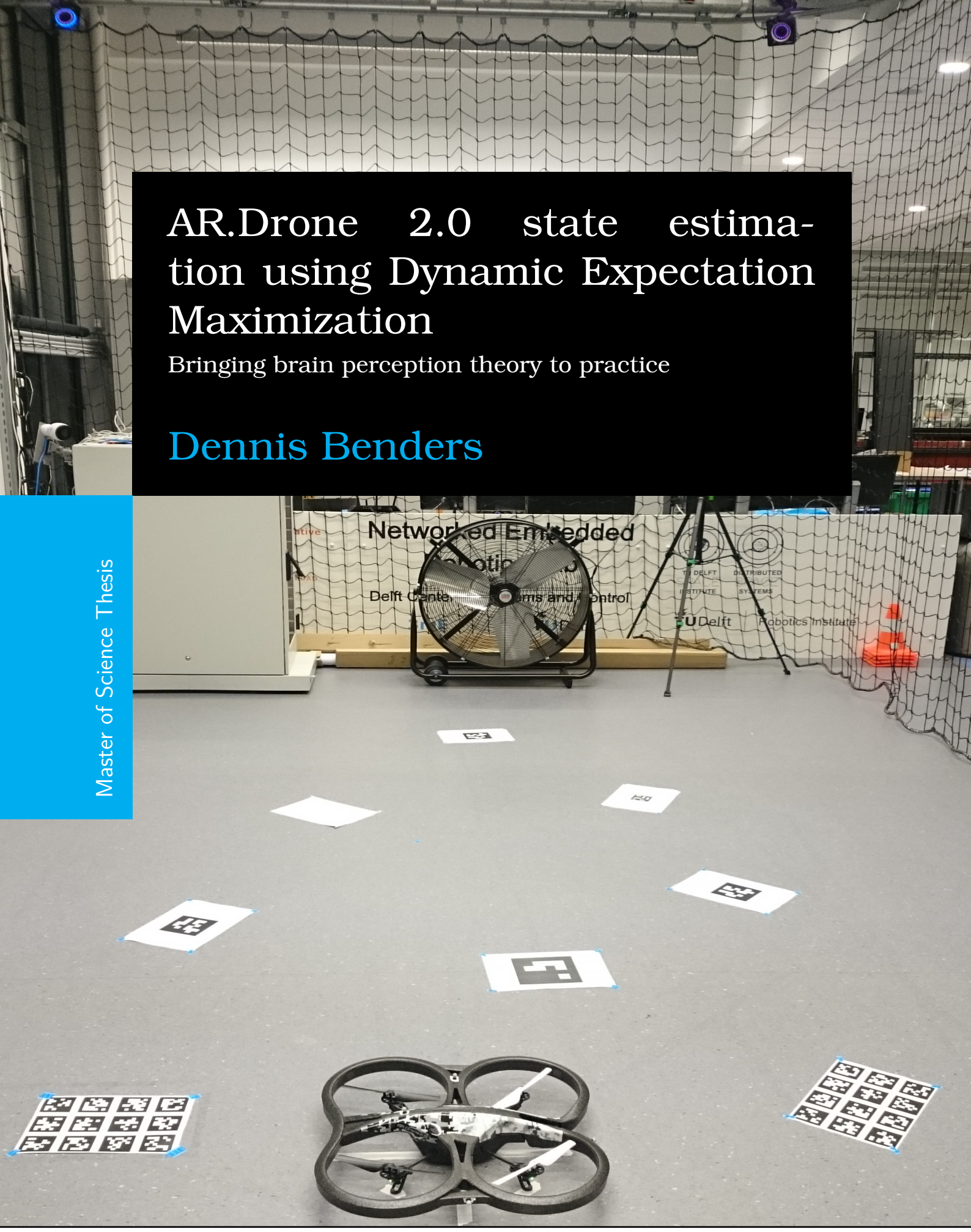


AR.Drone 2.0 state estimation using Dynamic Expectation Maximization

Bringing brain perception theory to practice

Dennis Benders

Master of Science Thesis

TU Delft Delft University of Technology

Department of Cognitive Robotics - Embedded Systems

AR.Drone 2.0 state estimation using Dynamic Expectation Maximization

Bringing brain perception theory to practice

Master of Science Thesis

For the degree of Master of Science in Embedded Systems at Delft University of Technology

Dennis Benders

November 2, 2021

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) and Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Department of Cognitive Robotics (CoR)

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

AR.Drone 2.0 state estimation using Dynamic Expectation Maximization

by

Dennis Benders

in partial fulfillment of the requirements for the degree of

Master of Science Embedded Systems

Dated: November 2, 2021

Supervisor(s):

Prof.dr.ir. M. Wisse. Thesis advisor

Ir. A. A. Meera. Daily supervisor

Reader(s):

Dr.ir. L. Ferranti. Third reader

Dr.ir. P. M. Esfahani. Fourth reader

Abstract

Recent developments in neuroscience research, mainly introduced by neuroscientist Karl J. Friston, have resulted in a concept called the Free Energy Principle (FEP). The FEP is a brain theory unifying action, perception and learning. An important observation is that autonomous robots have to perform similar tasks to the human brain. Therefore, these developments are very interesting from a control engineering perspective.

A lot of research is going on in the direction of Active Inference (AI), which includes the perception and action parts of the FEP used for filtering and control on physical systems. However, a research gap exists for the theory describing the complex mathematical backbone of perception and learning, called Dynamic Expectation Maximization (DEM). DEM is a parameter estimation algorithm that can be used to perform filtering as well as system identification. This thesis is part of the research aiming to evaluate the performance of DEM as a filtering as well as system identification technique. To be more specific, this thesis considers the filtering part.

In general, a filter is meant to derive the states of a system using a system model and sensor measurements. The system model is often not perfect, resulting in process noise, and the sensors provide noisy data, resulting in measurement noise. Many existing filtering techniques, including the conventional Kalman filter, assume these noises to be white. However, the process noise contains unmodelled system dynamics that introduce correlation. Therefore, by definition, the process noise is not white, but coloured. The main advantage of using the DEM filter is the ability to extract information from the correlation in the measurements and the noises to construct a better state estimate.

Simulation results in previous work indicated the potential of DEM as a filter. The next step is to prove the usefulness of the DEM filter on a physical system. In order to do so, this thesis describes the design of an experimental setup with a quadrotor Unmanned Aerial Vehicle (UAV) used to evaluate the performance of the DEM filter. The recorded flight data is used to analyze the process and measurement noises. Furthermore, the DEM filter is compared with the conventional Kalman filter. It turns out that for a relatively big subspace of DEM tuning parameters, the DEM filter is able to outperform the Kalman filter, which proves the potential of this filtering algorithm in a practical robotics setting.

Table of Contents

List of Figures

List of Tables

Glossary

List of Acronyms

AI	Active Inference
ASL	Autonomous Systems Lab
CAD	Computer-Aided Design
COM	Centre Of Mass
CCW	CounterClockWise
CW	ClockWise
DAQ	Data AcQuisition
DC	Direct Current
DCM	Dynamic Causal Model
DCSC	Delft Center for Systems and Control
DEM	Dynamic Expectation Maximization
DOF	Degrees Of Freedom
ELBO	Evidence Lower BOund
ENU	East-North-Up
EOM	Equations Of Motion
ETH	Eidgenössische Technische Hochschule
ESC	Electronic Speed Controller
FE	Free Energy
FEP	Free Energy Principle
FLU	Front-Left-Up
FOV	Field Of View
FPS	Frames Per Second
GUI	Graphical User Interface
IMU	Inertial Measurement Unit

IR	InfraRed
KL	Kullback-Leibler
LOS	Line Of Sight
LTI	Linear and Time-Invariant
MAP	Maximum A Posteriori
MoCap	Motion Capture
MSE	Mean Squared Error
NERDlab	Networked Embedded Robotics lab in Delft
PDF	Probability Density Function
PWM	Pulse Width Modulation
ROS	Robot Operating System
RPM	Revolutions Per Minute
SDK	Software Development Kit
SFU	Simon Fraser University
SSE	Sum of Squares Error
TTV	Time To Value
TUD	TU Delft
TUM	Technical University of Munich
UDP	User Datagram Protocol
USB	Universal Serial Bus
UAV	Unmanned Aerial Vehicle
VBI	Variational Bayesian Inference
VFE	Variational Free Energy
VI	Virtual Instrument
Wi-Fi	Wireless Fidelity
WWW	World Wide Web

List of Symbols

α_0	Initial rotation angle in bifilar pendulum experiment
$\boldsymbol{\epsilon} = \begin{bmatrix} \boldsymbol{\epsilon}_y \\ \boldsymbol{\epsilon}_x \end{bmatrix}$	Vector containing the state and output prediction error
$\boldsymbol{\eta}_u$	Prior expectation of input
$\boldsymbol{\lambda}$	Hyperparameters
$\boldsymbol{\mu}$	Mean/mode (subscript indicates the quantity considered)

$\boldsymbol{\omega}$	White noise signal, used to construct a coloured noise signal by filtering with a Gaussian filter (subscript indicates the quantity considered)
$\boldsymbol{\tau} = \begin{bmatrix} \tau_{r1} \\ \tau_{r2} \\ \tau_{r3} \end{bmatrix} + \begin{bmatrix} \tau_{g1} \\ \tau_{g2} \\ \tau_{g3} \end{bmatrix}$	Torque resulting from the quadrotor rotor blade rotations, expressed in the body frame. First vector relates to the direct torques generated by the rotors. Second vector relates to the quadrotor gyroscopic effects. By neglecting gyroscopic effects, the following holds: $\boldsymbol{\tau} = \begin{bmatrix} \tau_{r1} \\ \tau_{r2} \\ \tau_{r3} \end{bmatrix} = \begin{bmatrix} \tau_{\phi} \\ \tau_{\theta} \\ \tau_{\psi} \end{bmatrix}$
$\boldsymbol{\theta}$	Model parameters
$\boldsymbol{\vartheta}$	Without subscript: complete parameter set ($\boldsymbol{\vartheta} = \{X, \boldsymbol{\theta}, \boldsymbol{\lambda}\}$) With subscript: specific parameter subset (subscript indicates the parameter set considered)
$\dot{\psi}_c$	Commanded yaw rate
κ	Learning rate, used in DEM state estimation
ω	Disturbance term, used in nonlinear quadrotor model
$\Omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$	Quadrotor rotational/angular velocity, expressed with respect to the body frame
ω_i	Rotational velocity of rotor i
ω_r	Rotor rotational velocity (used to denote rotor rotational velocity in general, in contrast to ω_i, which is rotor-specific)
ϕ	Euler roll angle
ϕ_c	Commanded roll angle
Π	Precision matrix (inverse covariance matrix: Σ^{-1}; subscript indicates the quantity considered)
ψ	Euler yaw angle
Σ	Covariance matrix (subscript indicates the quantity considered)
σ	Standard deviation (subscript indicates the quantity considered)
τ	Time lag in autocorrelation calculation
θ	Euler pitch angle
θ_c	Commanded pitch angle
$U(\boldsymbol{\vartheta}, \boldsymbol{y})$	Internal energy
$\boldsymbol{c}_A$	Vector containing two coefficients, used to translate pwmA_i to ω_i
$\boldsymbol{c}_M$	Vector containing two coefficients, used to translate pwmM_i to ω_i
$\boldsymbol{c}_Q$	Vector containing two coefficients, used to translate ω_i to τ_{r3}. As opposed to c_Q (scalar), $\boldsymbol{c}_Q$ (vector) better fits the identification experiment data
$\boldsymbol{c}_T$	Vector containing two coefficients, used to translate ω_i to T, τ_{r1} and τ_{r2}. As opposed to c_T (scalar), $\boldsymbol{c}_T$ (vector) better fits the identification experiment data

$\boldsymbol{c}_{P\phi}$	Vector containing three coefficients, used to directly translate pwmA_i to τ_{r1}
$\boldsymbol{c}_{P\psi}$	Vector containing three coefficients, used to directly translate pwmA_i to τ_{r3}
$\boldsymbol{c}_{P\theta}$	Vector containing three coefficients, used to directly translate pwmA_i to τ_{r2}
$\boldsymbol{c}_{PT}$	Vector containing three coefficients, used to directly translate pwmA_i to T
$\boldsymbol{f} = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}$	Force resulting from the quadrotor rotor blade rotations, expressed in the body frame
$\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$	State update function (dynamic model)
$\boldsymbol{f}_s = \boldsymbol{f}_s(\boldsymbol{u})$	State update function (static model)
$\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})$	Output update function (dynamic model)
$\boldsymbol{g}_s = \boldsymbol{g}_s(\boldsymbol{x}, \boldsymbol{u})$	Output update function (static model)
$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$	Position, expressed in the inertial frame
$\boldsymbol{u}$	Inputs
$\boldsymbol{u}_e = \begin{bmatrix} \text{pwmA}_{eq} \\ \text{pwmA}_{eq} \\ \text{pwmA}_{eq} \\ \text{pwmA}_{eq} \end{bmatrix}$	Input equilibrium
$\boldsymbol{v} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$	Velocity, expressed in a frame with the same orientation as the body frame
$\boldsymbol{w}$	Process noise
$\boldsymbol{x}$	States
$\boldsymbol{x}_e = \begin{bmatrix} x_e \\ y_e \\ z_e \\ \dot{x}_e \\ \dot{y}_e \\ \dot{z}_e \\ \phi_e \\ \theta_e \\ \psi_e \\ \dot{\phi}_e \\ \dot{\theta}_e \\ \dot{\psi}_e \end{bmatrix}$	State equilibrium
$\boldsymbol{y}$	Outputs
$\boldsymbol{y}_n$	Part of output containing noise dynamics

