-\right)^T$
    3) Calculate Kalman Gain
        $\mathbf{K}_k = \mathbf{P}_{xy,k} \left(\mathbf{P}_{y,k}\right)^{-1}$
    4) Correct $m_k^-$ with measurement $\mathbf{Z}_k$ and update a posteriori estimate
    error covariance
        $m_k = m_k^- + \mathbf{K}_k \left(\mathbf{Z}_k - \bar{\mathbf{Z}}_k\right)$
        $\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_{y,k} \left(\mathbf{K}_k\right)^T$
end
```

Algorithm C.5: The Square Root Unscented Kalman Bucy Filter

```
**Input**: Initial mean $\bar{\mathbf{x}}(t_0) = m(t_0) \in \mathbb{R}^N$, cov. $\mathbf{P}_x(t_0)$, params. $\alpha$, $\beta$, $\kappa$;
**Output**: $m(t_k)$ is estimate of $\mathbf{x}(t_k)$
  Set $\lambda = \alpha^2(N+\kappa) - N$, $\gamma = \sqrt{N+\lambda}$. Calculate $w_m$, $w_c$ and $W$ as in C.4.
**begin**
    ***Time & Measurement Update***
    *1) Calculate Sigma Points*

        $$\hat{\mathbf{X}}(t) = [m(t) \quad m(t) \quad \cdots \quad m(t)] + \gamma [0 \quad A(t) \quad -A(t)]$$

    *2) Differential equations corresponding to predict and correct processes of discrete time UKF:*

        $$\mathbf{K}(t) = \hat{\mathbf{X}}(t) W \left(\mathbf{h}\left(\hat{\mathbf{X}}(t), t\right)\right)^T \left[\mathbf{V}(t)\mathbf{R}(t)(\mathbf{V}(t))^T\right]^{-1}$$
        $$M(t) = (\mathbf{A}(t))^{-1}\left[\hat{\mathbf{X}}(t) W \left(\mathbf{f}\left(\hat{\mathbf{X}}(t), t\right)\right)^T + \mathbf{f}\left(\hat{\mathbf{X}}(t), t\right) W \left(\hat{\mathbf{X}}(t)\right)^T + \mathbf{L}(t)\mathbf{Q}(t)(\mathbf{L}(t))^T - \mathbf{K}(t)\mathbf{V}(t)\mathbf{R}(t)(\mathbf{V}(t))^T(\mathbf{K}(t))^T\right]\left((\mathbf{A}(t))^{-1}\right)^T$$
        $$\frac{d}{dt}\left[\hat{\mathbf{X}}(t)\right]_i = \left[\mathbf{g}\left(\hat{\mathbf{X}}(t), \mathbf{A}(t), t\right)\right]_i = \mathbf{f}\left(\hat{\mathbf{X}}(t), t\right) w_m + \mathbf{K}(t)\left[\mathbf{z} - \mathbf{h}\left(\hat{\mathbf{X}}(t), t\right) w_m\right] + \gamma\left[0 \quad \mathbf{A}(t)\Phi(M(t)) \quad -\mathbf{A}(t)\Phi(M(t))\right]_i$$

    *2) Solve the differential equations, integrate over time interval to update Sigma Points:*

        $$\left[\hat{\mathbf{X}}(t_k)\right]_i = \left[\hat{\mathbf{X}}(t_{k-1})\right]_i + \int_{t_{k-1}}^{t_k} \left[\mathbf{g}\left(\hat{\mathbf{X}}(t_{k-1}), \mathbf{A}(t_{k-1}), t_k\right)\right]_i dt \qquad \forall i \in \{0, \ldots, 2N\}$$
**end**
```

Algorithm C.6: The Square Root continuous-discrete UKF