$\boldsymbol{z}$	Measurement noise
Δt	Sampling time
$\dot{z}_c$	Commanded vertical speed
$\mathcal{B} = \{\hat{\boldsymbol{b}}_x, \hat{\boldsymbol{b}}_y, \hat{\boldsymbol{b}}_z\}$	Body frame
$\mathcal{D}$	Derivative operator
$\mathcal{F}(\boldsymbol{\vartheta}, \boldsymbol{y})$	Variational Free Energy
$\mathcal{I} = \{\hat{\boldsymbol{i}}_x, \hat{\boldsymbol{i}}_y, \hat{\boldsymbol{i}}_z\}$	Inertial frame
$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	Gaussian/normal distribution with mode $\boldsymbol{\mu}$ and covariance matrix Σ
${}^{\mathcal{B}}R_{\mathcal{I}} = {}^{\mathcal{I}}R_{\mathcal{B}}^T$	Rotation matrix to convert coordinates expressed in the inertial frame to coordinates expressed in the body frame
${}^{\mathcal{I}}R_{\mathcal{B}}$	Rotation matrix to convert coordinates expressed in the body frame to coordinates expressed in the inertial frame
${}^{\mathcal{I}}R_{r,\mathcal{B}}$	Rotation matrix to convert angles expressed in the body frame to angles expressed in the inertial frame
${}^{\mathcal{I}}R_{X,\mathcal{B}}$	Rotation matrix used to rotate coordinates around the x-axis
${}^{\mathcal{I}}R_{Y,\mathcal{B}}$	Rotation matrix used to rotate coordinates around the y-axis
${}^{\mathcal{I}}R_{Z,\mathcal{B}}$	Rotation matrix used to rotate coordinates around the z-axis
τ_i	Torque generated by rotor i
pwmA_i	Motor PWM value given to motor of rotor i, according to the convention in the AR.Drone 2.0 software
pwmA_{eq}	AR.Drone 2.0 motor PWM value in hovering equilibrium
pwmM_i	Motor PWM value given to motor of rotor i, according to the convention in the MATLAB/Simulink toolbox
A	State matrix
A_d	Discretized state matrix
B	Input matrix
B_d	Discretized input matrix
C	Output matrix
C_d	Discretized output matrix
c_Q	Torque coefficient
c_T	Thrust coefficient
C_u	Covariance matrix of input prior
$c_{B\phi}$	Constant in B, used to translate pwmA_i to $\ddot{\phi}$
$c_{B\psi}$	Constant in B, used to translate pwmA_i to $\ddot{\psi}$
$c_{B\theta}$	Constant in B, used to translate pwmA_i to $\ddot{\theta}$
c_{Bz}	Constant in B, used to translate pwmA_i to $\ddot{z}$
d	Input embedding order
D_1	Damping parameter 1 in bifilar pendulum experiment
D_2	Damping parameter 2 in bifilar pendulum experiment
d_w	Distance between wires in bifilar pendulum experiment

$D_{KL}(\cdot)$	KL divergence, indicating how well one distribution approximates another distribution ($D_{KL}(\cdot) = 0$ means that both distributions are identical)
E	Finite differences matrix
g	Gravitational acceleration
$G(\boldsymbol{\vartheta}, \boldsymbol{y})$	Internal energy, expected under the recognition density
$H(\boldsymbol{\vartheta})$	Entropy of recognition density
$h(t)$	Gaussian filter (subscript indicates the quantity considered)
I	Without subscript: inertia matrix With subscript: identity matrix (subscript indicates matrix size)
I_r	Rotor inertia
I_{xx}	Mass moment of inertia around x-axis
$I_{xy} = I_{yx}$	Products of inertia around x- and y-axis (assuming a symmetric quadrotor)
$I_{xz} = I_{zx}$	Products of inertia around x- and z-axis (assuming a symmetric quadrotor)
I_{yy}	Mass moment of inertia around y-axis
$I_{yz} = I_{zy}$	Products of inertia around y- and z-axis (assuming a symmetric quadrotor)
I_{zz}	Mass moment of inertia around z-axis
K	Scaling factor of Gaussian filter (in this thesis equal to $\sqrt{\frac{\Delta t}{\sigma_h \sqrt{\pi}}}$)
l	Rotor arm length
l_w	Wire length in bifilar pendulum experiment
$ln(p(\boldsymbol{y}))$	Model log-evidence
m	Quadrotor mass
n	Dimension (subscript indicates the parameter set considered)
p	State and output embedding order (do not confuse with p (angular velocity), which is always used as part of a vector, in contrast to this embedding order)
$p(\boldsymbol{\vartheta})$	Posterior density of generative model
$p(\boldsymbol{\vartheta}, \boldsymbol{y})$	Joint density of generative model
$p(\boldsymbol{y})$	Marginal likelihood of generative model
$q(\boldsymbol{\vartheta})$	Recognition/ensemble density
s	Smoothness (subscript indicates quantity considered; using Gaussian filter assumption: $s = \sigma$)
$S(s^2)$	Temporal correlation matrix with smoothness parameter s
T	Total thrust generated by all four rotors
t	Time
T_i	Thrust generated by rotor i
T_o	Period of damped oscillation in bifilar pendulum experiment
$V(\boldsymbol{\vartheta})$	Variational energy

$w_1 = w_\phi$	Roll angle process noise
$w_2 = w_{\dot{\phi}}$	Roll rate process noise
$w_{2,res}$	Residual after fitting a Fourier series to w_2
X	Parameter set containing states and inputs
x_1	Quadrotor rotation angle in bifilar pendulum experiment
x_2	Quadrotor rotational velocity in bifilar pendulum experiment
Z	Normalization constant (subscript indicates the parameter set considered)
z	Roll angle measurement noise (do not confuse with z position, which is always used as part of a vector, in contrast to this measurement noise)
z_0	Quadrotor height in hovering equilibrium, expressed in $\mathcal{I}$
batA	Akku-King battery (has more charge capacity than original Parrot battery)
batP	Original Parrot battery

Acknowledgements

First of all, I would like to thank my supervisors Martijn and Ajith. Despite COVID-19 circumstances, you were available for regular online meetings. I really appreciate the content-related discussions we had during these meetings. They helped me in setting the right priorities in my thesis and really pushed my understanding of the topic to a higher level. I never expected to be able to understand this topic in such level of detail.

Secondly, I would like to thank Jos van Driel from the Meetshop at TU Delft for providing me with the force sensors, their readout system and the hand-tachometer. Without these components, I would not have been able to conduct the thrust and torque identification experiments the way I did. The experiments have stimulated my ability to understand the physical effects in mechanical systems a lot and I am very grateful for that.

Thirdly, I would like to thank Gijs van der Hoorn for helping me to gain insight in how to properly set up a GitHub repository. The insights obtained are not only useful for this thesis, but also for the rest of my career.

Fourthly, I would like to thank my main fellow students during this thesis: Bastiaan and Iris. You provided me a reference of other thesis work that helped me realize that a thesis can never be perfect. Because of you I was able to accept that my design choices were not always the ideal ones.

Fifthly, I would like to thank my girlfriend Fenna. In times of stress, especially towards the end of my thesis, you were always there with a positive mindset and trusted me to successfully finish my thesis. Furthermore, you helped me to accurately fill in the system identification experiment data in this report and to proofread the whole report, which I really appreciate.

Last, but certainly not least, I would like to thank my parents and sister. You allowed me to conduct the system identification experiments in the living room and took that into account by moving as little as possible during the measurements to avoid system vibrations. But far more important than that: thank you for helping me to keep a positive mindset and putting my whole thesis in perspective. The latter made me realize that I should not spend too much time on specific details, but that I should rather focus on the important parts to be able to finish my thesis eventually.

Delft, University of Technology — Dennis Benders
November 2, 2021

"The difference between theory and practice is in theory somewhat smaller than in practice."

— *Frank Westphal*

Chapter 1

Introduction

In this introductory chapter, Section 1-1 elaborates on the context and motivation of the research in this thesis. Based on the context, Section 1-2 states the main research question and thesis outlook, based on the research sub-goals. Section 1-3 provides an overview of the conventions used in mathematical expressions throughout this thesis. Finally, Section 1-4 shortly indicates which software is used in this thesis and where it can be found.

1-1 Context and motivation

In the world we are living in, humans continuously try to make processes more efficient in order to save time to focus on things that we think are important. This has resulted in the usage of machines and engines (consider important inventions, such as the steam engine). After these inventions, already quite some time ago, chips were invented and processes got electrified. Systems were able to execute a pre-determined physical action using their own sensor feedback (e.g. manipulators as part of assembly lines). We have now arrived at the point where machines are becoming more intelligent and can autonomously make decisions about the action they need to perform. These machines are called robots. Examples include, but are certainly not limited to, autonomous underwater vehicles, autonomously flying robots and self-driving cars. This development has led to people trying to define what a robot actually is. An example of the definition of a robot is [1]:

> *A robot is an autonomous machine capable of sensing its environment, carrying out computations to make decisions, and performing actions in the real world.*

In the definition, you may recognize three main tasks that a robot should be capable of: sensing, making decisions and performing actions. In control engineering terms this can also be expressed as filtering and controlling. The filtering part ensures that the desired information is extracted from the sensory signals. The controlling part performs calculations to decide what control action should be performed next, based on this information.

This thesis considers the filtering part of robots. A filter often contains a model with the important dynamics of the physical system concerned and how the sensory inputs are generated from the physical quantities describing the system. These physical quantities are called states. As mentioned above, robots are becoming so complex that the model does not provide all details necessary to describe how the robot states evolve over time. This results in a modelling error. The modelling error will be referred to as process noise. Furthermore, the robot sensors are non-ideal and therefore contain fluctuations in the values they provide to represent the states. These fluctuations are not not important for describing the robot states, resulting in a measurement error. The measurement error will be referred to as measurement noise. The goal of a filter is to remove the measurement noise as much as possible and be able to infer the states in the real world, despite the presence of process noise.

A lot of filtering solutions assume the process and measurement noises to have completely random effects over time, which is called white noise. However, if the model is not perfect, the process noise will contain non-modelled system dynamics. By definition, these system dynamics cannot change infinitely fast, so the process noise is not white, but coloured. The difference between white and coloured noise is the fact that white noise is not differentiable by definition, while coloured noise is. We can exploit the fact that process noise is differentiable to better infer the system states.

Recently, the neuroscientist Karl J. Friston has developed a theory that makes use of coloured noise, called the Free Energy Principle (FEP). Originally, FEP is a theory trying to explain the functionality of the brain [2], thereby unifying action, perception and learning [3]. It turns out that neuroscientists use a so-called Dynamic Causal Model (DCM) to describe processes in the brain, whereas control engineers use these same type of model to describe the dynamics of physical systems. Therefore, we can say that perception corresponds to filtering, action to controlling and learning to system identification. The FEP has led Friston to come up with a parameter estimation algorithm, called Dynamic Expectation Maximization (DEM) [4]. DEM includes perception and learning. This thesis focuses on the state estimation part of DEM perception (also referred to as the DEM filter).

As far as we are aware of in our research group, DEM state estimation has never been applied to a physical robotics system. Previous work has already shown promising simulation results of the DEM filter outperforming the conventional Kalman filter [5] in the presence of coloured noise using synthetic data [4], [6]. There is still a lack of evidence that the DEM filter will also outperform the Kalman filter using data recorded in a physical experimental setup. The main contribution of this thesis is to show the performance of the DEM filter using experimental data.

The physical system chosen to evaluate the filter performance is a quadrotor Unmanned Aerial Vehicle (UAV). The choice for a quadrotor UAV is based on two arguments. First of all, the quadrotor is a nonlinear system. Aspects like blade geometry as well as existing air flows around the quadrotor have impact on the quadrotor movements. Therefore, it is difficult to model all quadrotor dynamics and coloured process noise will always exist. Secondly, UAVs in general are becoming more important in our society, since they are a cheap solution to a wide variety of problems, including law enforcement, search and rescue operations, exploration of environments that pose a potential danger to humans, packet delivery, aerial photography, etcetera. Having a state estimation algorithm that is proven to work on UAVs will help in developing more stable and robust control algorithms and stimulates the usage of UAVs.

1-2 Research objective and thesis outlook

The main contribution of this thesis is to provide an answer to the following research question:

How does the DEM filter perform on experimental data of a quadrotor flight?

To answer this research question, several sub-goals are defined:

1) *Provide the fundamental ideas behind DEM and derive the DEM filter equations.*
DEM itself is a parameter estimation algorithm. It is able to do more than just state estimation: it can also estimate inputs, model parameters and hyperparameters. This estimation process is based on the minimization of Free Energy (FE). Chapter 2 will describe what free energy actually is and how it can be used to derive the DEM state estimation equations.

2) *Define the quadrotor dynamical model used in the DEM filter.*
As Chapter 2 will show, the DEM filter needs a model of the quadrotor in order to estimate the states, based on input-output data. To have a better understanding of how the state estimation works, the model that will be considered is a relatively simple Linear and Time-Invariant (LTI) state-space model. The definition of this model will be given in Chapter 3.

3) *Design the experimental setup used to record experiment data.*
Before being able to record data that can be used to perform state estimation using the DEM filter, suitable hardware and software platforms need to be chosen. The Parrot AR.Drone 2.0 and OptiTrack system will be used as hardware platform. Robot Operating System (ROS) packages for AR.Drone 2.0 and OptiTrack support will be used as software. Chapter 4 will elaborate on the design choices made to select the hardware and software and explain how the experiment is conducted.

4) *Identify the model parameters for the AR.Drone 2.0.*
The quadrotor model as provided in Chapter 3 contains several parameters that are quadrotor-specific. Therefore, their values need to be identified for the AR.Drone 2.0. Chapter 5 will describe the identification experiment design and results.

5) *Analyze the process and measurement noise.*
An important characteristic of the DEM filter is the fact that it can deal with coloured noise. Chapter 6 will show the presence of coloured noise and analyze corresponding noise characteristics to show whether or not the assumptions made in the DEM filter derivation are valid.

6) *Provide the filter results and discuss and draw conclusions from the results.*
After characterizing the noises, it is time to run the filter and evaluate its performance. Chapter 7 will provide the filter results and show in which tuning parameter subspace the DEM filter outperforms its benchmark, the conventional Kalman filter.
Based on the results, Chapter 8 will draw the main conclusions of this research, answer the research question, list the thesis contributions, discuss possible research improvements and define future work.

1-3 Notation conventions

A lot of different mathematical notations are used in the literature describing DEM and quadrotor modelling approaches. Therefore, this section gives an overview of the notation conventions followed in this thesis:

- The following conventions apply to the way quantities are denoted:
  - Vectors are written in bold (e.g. $\boldsymbol{x}$).
  - All matrices are written in capital letters (e.g. A). Take care: capital letters do not only denote matrices, but also coordinate frames, functions and scalars.
  - All time-variant quantities are defined in continuous time. Therefore, their explicit dependency on time is omitted for notional convenience (e.g. $\boldsymbol{x}(t)$ is written as $\boldsymbol{x}$). A quantity value in discrete-time is denoted by using a subscript with time indication (e.g. $\boldsymbol{x}_k$ and $\boldsymbol{y}_{t+\Delta t}$).
- The following conventions apply to the mathematical accents used:
  - Optimal value is indicated with an asterisk (e.g. Σ^*).
  - Temporal derivative value is indicated with a dot or superscript (e.g. $\dot{\boldsymbol{x}}$ and $\boldsymbol{x}^{(p)}$).
  - Estimated value is indicated with a hat (e.g. $\hat{\boldsymbol{x}}$).
  - Belief of value (in the brain/agent), including its derivatives, is indicated with a tilde (e.g. $\tilde{\boldsymbol{x}}$).
- The following conventions apply to the mathematical operators used:
  - Partial derivatives are written using a fraction, possibly with evaluation condition (e.g. $\dot{\boldsymbol{x}} = \frac{\partial \boldsymbol{x}}{\partial t}$, $\frac{\partial^2 U(\boldsymbol{\mu}_i, \boldsymbol{y})}{\partial \boldsymbol{\vartheta}_i^2}$ and $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)}$).
  - The ceil ($\lceil \cdot \rceil$) and floor ($\lfloor \cdot \rfloor$) operators are used to round up and down to the nearest integer, respectively (e.g. $\lceil \frac{p}{2} \rceil$ and $\lfloor \frac{p}{2} \rfloor$).
  - The Kronecker tensor product is used to multiply the complete quantity on the right-hand side with each element of the quantity on the left-hand side (e.g. $S(s_w^2) \otimes \Sigma_w$)
- The following convention applies to the coordinate frames used:
  - The body frame, $\mathcal{B}$, and inertial frame, $\mathcal{I}$, are used as superscript and subscript to indicate the direction of a coordinate rotation using a rotation matrix (e.g. ${}^{\mathcal{I}}R_{\mathcal{B}}$ is the rotation matrix used to convert coordinates expressed in $\mathcal{B}$ to coordinates expressed in $\mathcal{I}$).

1-4 Software

All results presented in this thesis are obtained using several ROS and MATLAB packages. Appendix F indicates where these packages can be found and how they can be installed in order to provide the ability to reproduce the results and to stimulate future research.

Chapter 2

Dynamic Expectation Maximization

This chapter describes the main algorithm discussed in this thesis: the DEM filter. The goal of this chapter is to give a mathematical derivation, together with an intuitive explanation, of the filter. The derivation starts in Section 2-1 by explaining the basic principle underlying DEM: the FEP. Using the FEP, it can be shown that the DEM filter is a biologically plausible filter in the sense that it should maximize VFE to get a proper state estimate (the same as what biological systems are assumed to do to survive). Maximizing VFE is a non-trivial and mathematically complex problem. Therefore, Section 2-2 will show how to maximize VFE in case of a static model by making two important approximations: the mean-field and Laplace approximation. Section 2-3 will extend the result for static models to hold for dynamic models as well. This involves taking derivatives of all system quantities and assuming the system noises to be coloured. Finally, Section 2-4 will show how the maximization of VFE for dynamic models leads to the state update rule of the DEM filter and what parameters are involved.

2-1 The Free Energy Principle

The FEP is based on the nature of biological systems. It is a theory providing a computationally tractable solution to the optimization of the so-called FE that tries to explain what biological systems drives to survive. This section will shortly touch upon the existence of biological systems and their need to have a model of their environment (Section 2-1-1), show what this model looks like (Section 2-1-2) and explain why and how we can use FE to mathematically describe the behaviour of biological systems (Section 2-1-3). In the end, Section 2-1-4 will provide a short summary of this section by connecting the ideas behind biological systems to the described mathematics.

2-1-1 The nature of biological systems

As Friston states in [2], biological systems are their own proof of existence. They can only survive if they are able to adapt to their environment (e.g. avoid extreme temperatures

by moving to an area with lower temperatures). This is explainable from a Darwinist or selectionist point of view: if a system is not able to adapt to the environment, it will make decisions that become fatal at some point. As a result, the surviving biological systems are the ones able to adapt to the environment.

In order to adapt to the environment, Von Helmholtz came up with the idea that a biological system should have some kind of environment representation (called a generative model [7]). Using the generative model, a system (also called agent) is able to forecast how the environment and its own internal milieu (also called generative process) is changing and what influence the actions of the agent will have on itself and the environment. For an agent to survive, it should find itself in a limited amount of states in this model. In order to do so, it is said to resist a tendency to disorder [3], [8]. Therefore, an agent that successfully adapts often finds itself in this limited amount of states. These states thus have a high probability of occurrence, while the remaining states have low probability. Equivalently, the agent is searching for states that have low entropy. Since entropy is the long-term average of sensory surprisal, it is equivalent to saying that an agent is trying to ensure that the environment and its internal milieu are in a state, such that it can predict what it will observe via its sensors. The states for which this holds are the states with low entropy that ensure that the agent is able to survive.

Evolution ensures that the generative model keeps improving over generations. Each generation passes its knowledge on to the next generation in the form of prior knowledge. It is therefore called innate value [3]. As mentioned, within a generation, each agent tries to find states with low entropy. However, sitting in a dark room and completely shielding yourself from reality for forever is not very helpful to survive. Therefore, an agent has to interact with the environment [9]. It turns out that an agent performs several tasks simultaneously. These tasks are taking place at four different time scales and are listed from long-term to short-term tasks below:

1. Updating prior knowledge: an agent inherits prior knowledge from the previous generation, but due to a changing environment and new insights, this knowledge expands during the lifetime of an agent and is updated via a hierarchical structure and the updating of the processes described below.

2. Learning: an agent tries to understand how the environment and its internal milieu are functioning and tries to derive causal relations from it in the form of the generative model. In control engineering, learning corresponds to finding the model parameters via system identification.

3. Attention: on a shorter time scale than learning the model, an agent tries to find out how much its predictions deviate from what is really happening in the world in the form of "probabilistic contingencies" [2]. In control engineering, attention corresponds to finding the process and measurement noise parameters.

4. Action and perception: an agent is acting on and perceiving sensory information from the environment and its internal milieu in real-time. There exists well-established research showing that the two tasks happen simultaneously [10]. In control engineering, action corresponds to control and perception to filtering.

In the rest of this chapter, we consider only one agent having prior knowledge and an environment representation. In theory, DEM includes the other tasks (except for action): learning (model parameter estimation), attention (hyperparameter estimation) and perception (state and input estimation).

2-1-2 The generative model

One can make the generative model as complex as desired. In the context of this thesis, the generative model follows the DCM convention, which is used by Friston to explain neuroscientific phenomena, as well as by control engineers to describe physical system behaviour. In general, a DCM is given by the following state and output equation:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{w} \tag{2-1}$$
$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{z} \tag{2-2}$$

The state equation describes how the states $\boldsymbol{x} \in \mathbb{R}^{n_x}$ evolve over time as a function of states and inputs/actions $\boldsymbol{u} \in \mathbb{R}^{n_u}$. An agent can only infer the environmental states, based on its sensory inputs $\boldsymbol{y} \in \mathbb{R}^{n_y}$. The sensory inputs do not necessarily capture all states. The state and output function $\boldsymbol{f} \in \mathbb{R}^{n_x}$ and $\boldsymbol{g} \in \mathbb{R}^{n_y}$ are parameterized by model parameters $\boldsymbol{\theta}$. The sensors of the agent are not perfect, thereby introducing measurement noise $\boldsymbol{z} \in \mathbb{R}^{n_y}$. Furthermore, the generative model of the agent does not fully represent the environment dynamics, causing the existence of process noise $\boldsymbol{w} \in \mathbb{R}^{n_x}$.

In general, the agent knows the structure of the state and output functions, as well as the definition of the states, inputs and outputs. It is also acknowledged that an agent (or a brain, more specifically) acts as an Bayesian inference machine [11], [12]. This means that both state and output equation are represented by a conditional Probability Density Function (PDF):

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{w} \quad \rightarrow \quad p(\boldsymbol{x}|\boldsymbol{\vartheta}) \tag{2-3}$$
$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{z} \quad \rightarrow \quad p(\boldsymbol{y}|\boldsymbol{\vartheta}) \tag{2-4}$$

in which:

$$\boldsymbol{\vartheta} = \begin{cases} X = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{u} \end{bmatrix} \\ \boldsymbol{\theta} \\ \boldsymbol{\lambda} \end{cases} \tag{2-5}$$

where $\boldsymbol{\vartheta}$ forms the complete set of parameters to be estimated, including states and inputs (combined represented by X), model parameters $\boldsymbol{\theta}$ and hyperparameters $\boldsymbol{\lambda}$ [4].

2-1-3 Variational Free Energy

In general, the generative model of the agent does not exactly resemble the generative process. Therefore, the agent has to derive the parameters in the generative model using its sensory observations in order to match the two as closely as possible. Deriving parameters from observations is equivalent to calculating the posterior density $p(\boldsymbol{\vartheta}|\boldsymbol{y})$. In machine learning, this is done using Bayes' theorem:

$$p(\boldsymbol{\vartheta}|\boldsymbol{y}) = \frac{p(\boldsymbol{\vartheta},\boldsymbol{y})}{p(\boldsymbol{y})} = \frac{p(\boldsymbol{y}|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta})}{p(\boldsymbol{y})} \tag{2-6}$$

However, calculating the likelihood $p(\boldsymbol{y})$ is intractable, since it requires taking the integral over all states in the generative process: $p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta})d\boldsymbol{\vartheta}$ [13]. Therefore, it is proposed to approximate the posterior density with the so-called recognition density $q(\boldsymbol{\vartheta})$ [14]:

$$q(\boldsymbol{\vartheta}) \approx p(\boldsymbol{\vartheta}|\boldsymbol{y}) \tag{2-7}$$

This is also referred to as Variational Bayesian Inference (VBI) [15] and forms the basis for the FEP.

The metric used to assess the goodness of approximation using $q(\boldsymbol{\vartheta})$ is the (reverse) Kullback-Leibler (KL) divergence [16]:

$$D_{KL}(q(\boldsymbol{\vartheta})||p(\boldsymbol{\vartheta}|\boldsymbol{y})) = \int_{-\infty}^{\infty} q(\boldsymbol{\vartheta}) ln\left(\frac{q(\boldsymbol{\vartheta})}{p(\boldsymbol{\vartheta}|\boldsymbol{y})}\right) d\boldsymbol{\vartheta} \tag{2-8}$$

An important property of the KL divergence is that it is a non-negative measure [17]. In the ideal case, $q(\boldsymbol{\vartheta}) = p(\boldsymbol{\vartheta}|\boldsymbol{y})$, the integrand of the KL divergence equals 0, and so does the KL divergence itself. The more the recognition density and posterior density differ in likeliness of one (or multiple) parameters, the higher the value of the KL divergence becomes.

Writing out the KL divergence expression (and taking the integral only over the domain in which $\boldsymbol{\vartheta}$ has non-zero values, thereby leaving out the $-\infty$ and ∞ bounds for notational convenience) gives:

$$\begin{aligned} D_{KL}(q(\boldsymbol{\vartheta})||p(\boldsymbol{\vartheta}|\boldsymbol{y})) &= \int q(\boldsymbol{\vartheta}) ln\left(\frac{q(\boldsymbol{\vartheta})}{p(\boldsymbol{\vartheta}|\boldsymbol{y})}\right) d\boldsymbol{\vartheta} \\ &= \int q(\boldsymbol{\vartheta}) ln\left(\frac{q(\boldsymbol{\vartheta})p(\boldsymbol{y})}{p(\boldsymbol{\vartheta},\boldsymbol{y})}\right) d\boldsymbol{\vartheta} \\ &= \int q(\boldsymbol{\vartheta}) ln\left(\frac{q(\boldsymbol{\vartheta})}{p(\boldsymbol{\vartheta},\boldsymbol{y})}\right) d\boldsymbol{\vartheta} + ln(p(\boldsymbol{y})) \\ &= \int q(\boldsymbol{\vartheta}) ln(q(\boldsymbol{\vartheta}))d\boldsymbol{\vartheta} - \int q(\boldsymbol{\vartheta}) ln(p(\boldsymbol{\vartheta},\boldsymbol{y}))d\boldsymbol{\vartheta} + ln(p(\boldsymbol{y})) \end{aligned} \tag{2-9}$$

This expression can be rearranged as:

$$\underbrace{ln(p(\boldsymbol{y}))}_{log-evidence} = \underbrace{\int q(\boldsymbol{\vartheta})ln(p(\boldsymbol{\vartheta},\boldsymbol{y}))d\boldsymbol{\vartheta} - \int q(\boldsymbol{\vartheta})ln(q(\boldsymbol{\vartheta}))d\boldsymbol{\vartheta}}_{\substack{\mathcal{F}(\boldsymbol{\vartheta},\boldsymbol{y}) \\ ELBO \\ VFE}} + \underbrace{D_{KL}(q(\boldsymbol{\vartheta})||p(\boldsymbol{\vartheta}|\boldsymbol{y}))}_{\substack{\geq 0 \\ KL-divergence}} \tag{2-10}$$

This equation tells us that the model log-evidence $ln(p(\boldsymbol{y}))$ is lower-bounded by the FE $\mathcal{F}(\boldsymbol{\vartheta},\boldsymbol{y})$, since the KL divergence is non-negative. Therefore, another name for FE is Evidence Lower BOund (ELBO) [18]. However, to stay in words related to the FEP and to avoid confusion with the concept of FE in physics (where energy is defined opposite to energies as defined here [19]), the FE will be called Variational Free Energy (VFE), thereby following the convention in [20].