```
Input: Initial mean $\bar{\mathbf{x}}(t_0) = m(t_0) \in \mathbb{R}^N$, cov. $\mathbf{P}_x(t_0)$, params. $\alpha$, $\beta$, $\kappa$;
Output: $m(t_k)$ is optimal estimate of $\mathbf{x}(t_k)$
  Set $\lambda = \alpha^2(N+\kappa) - N$, $\gamma = \sqrt{N+\lambda}$. Calculate $w_m$, $w_c$ and $W$ as in C.4.
begin
    Time Update
    1) Calculate Sigma Points
        $\hat{\mathbf{X}}(t_{k-1}) = [m(t_{k-1}) \quad \cdots \quad m(t_{k-1})] + \gamma [0 \quad A(t_{k-1}) \quad -A(t_{k-1})]$
    2) Propagate $\hat{\mathbf{X}}(t_{k-1})$
        $M(t_k) = (\mathbf{A}(t_{k-1}))^{-1} \Big[\hat{\mathbf{X}}(t_{k-1}) W \left(\mathbf{f}\left(\hat{\mathbf{X}}(t_{k-1}), t_k\right)\right)^T$
        $\quad + \mathbf{f}\left(\hat{\mathbf{X}}(t_{k-1}), t_k\right) W \left(\hat{\mathbf{X}}(t_{k-1})\right)^T$
        $\quad + \mathbf{L}(t_k) \mathbf{Q}(t_k) (\mathbf{L}(t_k))^T\Big] \left((\mathbf{A}(t_{k-1}))^{-1}\right)^T$
        $\frac{d}{dt}\left[\hat{\mathbf{X}}^-(t_k)\right]_i = \left[\mathbf{g}\left(\hat{\mathbf{X}}(t_{k-1}), \mathbf{A}(t_{k-1}), t_k\right)\right]_i$
        $\quad = \mathbf{f}\left(\hat{\mathbf{X}}(t_{k-1}), t_k\right) w_m$
        $\quad + \gamma [0 \quad \mathbf{A}(t_{k-1}) \Phi(M(t_k)) \quad -\mathbf{A}(t_{k-1}) \Phi(M(t_k))]_i$
        $\left[\hat{\mathbf{X}}^-(t_k)\right]_i = \left[\hat{\mathbf{X}}(t_{k-1})\right]_i + \int_{t_{k-1}}^{t_k} \left[\mathbf{g}\left(\hat{\mathbf{X}}(t_{k-1}), \mathbf{A}(t_{k-1}), t_k\right)\right]_i dt \quad 0 \le i \le 2N$
    Measurement Update
    2) Calculate predicted measurement from $\hat{\mathbf{X}}^-(t_k)$
        $\bar{\mathbf{z}}_k = \hat{\mathbf{Z}}_k w_m = \mathbf{h}\left(\hat{\mathbf{X}}_k^-, t_k\right) w_m$
        $\mathbf{A}'_{z,k} = \mathbf{qr}\left\{\sqrt{w_1^c}\left(\hat{\mathbf{Z}}_k - \bar{\mathbf{z}}_k\right)_{1:2N,k} \quad \sqrt{\mathbf{R}}\right\}$
        $\mathbf{A}_{z,k} = \mathbf{cholupdate}\left\{\mathbf{A}'_{z,k}, \left(\hat{\mathbf{Z}}_k - \bar{\mathbf{z}}_k\right)_{0,k}, w_0^c\right\}$
        $\mathbf{P}_{xz,k} = \mathbf{X}_k^- W \left(\hat{\mathbf{Z}}_k\right)^T$
    3) Calculate Kalman Gain
        $\mathbf{K}_k = \left(\mathbf{P}_{xz,k}\left((\mathbf{A}_{z,k})^{-1}\right)^T\right)(\mathbf{A}_{z,k})^{-1}$
    4) Correct $m_k^-$ with measurement $\mathbf{z}_k$ and update $\mathbf{A}_k$
        $m_k = m_k^- + \mathbf{K}_k(\mathbf{z}_k - \bar{\mathbf{z}}_k)$
        $\mathbf{U} = \mathbf{K}_k \mathbf{A}_{z,k}$
        $\mathbf{A}_k = \mathbf{cholupdate}\left\{\mathbf{A}_k^-, U, -1\right\}$
end
```

Algorithm C.7: The Square Root UKF with Latency Compensation