2-1-4 Intermediate summary

In summary, and to relate the mathematics to the agent representing a biological system: an agent needs to have a generative model that properly represents the generative process in order to survive. By having a good generative model, the agent is able to execute actions in such a way that it minimizes its sensory surprisal. A measure of correctness of its generative model is the model log-evidence, equal to the negative of sensory surprisal. Therefore, minimizing sensory surprisal is equivalent to maximizing model log-evidence. VFE forms a lower bound on model log-evidence, so the agent can maximize model log-evidence by maximizing VFE.

In control engineering, this corresponds to system identification. In this section it is shown that system identification can be performed in a biologically plausible way by maximizing one quantity: VFE.

The next section will show how to maximize VFE for static models. The result is then extended to hold for dynamics models as well, which can be used to derive the DEM filter equations.

2-2 Maximizing VFE for static models

Before being able to maximize VFE, we have to gain more understanding of what the VFE looks like. The integral expressions in Eq. (2-8) until (2-10) are represented as single integrals. However, following Eq. (2-5), $\boldsymbol{\vartheta}$ contains 3 types of parameters: states and inputs, model parameters and hyperparameters. Therefore, the VFE is actually defined as a triple integral expression over all parameter sets:

$$\mathcal{F}(\boldsymbol{\vartheta},\boldsymbol{y}) = \underbrace{\int\int\int q(\boldsymbol{\vartheta})ln(p(\boldsymbol{\vartheta},\boldsymbol{y}))d\boldsymbol{\vartheta_1}d\boldsymbol{\vartheta_2}d\boldsymbol{\vartheta_3}}_{G} \underbrace{- \int\int\int q(\boldsymbol{\vartheta})ln(q(\boldsymbol{\vartheta}))d\boldsymbol{\vartheta_1}d\boldsymbol{\vartheta_2}d\boldsymbol{\vartheta_3}}_{H} \tag{2-11}$$

where $\boldsymbol{\vartheta}_1 = \boldsymbol{\vartheta}_X$, $\boldsymbol{\vartheta}_2 = \boldsymbol{\vartheta}_\theta$ and $\boldsymbol{\vartheta}_3 = \boldsymbol{\vartheta}_\lambda$ denote the parameter subsets for states and inputs, model parameters and hyperparameters, respectively. Furthermore, $G(\boldsymbol{\vartheta},\boldsymbol{y})$ and $H(\boldsymbol{\vartheta})$ represent the Gibb's, or internal, energy ($U(\boldsymbol{\vartheta},\boldsymbol{y}) = ln(p(\boldsymbol{\vartheta},\boldsymbol{y}))$) expectation under the recognition

density and the recognition density entropy, respectively [4]. As we will see later, the internal energy plays an important role in maximizing the VFE.

Finding $q(\boldsymbol{\vartheta})$ that directly maximizes the VFE in Eq. (2-11) is again an intractable problem. Therefore, DEM considers 2 approximations:

- The mean-field approximation
- The Laplace approximation

Each of these approximations will be described in the following sections.

2-2-1 Mean-field approximation

As mentioned in Section 2-1-1, the parameters in the different subsets are optimized in processes that run at different temporal scales. The mean-field approximation uses this fact to factorize the recognition density into parts that belong to different subsets [4]:

$$q(\boldsymbol{\vartheta}) = \prod_{i=1}^{3} q(\boldsymbol{\vartheta}_i) \tag{2-12}$$

Using this approximation and the fundamental lemma of variational calculus, it can be proven that each $q(\boldsymbol{\vartheta}_i)$ can be written as [4]:

$$q(\boldsymbol{\vartheta}_i) = \frac{1}{Z_i} e^{V(\boldsymbol{\vartheta}_i)} \tag{2-13}$$

where Z_i is a normalization constant and $V(\boldsymbol{\vartheta}_i)$ is called variational energy.

Finding $q(\boldsymbol{\vartheta}_i)$ is still intractable, except for the case where conjugate priors are used [13]. Two alternatives exist.

The most general approach assumes a free-form approximation for $q(\boldsymbol{\vartheta}_i)$ and finds the stationary (i.e. time-invariant) solution for $q(\boldsymbol{\vartheta}_i)$ using ensemble dynamics. It turns out that the stationary solution of $q(\boldsymbol{\vartheta}_i)$ is the same as presented in Eq. (2-13). In ensemble dynamics, each particle in the ensemble corresponding to parameter subset i is moving uphill on the variational energy manifold corresponding to parameter subset i and meanwhile exposed to random fluctuations. Furthermore, the movement of all particles is influenced by the mean-field effect of the other parameter subsets. The combination of these effects and a big enough ensemble of particles ensure that the average of all particles coincides with the peak, or mode, of the variational energy manifold (and thus of the ensemble density $q(\boldsymbol{\vartheta}_i)$), while the dispersion of particles indicates the variance of $q(\boldsymbol{\vartheta}_i)$. In order to find the ensemble density mode and variance, the path of each particle in the ensemble needs to be calculated. A form of filtering (thus not including model parameter and hyperparameter estimation) that makes use of ensemble dynamics is variational filtering [21].

A simpler approach assumes a fixed-form approximation for $q(\boldsymbol{\vartheta}_i)$: a Gaussian distribution. This is called the Laplace approximation. It can be shown that a particle subject to the

same effects as before, except for the random fluctuations, will converge to the mode over time [4]. Since the variance of $q(\boldsymbol{\vartheta}_i)$ can be calculated in an exact way (as the next section will show), we can reduce the optimization problem to only finding the path of one particle without random fluctuations, instead of having to calculate the path for every particle. The details of the Laplace approximation are described in the next section.

2-2-2 Laplace approximation

The Laplace assumption entails assuming that the posterior density $p(\boldsymbol{\vartheta}|\boldsymbol{y})$ has a Gaussian shape and should therefore be approximated with a Gaussian recognition density. As a result, $q(\boldsymbol{\vartheta}_i)$ is written as:

$$q(\boldsymbol{\vartheta}_i) = \underbrace{\frac{1}{\sqrt{(2\pi)^{n_i}\,|\Sigma_i|}}}_{\frac{1}{Z_i}} \underbrace{e^{-\frac{1}{2}(\boldsymbol{\vartheta}_i-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\boldsymbol{\vartheta}_i-\boldsymbol{\mu}_i)}}_{e^{V(\boldsymbol{\vartheta}_i)}} \tag{2-14}$$

where n_i indicates the parameter subset dimension of $\boldsymbol{\vartheta}_i$.

As [22] states: "A Gaussian approximation is motivated by the fact that, in the large data limit and given some regularity conditions, the posterior approaches a Gaussian around the MAP", where 'MAP' stands for Maximum A Posteriori and indicates the mode estimation of the posterior density $p(\boldsymbol{\vartheta}|\boldsymbol{y})$.

Using the Laplace approximation, it can be shown that G can be simplified to [4]:

$$\begin{aligned} G &= \int\int\int q(\boldsymbol{\vartheta})U(\boldsymbol{\vartheta},\boldsymbol{y})d\boldsymbol{\vartheta_1}d\boldsymbol{\vartheta_2}d\boldsymbol{\vartheta_3} \\ &\approx \int\int\int q(\boldsymbol{\vartheta})\left(U(\boldsymbol{\mu},\boldsymbol{y}) + (\boldsymbol{\vartheta}-\boldsymbol{\mu})^T\left.\frac{\partial^2 U(\boldsymbol{\vartheta},\boldsymbol{y})}{\partial\boldsymbol{\vartheta}^2}\right|_{\boldsymbol{\vartheta}=\boldsymbol{\mu}}(\boldsymbol{\vartheta}-\boldsymbol{\mu})\right)d\boldsymbol{\vartheta_1}d\boldsymbol{\vartheta_2}d\boldsymbol{\vartheta_3} \\ &= U(\boldsymbol{\mu},\boldsymbol{y}) + \frac{1}{2}\sum_{i=1}^{3} tr\left(\Sigma_i\frac{\partial^2 U(\boldsymbol{\mu}_i,\boldsymbol{y})}{\partial\boldsymbol{\vartheta}_i^2}\right) \end{aligned} \tag{2-15}$$

where $\frac{\partial^2 U(\boldsymbol{\mu}_i,\boldsymbol{y})}{\partial\boldsymbol{\vartheta}_i^2}$ is the called the Hessian and is evaluated at the mode of parameter subset i.

To simplify H, we first write out $ln(q(\boldsymbol{\vartheta}))$:

$$\begin{aligned} ln(q(\boldsymbol{\vartheta})) &= ln\left(\frac{1}{Z}e^{V(\boldsymbol{\vartheta})}\right) \\ &= -ln(Z) + V(\boldsymbol{\vartheta}) \end{aligned} \tag{2-16}$$

Now, H can be simplified to [4]:

$$\begin{aligned} H &= \int\int\int q(\boldsymbol{\vartheta}) ln(q(\boldsymbol{\vartheta})) d\boldsymbol{\vartheta_1} d\boldsymbol{\vartheta_2} d\boldsymbol{\vartheta_3} \\ &= \int\int\int q(\boldsymbol{\vartheta}) \left(-ln(Z) + V(\boldsymbol{\vartheta})\right) d\boldsymbol{\vartheta_1} d\boldsymbol{\vartheta_2} d\boldsymbol{\vartheta_3} \\ &= \frac{1}{2} \sum_{i=1}^{3} \left(n_i ln(2\pi e) + ln(|\Sigma_i|)\right) \end{aligned} \tag{2-17}$$

Therefore, the VFE can be written as:

$$\mathcal{F}(\boldsymbol{\vartheta}, \boldsymbol{y}) \approx U(\boldsymbol{\mu}, \boldsymbol{y}) + \frac{1}{2} \sum_{i=1}^{3} \left(tr\left(\Sigma_i \frac{\partial^2 U(\boldsymbol{\mu}_i, \boldsymbol{y})}{\partial \boldsymbol{\vartheta}_i^2} \right) + n_i ln(2\pi e) + ln(|\Sigma_i|) \right) \tag{2-18}$$

One can find the exact optimal solution for Σ_i, Σ_i^*, by differentiating $\mathcal{F}(\boldsymbol{\vartheta}, \boldsymbol{y})$ with respect to Σ_i and setting equal to zero. The solution is found as [4]:

$$\Sigma_i^* = -\left(\frac{\partial^2 U(\boldsymbol{\mu}_i, \boldsymbol{y})}{\partial \boldsymbol{\vartheta}_i^2} \right)^{-1} \tag{2-19}$$

The VFE can now be simplified and becomes:

$$\mathcal{F}(\boldsymbol{\vartheta}, \boldsymbol{y}) \approx U(\boldsymbol{\mu}, \boldsymbol{y}) + \frac{1}{2} \sum_{i=1}^{3} \left(n_i ln(2\pi) + ln(|\Sigma_i|)\right) \tag{2-20}$$

Except for the internal energy evaluated at its mode, $U(\boldsymbol{\mu}, \boldsymbol{y})$, all quantities in this expression are constant. Therefore, maximizing VFE boils down to maximizing $U(\boldsymbol{\mu}, \boldsymbol{y})$:

$$\mathcal{F}(\boldsymbol{\vartheta}, \boldsymbol{y}) \propto U(\boldsymbol{\mu}, \boldsymbol{y}) \tag{2-21}$$

The internal energy is described in more detail in the next section.

2-2-3 Internal energy

So far, the derivation of DEM holds for state and input, model parameter and hyperparameter estimation. In the end, we are only interested in the DEM state estimation equations. We therefore assume the model parameters and hyperparameters to be constant, i.e.:

$$\boldsymbol{\vartheta} = X \tag{2-22}$$

Using this assumption, the internal energy can be written as:

$$\begin{aligned} U(X, \boldsymbol{y}) &= ln(p(X, \boldsymbol{y})) \\ &= ln(p(\boldsymbol{y}|X)p(X)) \\ &= ln(p(\boldsymbol{y}|X)) + ln(p(X)) \\ &= ln(p(\boldsymbol{y}|X)) + ln(p(\boldsymbol{x}, \boldsymbol{u})) \\ &= ln(p(\boldsymbol{y}|X)) + ln(p(\boldsymbol{x}|\boldsymbol{u})p(\boldsymbol{u})) \\ &= ln(p(\boldsymbol{y}|X)) + ln(p(\boldsymbol{x}|\boldsymbol{u})) + ln(p(\boldsymbol{u})) \end{aligned} \tag{2-23}$$

Given a static model with constant model parameters and hyperparameters, Eq. (2-3) and (2-4) become:

$$\boldsymbol{x} = \boldsymbol{f}_s(\boldsymbol{u}) + \boldsymbol{w} \quad \rightarrow \quad p(\boldsymbol{x}|\boldsymbol{u}) \tag{2-24}$$

$$\boldsymbol{y} = \boldsymbol{g}_s(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{z} \rightarrow \quad p(\boldsymbol{y}|X) \tag{2-25}$$

Of course, these equations can be merged into a single equation, but they are written this way to have the same format as used for dynamic models in Section 2-3. The distinction is made using subscript s. The terms in the equation above exactly correspond to the ones constructing part of the internal energy and will be evaluated below.

$ln(p(\boldsymbol{y}|X))$ is the likelihood derived from Eq. (2-25). Under the Laplace approximation, we can model it as a Gaussian distribution, i.e.:

$$p(\boldsymbol{y}|X) \sim \mathcal{N}(\boldsymbol{g}_s, \Sigma_z) \tag{2-26}$$

where the mode is given by output function $\boldsymbol{g}_s(\boldsymbol{x}, \boldsymbol{u})$ and the covariance is caused by the Gaussian distributed measurement noise $p(\boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{0}, \Sigma_z)$.

Similarly, $ln(p(\boldsymbol{x}|\boldsymbol{u}))$ is the likelihood from Eq. (2-24) and is also Gaussian distributed:

$$p(\boldsymbol{x}|\boldsymbol{u}) \sim \mathcal{N}(\boldsymbol{f}_s, \Sigma_w) \tag{2-27}$$

where the mode is given by state function $\boldsymbol{f}_s(\boldsymbol{u})$ and the covariance is caused by the Gaussian distributed process noise $p(\boldsymbol{w}) \sim \mathcal{N}(\boldsymbol{0}, \Sigma_w)$.

The input prior $p(\boldsymbol{u})$ also follows a Gaussian distribution and is given by:

$$p(\boldsymbol{u}) \sim \mathcal{N}(\boldsymbol{\eta}_u, C_u) \tag{2-28}$$

where $\boldsymbol{\eta}_u$ represents the expectation and C_u the covariance matrix of the input prior.

With this information, all terms in Eq. (2-23) can be written as:

$$\begin{aligned} ln(p(\boldsymbol{y}|X)) &= ln\left(\frac{1}{\sqrt{(2\pi)^{n_y}\,|\Sigma_z|}} e^{-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{g}_s)^T \Pi_z (\boldsymbol{y}-\boldsymbol{g}_s)}\right) \\ &= \frac{1}{2}\left(-n_y ln(2\pi) + ln(\Pi_z) - (\boldsymbol{y}-\boldsymbol{g}_s)^T \Pi_z (\boldsymbol{y}-\boldsymbol{g}_s)\right) \end{aligned} \tag{2-29}$$

$$\begin{aligned} ln(p(\boldsymbol{x}|\boldsymbol{u})) &= ln\left(\frac{1}{\sqrt{(2\pi)^{n_x}\,|\Sigma_w|}}e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{f}_s)^T\Pi_w(\boldsymbol{x}-\boldsymbol{f}_s)}\right) \\ &= \frac{1}{2}\left(-n_x ln(2\pi) + ln(\Pi_w) - (\boldsymbol{x}-\boldsymbol{f}_s)^T\Pi_w(\boldsymbol{x}-\boldsymbol{f}_s)\right) \end{aligned} \tag{2-30}$$

$$\begin{aligned} ln(p(\boldsymbol{u})) &= ln\left(\frac{1}{\sqrt{(2\pi)^{n_u}\,|C_u|}}e^{-\frac{1}{2}(\boldsymbol{u}-\boldsymbol{\eta}_u)^T\Pi_u(\boldsymbol{u}-\boldsymbol{\eta}_u)}\right) \\ &= \frac{1}{2}\left(-n_u ln(2\pi) + ln(\Pi_u) - (\boldsymbol{u}-\boldsymbol{\eta}_u)^T\Pi_u(\boldsymbol{u}-\boldsymbol{\eta}_u)\right) \end{aligned} \tag{2-31}$$

The internal energy is constructed by the summation of these terms. Since we are only interested in maximizing the internal energy, we can ignore the constant terms. Furthermore, we are only interested in the state estimation equations of DEM. Therefore, we can also disregard the $ln(p(\boldsymbol{u}))$ term (no uncertainty regarding the input exists) and be left with:

$$U(X,\boldsymbol{y}) \propto -\frac{1}{2}\left((\boldsymbol{y}-\boldsymbol{g}_s)^T\Pi_z(\boldsymbol{y}-\boldsymbol{g}_s) + (\boldsymbol{x}-\boldsymbol{f}_s)^T\Pi_w(\boldsymbol{x}-\boldsymbol{f}_s)\right) \tag{2-32}$$

The internal energy is thus maximized by optimizing a cost function consisting of state and output error terms, weighted by the precision (inverse covariance) matrices of process and measurement noise respectively (i.e. a quadratic, precision weighted, cost function). In shorthand notation, this expression is written as:

$$U(X,\boldsymbol{y}) \propto -\frac{1}{2}\boldsymbol{\epsilon}^T\Pi\boldsymbol{\epsilon} \tag{2-33}$$

where $\boldsymbol{\epsilon} = \begin{bmatrix}\boldsymbol{y}-\boldsymbol{g}_s \\ \boldsymbol{x}-\boldsymbol{f}_s\end{bmatrix} \in \mathbb{R}^{n_y+n_x}$ and $\Pi = \begin{bmatrix}\Pi_z & 0 \\ 0 & \Pi_w\end{bmatrix} \in \mathbb{R}^{(n_y+n_x)\times(n_y+n_x)}$.

As mentioned before, this expression should be evaluated at its mode. The internal energy mode equals the current state value $\boldsymbol{x}$. By iteratively optimizing these quantities to find the variational energy that maximizes VFE, the mode starts converging towards its true value. Given the fact that a single particle (corresponding to the current mode) will converge to the true mode in a finite amount of time, the true values for $\boldsymbol{x}$ can be found in a finite amount of time, which is exactly the purpose of state estimation.

2-2-4 Intermediate summary

In the previous sections, we have seen that an agent should maximize its VFE in order to end up in generative process states in which it can survive. This section has shown that the VFE is maximized by finding the correct form of the recognition density for each parameter subset $q(\boldsymbol{\vartheta}_i)$, given in Eq. (2-13), under the mean-field approximation. Using the Laplace approximation, maximizing VFE boils down to maximizing the internal energy. The internal energy was defined by assuming constant model parameters and hyperparameters. Maximizing internal energy then turned into an optimization problem of maximizing a quadratic, precision weighted, cost function involving the state and output error and process and measurement

noise precision matrices. It is shown that by maximizing the internal energy, we can estimate the system states. Therefore, we have derived a biologically plausible implementation of state estimation.

The result in this section holds for static models. However, a quadrotor is a dynamic system. Therefore, the result is extended to hold for dynamic models as well in the next section.

2-3 Maximizing VFE for dynamic models

Maximizing VFE for dynamic models is very similar to maximizing VFE for static models. The mean-field and Laplace approximation still apply. The same holds for the assumption of constant model parameters and hyperparameters. However, there are a few conceptual differences. First of all, the states vary over time in a dynamic model. As a result, the optimal variational energy mode also changes over time. Therefore, generalized coordinates of motion are introduced. Section 2-3-1 describes the adjusted generative model in which each quantity is replaced by its generalized version. Secondly, generalized coordinates of motion imply that both process and measurement noise should be differentiable. As a result, they cannot be white, but should be coloured instead. Details on coloured noise are given in Section 2-3-2. Finally, maximizing VFE for dynamic models also boils down to maximizing the internal energy. Therefore, Section 2-3-3 will derive the internal energy for dynamic models in a similar way as for the static case.

2-3-1 Adjusted generative model

From now on, it is assumed that the generative process can be described using an LTI generative model in order to simplify filter analysis. This assumption turns the dynamic state and output equations of Eq. (2-1) and (2-2) into the following form:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} + \boldsymbol{w} \tag{2-34}$$

$$\boldsymbol{y} = C\boldsymbol{x} + \boldsymbol{z} \tag{2-35}$$

This dynamic model describes varying states, inputs and outputs over time. This implies that the posterior density also changes over time. As a result, the recognition density and the variational energy should vary over time. Therefore, instead of having particles on a constant variational energy manifold, we now have to deal with particles on a moving variational energy manifold. Under the mean-field approximation, Section 2-2-1 showed that one particle climbing uphill on the variational energy manifold and being influenced by mean-field effects of the model parameters and hyperparameters is able to reach the variational energy mode in a finite amount of time, thereby maximizing VFE. It can be shown using linear stability theory that one particle is also able to track the mode of a time-varying variational energy manifold once it knows how the manifold is changing over time (i.e. it knows its derivatives up to a limited order) [4].

Given this result, we are interested in the derivative information of the quantities in the dynamic model, including states, inputs, outputs, process noises and measurement noises.

The dynamic model can be extended to include the derivatives of these quantities in the form of generalized coordinates of motion. For example, the generalized states vector is now given by:

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x}' \\ \boldsymbol{x}'' \\ \vdots \\ \boldsymbol{x}^{(p)} \end{bmatrix} \tag{2-36}$$

where $\boldsymbol{x}'$, $\boldsymbol{x}''$ and $\boldsymbol{x}^{(p)}$ indicate the 1st-, 2nd- and pth-order derivatives of $\boldsymbol{x}$, respectively. $p \in \mathbb{Z}$ is called the state embedding order. In the DEM filter, a different embedding order is assigned to the inputs. The input embedding order is denoted by $d \in \mathbb{Z}$. [4] concludes that taking care of derivatives up to and including order $p = 6$ and $d = 2$ still provide sufficient information, whereas the amount of information drops beyond these embedding orders. This statement is confirmed in Section 7-2.

Rewriting Eq. (2-1) and (2-2) in generalized coordinates with embedding orders $d = p - 1$ (to provide an example for different embedding orders) gives [4]:

$$\begin{aligned} \boldsymbol{x}' &= \boldsymbol{f} + \boldsymbol{w} \\ \boldsymbol{x}'' &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{x}' + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{u}' + \boldsymbol{w}' \\ &\vdots \\ \boldsymbol{x}^{(p-1)} &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{x}^{(p-2)} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{u}^{(d-1)} + \boldsymbol{w}^{(p-2)} \\ \boldsymbol{x}^{p} &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{x}^{(p-1)} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{u}^{d} + \boldsymbol{w}^{(p-1)} \end{aligned} \tag{2-37}$$

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{g} + \boldsymbol{z} \\ \boldsymbol{y}' &= \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\boldsymbol{x}' + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}\boldsymbol{u}' + \boldsymbol{z}' \\ &\vdots \\ \boldsymbol{y}^{(p-1)} &= \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\boldsymbol{x}^{(p-1)} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}\boldsymbol{u}^{d} + \boldsymbol{z}^{(p-1)} \\ \boldsymbol{y}^{p} &= \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\boldsymbol{x}^{p} + \boldsymbol{z}^{p} \end{aligned} \tag{2-38}$$

where $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} = 0$ as there is no direct coupling between input and output.

The generalized LTI state-space model (with $d = p - 1$) is thus given by:

$$\underbrace{\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{x}'' \\ \vdots \\ \boldsymbol{x}^{(p)} \\ 0 \end{bmatrix}}_{\mathcal{D}\tilde{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} A & 0 & \dots & 0 & 0 \\ 0 & A & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & A & 0 \\ 0 & 0 & \dots & 0 & A \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x}' \\ \vdots \\ \boldsymbol{x}^{(p-1)} \\ \boldsymbol{x}^{(p)} \end{bmatrix}}_{\tilde{\boldsymbol{x}}} + \underbrace{\begin{bmatrix} B & 0 & \dots & 0 \\ 0 & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B \\ 0 & 0 & \dots & 0 \end{bmatrix}}_{\tilde{B}} \underbrace{\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{u}' \\ \vdots \\ \boldsymbol{u}^{(d-1)} \\ \boldsymbol{u}^{(d)} \end{bmatrix}}_{\tilde{\boldsymbol{u}}} + \underbrace{\begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{w}' \\ \vdots \\ \boldsymbol{w}^{(p-1)} \\ \boldsymbol{w}^{(p)} \end{bmatrix}}_{\tilde{\boldsymbol{w}}} \tag{2-39}$$

$$\underbrace{\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}' \\ \vdots \\ \boldsymbol{y}^{(p-1)} \\ \boldsymbol{y}^{(p)} \end{bmatrix}}_{\tilde{\boldsymbol{y}}} = \underbrace{\begin{bmatrix} C & 0 & \dots & 0 & 0 \\ 0 & C & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & C & 0 \\ 0 & 0 & \dots & 0 & C \end{bmatrix}}_{\tilde{C}} \underbrace{\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x}' \\ \vdots \\ \boldsymbol{x}^{(p-1)} \\ \boldsymbol{x}^{(p)} \end{bmatrix}}_{\tilde{\boldsymbol{x}}} + \underbrace{\begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{z}' \\ \vdots \\ \boldsymbol{z}^{(p-1)} \\ \boldsymbol{z}^{(p)} \end{bmatrix}}_{\tilde{\boldsymbol{z}}} \tag{2-40}$$

where $\tilde{\boldsymbol{x}} \in \mathbb{R}^{n_x \cdot (p+1)}$, $\tilde{\boldsymbol{u}} \in \mathbb{R}^{n_u \cdot (d+1)}$, $\tilde{\boldsymbol{y}} \in \mathbb{R}^{n_y \cdot (p+1)}$, $\tilde{\boldsymbol{w}} \in \mathbb{R}^{n_x \cdot (p+1)}$ and $\tilde{\boldsymbol{z}} \in \mathbb{R}^{n_y \cdot (p+1)}$ are the generalized states, inputs, outputs, process noises and measurement noises, respectively. $\tilde{A} \in \mathbb{R}^{(n_x \cdot (p+1)) \times (n_x \cdot (p+1))}$, $\tilde{B} \in \mathbb{R}^{(n_x \cdot (p+1)) \times (n_u \cdot (d+1))}$ and $\tilde{C} \in \mathbb{R}^{(n_y \cdot (p+1)) \times (n_x \cdot (p+1))}$ represent the generalized state, input and output matrices. $\mathcal{D} \in \mathbb{R}^{(n_x \cdot (p+1)) \times (n_x \cdot (p+1))}$ denotes a derivative operator that implements the derivative operation in the original dynamic model with a simple shift of higher-order derivatives. It is defined by:

$$\mathcal{D} = \begin{bmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix} \otimes I_{n_x} \tag{2-41}$$

where $\otimes$ represents the Kronecker tensor product.