```
Input: Initial mean $\bar{\mathbf{x}}_{\mathbf{0}} \in \mathbb{R}^N$, covariance $\mathbf{P}_{x,0}$, parameters $\alpha$, $\beta$, $\kappa$;
Output: $m_k$ is optimal estimate of $\mathbf{x}_k$ with covariance $\mathbf{P}_{x,k}$
  Set $\lambda = \alpha^2 (N+\kappa) - N$, $\gamma = \sqrt{N+\lambda}$. Calculate $w_m$, $w_c$ and $W$ as in C.4.
begin
    Time Update
    1) Calculate Sigma Points
        $\mathbf{X}_{k-1} = [\bar{\mathbf{x}}_{k-1} \quad \bar{\mathbf{x}}_{k-1} + \gamma\mathbf{A}_{k-1} \quad \bar{\mathbf{x}}_{k-1} - \gamma\mathbf{A}_{k-1}]$
    2) Propagate $\hat{\mathbf{X}}_{k-1}$ through state transition function, update $\bar{\mathbf{x}}_k^-$ and $\mathbf{A}_k^-$
        $\mathbf{X}'_k = \breve{\mathbf{f}}(\mathbf{X}_{k-1}, t_k, \mathbf{v}_k)$
        $\bar{\mathbf{x}}_k^- = \hat{\mathbf{X}}'_k w_m$
        $\mathbf{A}'_k = \mathbf{qr}\left\{ \sqrt{w_1^c}\left(\hat{\mathbf{X}}_k - \bar{\mathbf{x}}_k^-\right)_{1:2N,k} \quad \sqrt{\mathbf{Q}} \right\}$
        $\mathbf{A}_k^- = \mathbf{cholupdate}\left\{ \mathbf{A}'_k, \left(\hat{\mathbf{X}}_k - \bar{\mathbf{x}}_k^-\right)_{0,k}, w_0^c \right\}$
    Measurement Update
    1) Recalculate sigma points using predicted mean and covariance square root
        $\mathbf{X}_k^- = \left[\bar{\mathbf{x}}_k^- \quad \bar{\mathbf{x}}_k^- + \gamma\mathbf{A}_k^- \quad \bar{\mathbf{x}}_k^- - \gamma\mathbf{A}_k^-\right]$
    2) Calculate predicted measurement from $\hat{\mathbf{X}}^-(t_k)$
        $\bar{\mathbf{z}}_k = \hat{\mathbf{Z}}_k w_m = \breve{\mathbf{h}}\left(\mathbf{X}_k^-, t_k, \mathbf{n}_k\right) w_m$
        $\mathbf{A}'_{y,k} = \mathbf{qr}\left\{ \sqrt{w_1^c}\left(\hat{\mathbf{Z}}_k - \bar{\mathbf{z}}_k\right)_{1:2N,k} \quad \sqrt{\mathbf{R}} \right\}$
        $\mathbf{A}_{y,k} = \mathbf{cholupdate}\left\{ \mathbf{A}'_{y,k}, \left(\hat{\mathbf{Z}}_k - \bar{\mathbf{z}}_k\right)_{0,k}, w_0^c \right\}$
        $\mathbf{P}_{xy,k} = \mathbf{X}_k^- W \left(\hat{\mathbf{Z}}_k\right)^T$
    3) Calculate Kalman Gain and Correct $m_k^-$ with measurement $\mathbf{z}_k$ and update $\mathbf{A}_k$
        $\mathbf{K}_k = \mathbf{P}_{xy,k}\left(\left(\mathbf{A}_{y,k}\right)^{-1}\right)^T \left(\mathbf{A}_{y,k}\right)^{-1}$
        $\hat{\mathbf{x}}_k = \bar{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{z}_k - \bar{\mathbf{z}}_k\right)$
        $\mathbf{U} = \mathbf{K}_k \mathbf{A}_{y,k}$
        $\mathbf{A}_k = \mathbf{cholupdate}\left\{\mathbf{A}_k^-, U, -1\right\}$
    4) Crop posterior $m_k^-$ and $\mathbf{A}_k$ if necessary (if $\mathbf{z}_k = \mathbf{z}_{lag}$)
        $\hat{\mathbf{x}}_k^a = \begin{pmatrix} \hat{\mathbf{x}}_k \\ \hat{\mathbf{x}}_{lag} \end{pmatrix} \Rightarrow \hat{\mathbf{x}}_k$
        $\mathbf{A}_{x_k^a}^a = \begin{bmatrix} \mathbf{A}_{x_k} & \mathbf{A}_{x_k x_{lag}} \\ \mathbf{A}_{x_{lag} x_k} & \mathbf{A}_{x_{lag}} \end{bmatrix} \Rightarrow \mathbf{A}_{x_k}$
end
```