In order to make Eq. (2-39) and (2-40) physically realistic, the states, inputs, outputs and noises should be differentiable. Since every physical system has a finite eigenfrequency and inputs given by a controller have a finite frequency too, states, inputs and outputs are differentiable by definition. The generalized inputs and outputs are constructed using an inverse Taylor expansion. For example, for $\boldsymbol{y}$ this yields [4]:

$$\begin{aligned}\tilde{\boldsymbol{y}}_t &= \tilde{E}^{-1}\begin{bmatrix}\boldsymbol{y}_{t-\lceil\frac{p}{2}\rceil\Delta t}\\ \vdots \\ \boldsymbol{y}_t \\ \vdots \\ \boldsymbol{y}_{t+\lfloor\frac{p}{2}\rfloor\Delta t}\end{bmatrix} \\ \tilde{E} &= E \otimes I_{n_y} \\ E_{ij} &= \frac{\left((i-\frac{p+1}{2}\Delta t)\right)^{(j-1)}}{(j-1)!}\end{aligned} \tag{2-42}$$

where $\boldsymbol{y}_{t-\Delta t}$, $\boldsymbol{y}_t$ and $\boldsymbol{y}_{t+\Delta t}$ indicate the system output samples at the previous, current and next time instance, respectively, and E is called the finite differences matrix. The exact theory behind generating generalized coordinates of motion is beyond the scope of this thesis. For more information, please consult [23].

Using the generalized inputs and outputs, the generalized state is estimated. In a lot of existing filters, the process and measurement noise are assumed to be white. In a white noise signal, each sample is independent from the previous sample and therefore the noise is not differentiable by definition. Instead of white noise, DEM assumes the noises to be coloured, such that the noise derivatives can be constructed. The next section will elaborate on the definition of coloured noise and how it is characterized.

2-3-2 Coloured noise

As indicated in [4], coloured noise can be constructed by filtering a Gaussian distributed white noise signal with a Gaussian filter[1]. For example, for a 1D process noise signal this yields:

$$w = \omega_w * h_w(t) \tag{2-43}$$

where ω_w is the white noise signal being convoluted with the Gaussian filter $h_w(t)$. In general, a Gaussian filter is given by the following equation [24]:

$$h(t) = Ke^{-\frac{t^2}{2\sigma_h^2}} \tag{2-44}$$

where K is a scaling factor, t denotes the time instance and σ_h represents the Gaussian standard deviation (also called kernel width). In this case, K can be determined by constraining the coloured noise to have the same variance as the original white noise out of which it is constructed. The resulting Gaussian filter becomes [24]:

$$h(t) = \sqrt{\frac{\Delta t}{\sigma_h\sqrt{\pi}}}e^{-\frac{t^2}{2\sigma_h^2}} \tag{2-45}$$

[1]It should be noted that the Gaussian filter is only a method to generate coloured noise. Other methods may also exist, but are not considered in this thesis. Section 6-3 will discuss the Gaussian filter assumption.

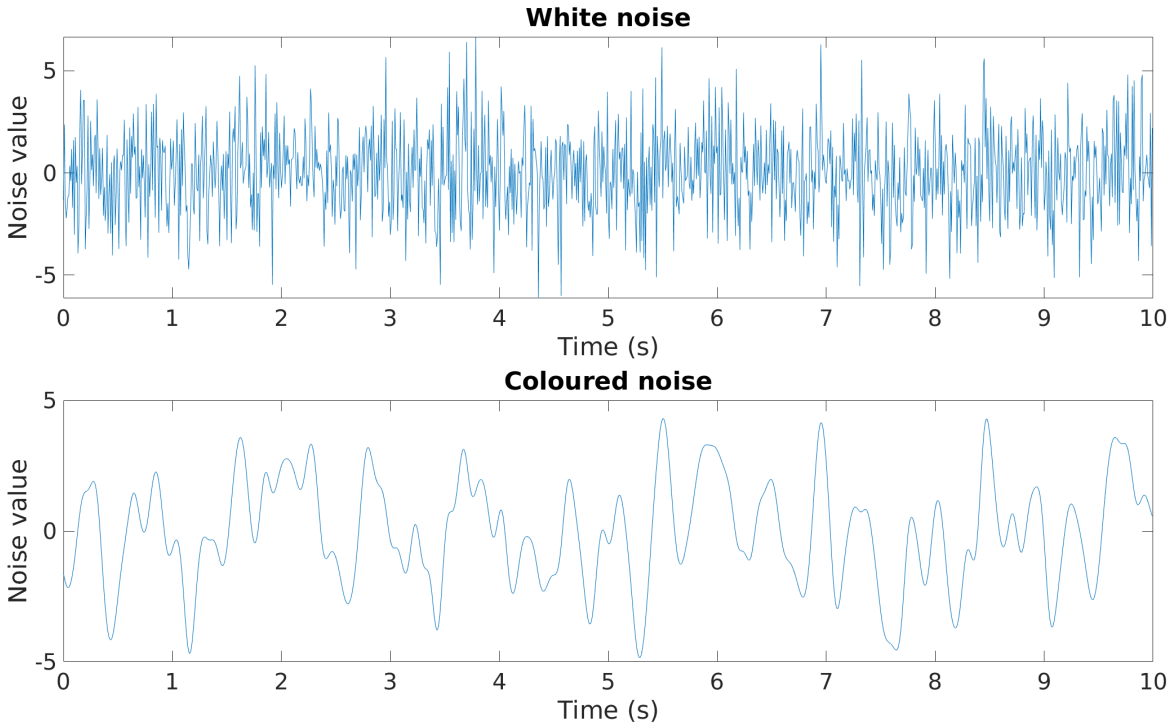


Figure 2-1: Gaussian distributed white noise with $\sigma_\omega^2 = 4$ and coloured noise with $\sigma_\omega^2 = 4$ and $s = 5 \cdot 10^{-2}$ s.

Figure 2-1 shows a white noise signal and the result of the convolution of this white noise signal with a Gaussian filter. The figure clearly shows that the white noise signal contains random behaviour, while the coloured noise signal contains structure. Due to this structure, we are able to take derivatives of the noise, which will be leveraged by the DEM filter.

Noises with structure are also called smooth noises. The so-called smoothness of these noises is denoted by s. Logically, the higher s, the more structure the noise contains. In case of the Gaussian filter, it holds that $s = \sigma_h$. The smoothness turns out to be an important noise parameter when considering the covariance matrix of generalized noise signals (called the generalized covariance matrix). The generalized covariance matrix for generalized process noise $\tilde{\boldsymbol{w}}$ with embedding order p, for example, is given by [23], [24]:

$$\tilde{\Sigma}_w = \begin{bmatrix} C(w_1, w_1) & \dots & C(w_1, w_{n_x}) & & C(w_1, w_1^{(p)}) & \dots & C(w_1, w_{n_x}^{(p)}) \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ C(w_{n_x}, w_1) & \dots & C(w_{n_x}, w_{n_x}) & & C(w_{n_x}, w_1^{(p)}) & \dots & C(w_{n_x}, w_{n_x}^{(p)}) \\ & \vdots & & \ddots & & \vdots & \\ C(w_1^{(p)}, w_1) & \dots & C(w_1^{(p)}, w_{n_x}) & & C(w_1^{(p)}, w_1^{(p)}) & \dots & C(w_1^{(p)}, w_{n_x}^{(p)}) \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ C(w_{n_x}^{(p)}, w_1) & \dots & C(w_{n_x}^{(p)}, w_{n_x}) & & C(w_{n_x}^{(p)}, w_1^{(p)}) & \dots & C(w_{n_x}^{(p)}, w_{n_x}^{(p)}) \end{bmatrix} \tag{2-46}$$

where $C(a, b)$ denotes the covariance of a and b.

Under the assumption that coloured noise is generated using white noise and a Gaussian filter, it turns out that noise derivatives are correlated to each other as a function of the Gaussian filter autocorrelation, evaluated as zero. As a result, the generalized process noise covariance matrix can be written in terms of s_w (assuming $p = 6$) as [23]:

$$\begin{aligned}\tilde{\Sigma}_w &= S(s_w^2) \otimes \Sigma_w \\ &= \begin{bmatrix} 1 & 0 & -\frac{1}{2s_w^2} & 0 & \frac{3}{(2s_w^2)^2} & 0 & -\frac{15}{(2s_w^2)^3} \\ 0 & \frac{1}{2s_w^2} & 0 & -\frac{3}{(2s_w^2)^2} & 0 & \frac{15}{(2s_w^2)^3} & 0 \\ -\frac{1}{2s_w^2} & 0 & \frac{3}{(2s_w^2)^2} & 0 & -\frac{15}{(2s_w^2)^3} & 0 & \frac{105}{(2s_w^2)^4} \\ 0 & -\frac{3}{(2s_w^2)^2} & 0 & \frac{15}{(2s_w^2)^3} & 0 & -\frac{105}{(2s_w^2)^4} & 0 \\ \frac{3}{(2s_w^2)^2} & 0 & -\frac{15}{(2s_w^2)^3} & 0 & \frac{105}{(2s_w^2)^4} & 0 & -\frac{945}{(2s_w^2)^5} \\ 0 & \frac{15}{(2s_w^2)^3} & 0 & -\frac{105}{(2s_w^2)^4} & 0 & \frac{945}{(2s_w^2)^5} & 0 \\ -\frac{15}{(2s_w^2)^3} & 0 & \frac{105}{(2s_w^2)^4} & 0 & -\frac{945}{(2s_w^2)^5} & 0 & \frac{10{,}395}{(2s_w)^6} \end{bmatrix} \otimes \Sigma_w \end{aligned} \tag{2-47}$$

where $S(s_w^2)$ is called the temporal correlation matrix [4] and Σ_w is the process noise covariance matrix.

Similarly, the generalized process noise precision matrix can be calculated as:

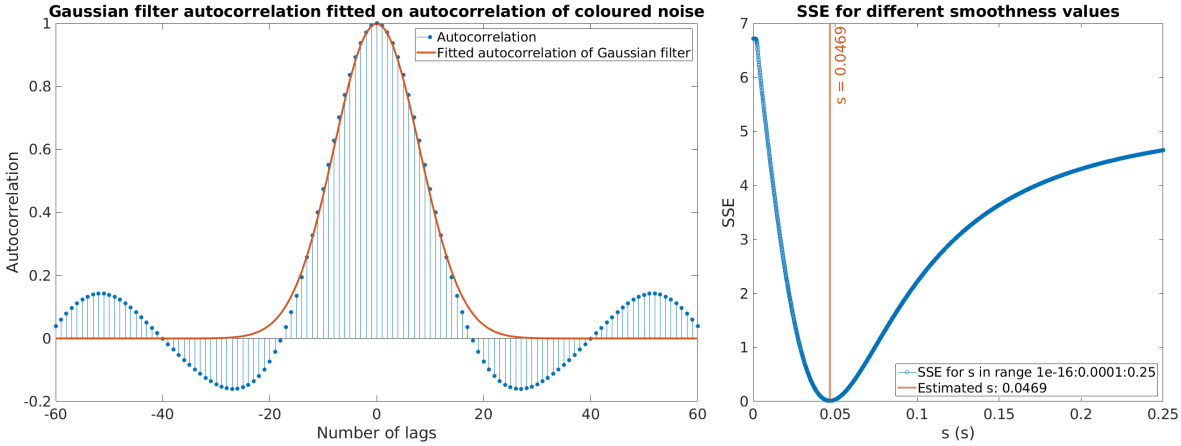
$$\begin{aligned}\tilde{\Pi}_w &= \tilde{\Sigma}_w^{-1} \\ &= \left(S(s_w^2) \otimes \Sigma_w\right)^{-1} \\ &= S(s_w^2)^{-1} \otimes \Sigma_w^{-1} \\ &= S(s_w^2)^{-1} \otimes \Pi_w \end{aligned} \tag{2-48}$$

As shown in Section 2-2-3, the process and measurement noise covariance matrices are needed to be able to maximize VFE. If process and measurement noises are coloured, a smoothness value is required besides the covariance matrices.

A method to derive the smoothness value of a coloured noise signal is to fit the autocorrelation of a Gaussian filter to the noise autocorrelation. In theory, a white noise signal does not contain correlation. Therefore, its autocorrelation is given by a delta function. Given this fact and the fact that the autocorrelation of a convolution equals the convolution of the autocorrelations, the autocorrelation of the coloured noise signal, generated by convoluting a white noise signal with a Gaussian filter, is equal to the Gaussian filter autocorrelation. In practice, the white noise autocorrelation is not an ideal delta function, causing the generated coloured noise autocorrelation to not exactly equal the Gaussian filter autocorrelation (as Figure 2-2a will show). However, fitting the Gaussian filter autocorrelation to the coloured noise autocorrelation peak around zero time lag gives an indication of the local noise smoothness leveraged by the DEM filter. An example of determining the smoothness of a coloured noise signal is given below.

The autocorrelation of a Gaussian filter is given by [24]:

$$\rho_h(\tau) = e^{-\frac{\tau^2}{2 \cdot 2\sigma_h^2}} = e^{-\frac{\tau^2}{2\sigma_\rho^2}} \tag{2-49}$$



(a) Autocorrelation **(b)** SSE

Figure 2-2: The Gaussian filter of which the autocorrelation best fits the autocorrelation of the coloured noise in terms of SSE has kernel width $s = 4.69 \cdot 10^{-2}$ s. This value is very close to the kernel width of the Gaussian filter that was used to generate the coloured noise signal, having a value of $s = 5 \cdot 10^{-2}$ s.

Therefore, the kernel width of the Gaussian filter, or smoothness, is related to the Gaussian autocorrelation via:

$$\begin{aligned} 2 \cdot 2\sigma_h^2 &= 2\sigma_\rho^2 \\ \sigma_h^2 &= \frac{\sigma_\rho^2}{2} \\ \sigma_h &= \frac{\sigma_\rho}{\sqrt{2}} \end{aligned} \tag{2-50}$$

The smoothness value for a noise signal can thus be obtained by dividing the kernel width of the best Gaussian autocorrelation fit by $\sqrt{2}$.

Figure 2-2a shows the autocorrelation of the coloured noise signal displayed in Figure 2-1, together with its best Gaussian autocorrelation fit in terms of Sum of Squares Error (SSE). Furthermore, Figure 2-2b shows the Gaussian filter smoothness value corresponding to the best Gaussian autocorrelation fit. The purpose of these figures it to show that it is possible to derive the kernel width, or smoothness, of the Gaussian filter, using which the randomly generated coloured noise signal is created, by fitting a Gaussian autocorrelation to the coloured noise autocorrelation and dividing its kernel width by $\sqrt{2}$.

Coloured noises can thus be characterized by a covariance matrix and a smoothness value. Chapter 6 will analyze the noises of the experiment data and derive the corresponding covariance matrices and smoothness values.

It should be noted that DEM uses one smoothness value for all noises [4]. Chapter 7 will show for which smoothness values the DEM filter performs well. Section 6-3 will compare these values to the ones obtained in Section 6-2-1 and 6-2-2 to see whether the above mentioned method works properly with experiment data (i.e. whether the Gaussian filter assumption is useful for deriving the noise smoothness value).

2-3-3 Internal energy

The goal of this section is to construct the internal energy for dynamic models. This can be achieved in a very similar way to presented in Section 2-2-3.

Previously, the internal energy was defined as $U(X, \boldsymbol{y}) = ln(p(X, \boldsymbol{y}))$. For the dynamic case, we have to replace X and $\boldsymbol{y}$ by their generalized versions. By following the same derivation as in Eq. (2-23), the internal energy for dynamic models becomes:

$$U(\tilde{X}, \tilde{\boldsymbol{y}}) = ln(p(\tilde{\boldsymbol{y}}|\tilde{X})) + ln(p(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{u}})) + ln(p(\tilde{\boldsymbol{u}})) \tag{2-51}$$

Using the dynamic model definition from Eq. (2-39) and (2-40), we can redefine Eq. (2-3) and (2-4) for a dynamic model:

$$\mathcal{D}\tilde{\boldsymbol{x}} = \tilde{A}\tilde{\boldsymbol{x}} + \tilde{B}\tilde{\boldsymbol{u}} + \tilde{\boldsymbol{w}} \quad \rightarrow \quad p(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{u}}) \tag{2-52}$$

$$\tilde{\boldsymbol{y}} = \tilde{C}\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{z}} \quad \rightarrow \quad p(\tilde{\boldsymbol{y}}|\tilde{X}) \tag{2-53}$$

Similar to the PDFs in Section 2-2-3, the PDFs in this section are also Gaussian distributed and therefore given by:

$$p(\tilde{\boldsymbol{y}}|\tilde{X}) \sim \mathcal{N}(\tilde{C}\tilde{\boldsymbol{x}}, \tilde{\Sigma}_z) \tag{2-54}$$

$$p(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{u}}) \sim \mathcal{N}(\tilde{A}\tilde{\boldsymbol{x}} + \tilde{B}\tilde{\boldsymbol{u}}, \tilde{\Sigma}_w) \tag{2-55}$$

$$p(\tilde{\boldsymbol{u}}) \sim \mathcal{N}(\tilde{\boldsymbol{\eta}}_u, \tilde{C}_u) \tag{2-56}$$

The uncertainties in state and output equation, $\tilde{\Sigma}_w$ and $\tilde{\Sigma}_z$, follow the structure outlined in Eq. (2-47). They are caused by the generalized Gaussian distributed process noise $p(\tilde{\boldsymbol{w}}) \sim \mathcal{N}(\boldsymbol{0}, \tilde{\Sigma}_w)$ and measurement noise $p(\tilde{\boldsymbol{z}}) \sim \mathcal{N}(\boldsymbol{0}, \tilde{\Sigma}_z)$, respectively. Furthermore, $\tilde{\boldsymbol{\eta}}_u$ and $\tilde{C}_u$ represent the generalized input expectation and covariance matrix.

The internal energy for dynamic models can now be written as:

$$\begin{aligned} ln(p(\tilde{\boldsymbol{y}}|\tilde{X})) &= ln\left(\frac{1}{\sqrt{(2\pi)^{n_y\cdot(p+1)}\left|\tilde{\Sigma}_z\right|}} e^{-\frac{1}{2}(\tilde{\boldsymbol{y}}-\tilde{C}\tilde{\boldsymbol{x}})^T\tilde{\Pi}_z(\tilde{\boldsymbol{y}}-\tilde{C}\tilde{\boldsymbol{x}})}\right) \\ &= \frac{1}{2}\left(-\left(n_y\cdot(p+1)\right)ln(2\pi) + ln(\tilde{\Pi}_z) - (\tilde{\boldsymbol{y}}-\tilde{C}\tilde{\boldsymbol{x}})^T\tilde{\Pi}_z(\tilde{\boldsymbol{y}}-\tilde{C}\tilde{\boldsymbol{x}})\right) \end{aligned} \tag{2-57}$$

$$\begin{aligned} ln(p(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{u}})) &= ln\left(\frac{1}{\sqrt{(2\pi)^{n_x\cdot(p+1)}\left|\tilde{\Sigma}_w\right|}} e^{-\frac{1}{2}(\mathcal{D}\tilde{\boldsymbol{x}}-\tilde{A}\tilde{\boldsymbol{x}}-\tilde{B}\tilde{\boldsymbol{u}})^T\tilde{\Pi}_w(\mathcal{D}\tilde{\boldsymbol{x}}-\tilde{A}\tilde{\boldsymbol{x}}-\tilde{B}\tilde{\boldsymbol{u}})}\right) \\ &= \frac{1}{2}\left(-\left(n_x\cdot(p+1)\right)ln(2\pi) + ln(\tilde{\Pi}_w) - (\mathcal{D}\tilde{\boldsymbol{x}}-\tilde{A}\tilde{\boldsymbol{x}}-\tilde{B}\tilde{\boldsymbol{u}})^T\tilde{\Pi}_w(\mathcal{D}\tilde{\boldsymbol{x}}-\tilde{A}\tilde{\boldsymbol{x}}-\tilde{B}\tilde{\boldsymbol{u}})\right) \end{aligned} \tag{2-58}$$

$$\begin{aligned} ln(p(\tilde{\boldsymbol{u}})) &= ln\left(\frac{1}{\sqrt{(2\pi)^{n_u\cdot(d+1)}\left|\tilde{C}_u\right|}}e^{-\frac{1}{2}(\tilde{\boldsymbol{u}}-\tilde{\boldsymbol{\eta}}_u)^T\tilde{\Pi}_u(\tilde{\boldsymbol{u}}-\tilde{\boldsymbol{\eta}}_u)}\right) \\ &= \frac{1}{2}\Big(-(n_u\cdot(d+1))\,ln(2\pi)+ln(\tilde{\Pi}_u)-(\tilde{\boldsymbol{u}}-\tilde{\boldsymbol{\eta}}_u)^T\tilde{\Pi}_u(\tilde{\boldsymbol{u}}-\tilde{\boldsymbol{\eta}}_u)\Big) \end{aligned} \tag{2-59}$$

Again, we are only interested in state estimation, not input estimation. Therefore, we can define the part of the internal energy that can be maximized as:

$$U(\tilde{X},\tilde{\boldsymbol{y}}) \propto -\frac{1}{2}\Big((\tilde{\boldsymbol{y}}-\tilde{C}\tilde{\boldsymbol{x}})^T\tilde{\Pi}_z(\tilde{\boldsymbol{y}}-\tilde{C}\tilde{\boldsymbol{x}})+(\mathcal{D}\tilde{\boldsymbol{x}}-\tilde{A}\tilde{\boldsymbol{x}}-\tilde{B}\tilde{\boldsymbol{u}})^T\tilde{\Pi}_w(\mathcal{D}\tilde{\boldsymbol{x}}-\tilde{A}\tilde{\boldsymbol{x}}-\tilde{B}\tilde{\boldsymbol{u}})\Big) \tag{2-60}$$

In shorthand notation given by:

$$U(\tilde{X},\tilde{\boldsymbol{y}}) \propto -\frac{1}{2}\tilde{\boldsymbol{\epsilon}}^T\tilde{\Pi}\tilde{\boldsymbol{\epsilon}} \tag{2-61}$$

where $\tilde{\boldsymbol{\epsilon}} = \begin{bmatrix} \tilde{\boldsymbol{y}}-\tilde{C}\tilde{\boldsymbol{x}} \\ \mathcal{D}\tilde{\boldsymbol{x}}-\tilde{A}\tilde{\boldsymbol{x}}-\tilde{B}\tilde{\boldsymbol{u}} \end{bmatrix} \in \mathbb{R}^{(n_y+n_x)\cdot(p+1)}$ and $\tilde{\Pi} = \begin{bmatrix} \tilde{\Pi}_z & 0 \\ 0 & \tilde{\Pi}_w \end{bmatrix} \in \mathbb{R}^{(n_y+n_x)\cdot(p+1)\times(n_y+n_x)\cdot(p+1)}$.

Again, this expression should be evaluated at its mode. In this case, the internal energy mode equals the current generalized state value $\tilde{\boldsymbol{x}}$. Using generalized coordinates of motion, we can make sure that a single particle is able to track the mode of the time-varying variational energy manifold in finite time. We can thus find the true values for $\tilde{\boldsymbol{x}}$ in a finite amount of time, meaning that we are able to perform generalized state estimation.

2-3-4 Intermediate summary

Section 2-2 has shown how we can implement state estimation in a biologically plausible way. This section has extended the result for dynamic models. In dynamic models, system quantities change over time and so does the VFE. Whereas in the static it was shown that a particle is able to converge to the variational energy mode in finite time, it does not hold true in the dynamic case. Therefore, we need to add extra information. By including the variational energy manifold movement information and letting the particle follow this movement, the particle is actually able to find the mode in finite time.

The movement information is given by the derivatives of system quantities. Therefore, it is assumed that all system signals are smooth, such that we are able to calculate these derivatives. This should also hold for the process and measurement noises. Whereas most filtering techniques assume these noises to be white, DEM assumes them to be coloured. Section 2-3-2 has given the characteristics of coloured noises: the covariance matrix and smoothness value. This information is used for deriving the internal energy for dynamic models. It turns out that maximizing the internal energy also maximizes the VFE for dynamic models. The only difference in internal energy definition with respect to the static case, is the replacement of quantities by their generalized version. As a result, maximizing internal

energy for dynamic models entails maximizing a generalized, quadratic, precision weighted, error cost function.

The next section shows that this cost function can be maximized using gradient ascent to derive the DEM filter equation.

2-4 DEM filter

The goal of this section is to arrive at a state update equation that can be implemented in order to determine the performance of the DEM filter using experiment data. This is done by following a similar reasoning as presented in [6].

Since the actual state, including its derivatives, is unknown by the filter, $\tilde{\boldsymbol{x}}$ is replaced by its estimate, $\hat{\tilde{\boldsymbol{x}}}$, in this section.

As can be derived from the previous section, the state update can be written as:

$$\dot{\hat{\tilde{\boldsymbol{x}}}} = \kappa \frac{\partial V(t)}{\partial \hat{\tilde{\boldsymbol{x}}}} + \mathcal{D}\hat{\tilde{\boldsymbol{x}}} \tag{2-62}$$

where $\dot{\hat{\tilde{\boldsymbol{x}}}}$ indicates the time-derivative of the estimate of the generalized state. $\kappa \frac{\partial V(t)}{\partial \hat{\tilde{\boldsymbol{x}}}}$ represents the gradient ascent optimization algorithm with learning rate κ and gradient $\frac{\partial V(t)}{\partial \hat{\tilde{\boldsymbol{x}}}}$, applied to the variational energy manifold. The variational energy is written as $V(t)$ to indicate that the it depends on time. $\mathcal{D}\hat{\tilde{\boldsymbol{x}}}$ denotes the movement of the states in order to follow their changing behaviour over time.

It turns out that $\frac{\partial V(t)}{\partial \hat{\tilde{\boldsymbol{x}}}}$ is equal to the state-derivative of the cost function defined using internal energy [4]:

$$\frac{\partial V(t)}{\partial \hat{\tilde{\boldsymbol{x}}}} = -\frac{\partial \tilde{\boldsymbol{\epsilon}}(t)^T}{\partial \hat{\tilde{\boldsymbol{x}}}} \tilde{\Pi} \tilde{\boldsymbol{\epsilon}}(t) \tag{2-63}$$

where $\tilde{\boldsymbol{\epsilon}}(t) \triangleq \begin{bmatrix} \tilde{\boldsymbol{\epsilon}}_y \\ \tilde{\boldsymbol{\epsilon}}_x \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{y}} - \tilde{C}\hat{\tilde{\boldsymbol{x}}} \\ \mathcal{D}\hat{\tilde{\boldsymbol{x}}} - \tilde{A}\hat{\tilde{\boldsymbol{x}}} - \tilde{B}\tilde{\boldsymbol{u}} \end{bmatrix}$ and $\frac{\partial \tilde{\boldsymbol{\epsilon}}(t)^T}{\partial \hat{\tilde{\boldsymbol{x}}}} = \begin{bmatrix} \frac{\partial \tilde{\boldsymbol{\epsilon}}_y}{\partial \hat{\tilde{\boldsymbol{x}}}} \\ \frac{\partial \tilde{\boldsymbol{\epsilon}}_x}{\partial \hat{\tilde{\boldsymbol{x}}}} \end{bmatrix}^T = \begin{bmatrix} -\tilde{C} \\ \mathcal{D} - \tilde{A} \end{bmatrix}^T$.

Writing out the gradient gives:

$$\begin{aligned}
\frac{\partial V(t)}{\partial \hat{\tilde{\boldsymbol{x}}}} &= -\frac{\partial \tilde{\boldsymbol{\epsilon}}^T}{\partial \hat{\tilde{\boldsymbol{x}}}} \tilde{\Pi} \tilde{\boldsymbol{\epsilon}} \\
&= -\begin{bmatrix} -\tilde{C} \\ \mathcal{D} - \tilde{A} \end{bmatrix}^T \begin{bmatrix} \tilde{\Pi}_z & 0 \\ 0 & \tilde{\Pi}_w \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{y}} - \tilde{C}\hat{\tilde{\boldsymbol{x}}} \\ \mathcal{D}\hat{\tilde{\boldsymbol{x}}} - \tilde{A}\hat{\tilde{\boldsymbol{x}}} - \tilde{B}\tilde{\boldsymbol{u}} \end{bmatrix} \\
&= -\begin{bmatrix} -\tilde{C}^T & (\mathcal{D} - \tilde{A})^T \end{bmatrix} \begin{bmatrix} \tilde{\Pi}_z & 0 \\ 0 & \tilde{\Pi}_w \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{y}} - \tilde{C}\hat{\tilde{\boldsymbol{x}}} \\ \mathcal{D}\hat{\tilde{\boldsymbol{x}}} - \tilde{A}\hat{\tilde{\boldsymbol{x}}} - \tilde{B}\tilde{\boldsymbol{u}} \end{bmatrix} \\
&= -\begin{bmatrix} -\tilde{C}^T\tilde{\Pi}_z & (\mathcal{D} - \tilde{A})^T\tilde{\Pi}_w \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{y}} - \tilde{C}\hat{\tilde{\boldsymbol{x}}} \\ \mathcal{D}\hat{\tilde{\boldsymbol{x}}} - \tilde{A}\hat{\tilde{\boldsymbol{x}}} - \tilde{B}\tilde{\boldsymbol{u}} \end{bmatrix} \\
&= \begin{bmatrix} \tilde{C}^T\tilde{\Pi}_z & -(\mathcal{D} - \tilde{A})^T\tilde{\Pi}_w \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{y}} - \tilde{C}\hat{\tilde{\boldsymbol{x}}} \\ (\mathcal{D} - \tilde{A})\hat{\tilde{\boldsymbol{x}}} - \tilde{B}\tilde{\boldsymbol{u}} \end{bmatrix} \\
&= \left(-\tilde{C}^T\tilde{\Pi}_z\tilde{C} - (\mathcal{D} - \tilde{A})^T\tilde{\Pi}_w(\mathcal{D} - \tilde{A})\right)\hat{\tilde{\boldsymbol{x}}} + \tilde{C}^T\tilde{\Pi}_z\tilde{\boldsymbol{y}} + (\mathcal{D} - \tilde{A})^T\tilde{\Pi}_w\tilde{B}\tilde{\boldsymbol{u}}
\end{aligned} \tag{2-64}$$

Using the final expression for $\frac{\partial V(t)}{\partial \hat{\tilde{\boldsymbol{x}}}}$, the state update rule given in Eq. (2-62) can be written as [6]:

$$\dot{\hat{\tilde{\boldsymbol{x}}}} = \left(\mathcal{D} - \kappa\tilde{C}^T\tilde{\Pi}_z\tilde{C} - \kappa(\mathcal{D} - \tilde{A})^T\tilde{\Pi}_w(\mathcal{D} - \tilde{A})\right)\hat{\tilde{\boldsymbol{x}}} + \kappa\begin{bmatrix} \tilde{C}^T\tilde{\Pi}_z & (\mathcal{D} - \tilde{A})^T\tilde{\Pi}_w\tilde{B} \end{bmatrix}\begin{bmatrix} \tilde{\boldsymbol{y}} \\ \tilde{\boldsymbol{u}} \end{bmatrix} \tag{2-65}$$

It can be concluded that the filter contains several parameters. The parameters and their meaning are summarized below:

- Smoothness (s): indicating how smooth the process and measurement noises are, as explained in Section 2-3-2. The higher s, the higher the values in the noise precision matrices ($\tilde{\Pi}_w$ and $\tilde{\Pi}_z$), the more the derivatives of the states, inputs and outputs are weighted in the state update rule. The impact of this parameter is analyzed in Chapter 7.
- States and outputs embedding order (p): indicating how much derivatives are taken into account to represent the motion of the states and outputs. The impact of this parameter is also analyzed in Chapter 7.
- Inputs embedding order (d): indicating how much derivatives are taken into account to represent the motion of the inputs. The impact of this parameter is also analyzed in Chapter 7.
- Process noise covariance matrix (Σ_w): indicating how much the process noises (co)vary. The same matrix should be used in the Kalman filter for a fair comparison between the two filters. This matrix is constructed in Chapter 6.
- Measurement noise covariance matrix (Σ_z): indicating how much the measurement noises (co)vary. The same matrix should be used in the Kalman filter for a fair comparison between the two filters. This matrix is also constructed in Chapter 6.

- Learning rate (κ): indicating how quickly the state will climb uphill towards it true value, based on the variational energy manifold. The higher this value, the faster the state converges to its true value. However, a value being too high results in filter instability. This parameter is not analyzed in this thesis. However, attention is paid to the stability of the filter result. It can be concluded from Chapter 7 that none of the filter results exploded, thereby verifying this assumption.

In order to analyze the filter results in Chapter 7, the update rule from Eq. (2-65) is implemented in MATLAB by creating an LTI system using MATLAB command `ss` with the following system matrices:

$$
\begin{aligned}
A &= \left(\mathcal{D} - \kappa\tilde{C}^T\tilde{\Pi}_z\tilde{C} - \kappa(\mathcal{D}-\tilde{A})^T\tilde{\Pi}_w(\mathcal{D}-\tilde{A})\right) \in \mathbb{R}^{(n_x\cdot(p+1))\times(n_x\cdot(p+1))} \\
B &= \kappa\begin{bmatrix}\tilde{C}^T\tilde{\Pi}_z & (\mathcal{D}-\tilde{A})^T\tilde{\Pi}_w\tilde{B}\end{bmatrix} \in \mathbb{R}^{(n_x\cdot(p+1))\times(n_y\cdot(p+1)+n_u\cdot(d+1))} \\
C &= 0 \in \mathbb{R}^{1\times(n_x\cdot(p+1))} \\
D &= 0 \in \mathbb{R}^{1\times((n_y\cdot(p+1)+n_u\cdot(d+1))}
\end{aligned}
$$

In order to run the filter for every new input-output pair, this system is discretized with MATLAB command `c2d` using the 'zero-order-hold' method.

Chapter 3

Quadrotor model

Both the DEM and Kalman filter need a model of the quadrotor dynamics. Therefore, the goal of this chapter is to derive a useful, as simple as possible, quadrotor model that can be used in both filters. First of all, modelling conventions regarding reference frames, orientation and rotation and the modelling formalism are described in Section 3-1. These conventions are used in Section 3-2 to construct a six DOF quadrotor model consisting of linear position and velocity, orientation and angular velocity states in 3D. The model inputs are chosen according to the available sensors in the lab setup. Since we are interested in a relatively simple linear model, Section 3-3 describes some simplifications and gives the resulting linear model. Appendix A-3 provides the model linearization. Finally, to not unnecessarily complicate the filter results, the 12-state linear model is reduced to two states in Section 3-4.

3-1 Modelling conventions

Before defining the quadrotor model in detail, it is important to know in what frame quantities are defined. Figure 3-1 shows the frame conventions of the inertial frame (corresponding to the frame of the camera system, as explained in Chapter 4) and the body frame (corresponding to the quadrotor frame).

Besides the reference frames, we need to choose an orientation representation in a 3D space. Orientations for a 3D rigid body can be parameterized using quaternions and using Euler angles (used to construct rotation matrices). Although quaternions have been used for quadrotor modelling [27], [28], they cannot provide a unique representation of a 3D orientation [29]. Euler angles have the disadvantage that they result in a gimbal lock (singularity for $\theta = \frac{\pi}{2}$ rad) [29], [30]. However, given the fact that a gimbal lock will never happen according to the experimental setup described in Section 4-2 and the fact that rotation matrices are commonly used, they will be chosen to define the models below.

Appendices A-1 and A-2 give the derivation of the rotation matrix for translational dynamics (used to translate coordinates expressed in $\mathcal{B}$ into $\mathcal{I}$) as well as the rotation matrix for

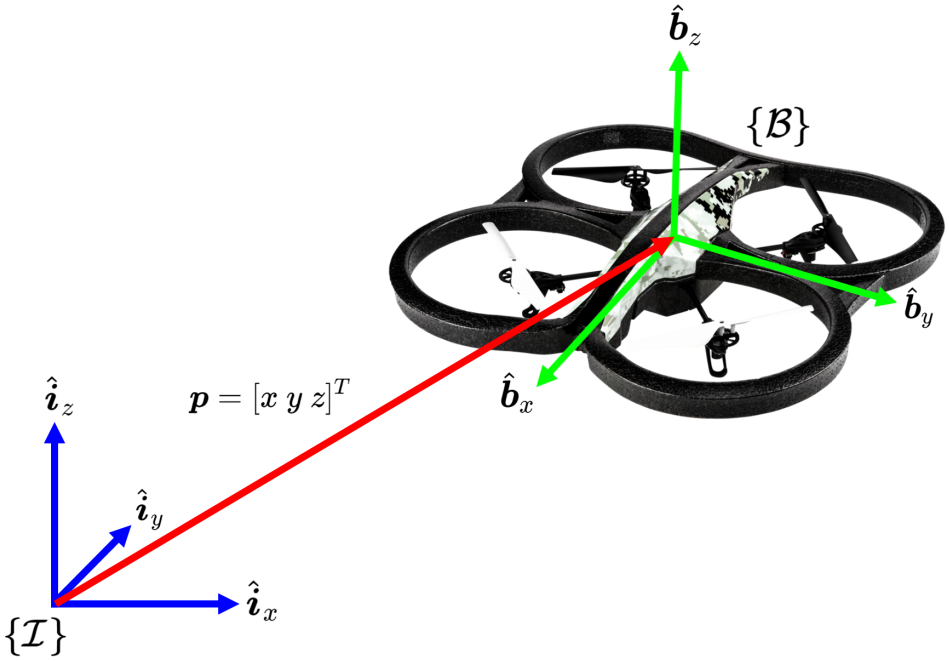


Figure 3-1: Reference frames: the inertial frame $\mathcal{I}$ coincides with the East-North-Up (ENU) standard frame as defined in ROS REP 103 [25]. The body frame $\mathcal{B}$ aligns with the Front-Left-Up (FLU) frame of the quadrotor, such that the origin coincides with the quadrotor Centre Of Mass (COM). The body frame origin is found by translating the inertial frame origin by $\boldsymbol{p} = [x\ y\ z]^T$. Adapted from [26].

rotational dynamics (used to translate angular velocities expressed in $\mathcal{B}$ into Euler angle rates). These matrices are summarized below, respectively [31]:

$$ {}^{\mathcal{I}}R_{\mathcal{B}} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{3-1} $$

$$ {}^{\mathcal{I}}R_{r,\mathcal{B}} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi sec\theta & c\phi sec\theta \end{bmatrix} \tag{3-2} $$

As Appendix A-2 shows, the angular rotations approximate the Euler angle rates if the Euler angles ϕ, θ and ψ are relatively small, meaning that the Euler angles represent the quadrotor attitude in the body frame. This is indicated in Figure 3-2.

The last modelling convention considers the formalism used to define the quadrotor model. This could either be the Euler-Lagrange or the Newton-Euler formalism. In most cases, the Newton-Euler formalism is used. Although it is less general [32], it is more intuitive to read. Therefore, the model provided below will follow the Newton-Euler formalism.

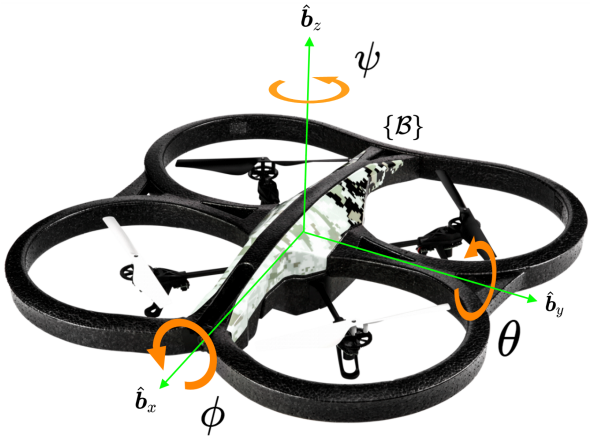


Figure 3-2: Roll (ϕ), pitch (θ) and yaw (ψ) Euler angles that are used to translate coordinates expressed in $\mathcal{B}$ into coordinates expressed in $\mathcal{I}$. If these angles are small, they approximate the angles around the body frame x-, y- and z-axis, respectively (as shown in this figure) and therefore express the quadrotor attitude. Adapted from [26].

3-2 Nonlinear model

This section provides a six Degrees Of Freedom (DOF) nonlinear quadrotor model. Before providing the nonlinear state-space model in Section 3-2-3, Section 3-2-1 and 3-2-2 derive the model equations via Newton-Euler Equations Of Motion (EOM) and quadrotor dynamics, respectively.

It is important to note that when modelling a quadrotor, the following assumptions are usually made [31], [33]:

1. The quadrotor structure is rigid.
2. The quadrotor structure is symmetrical around all three axes.
3. The quadrotor CoM coincides with the body frame origin.
4. The propellers are rigid.
5. The thrust and drag produced by the rotors are proportional to the square of the rotor rotational velocity.

The same assumptions apply to the model presented below.

3-2-1 Newton-Euler Equations Of Motion

The Newton-Euler EOM, applied to the quadrotor in the body frame, are given by:

$$\boldsymbol{f} = m\dot{\boldsymbol{v}} + \boldsymbol{\Omega} \times m\boldsymbol{v} \tag{3-3}$$

$$\boldsymbol{\tau} = I\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times I\boldsymbol{\Omega} \tag{3-4}$$

where:

- $\boldsymbol{f} \in \mathbb{R}^3$ is the force in the body frame, generated by the rotating rotors and gravity.
- $\boldsymbol{\tau} \in \mathbb{R}^3$ is the torque in the body frame, generated by the rotating rotors.
- m is the quadrotor mass.
- $I \in \mathbb{R}^{3\times3}$ is the quadrotor inertia matrix. Due to the quadrotor symmetry, it has only non-zero elements on its diagonal.
- $\boldsymbol{v} \in \mathbb{R}^3$ is the quadrotor velocity with respect to the body frame orientation.
- $\dot{\boldsymbol{v}} \in \mathbb{R}^3$ is the quadrotor acceleration with respect to the body frame orientation.
- $\boldsymbol{\Omega} \in \mathbb{R}^3$ is the quadrotor rotational velocity with respect to the body frame orientation.
- $\dot{\boldsymbol{\Omega}} \in \mathbb{R}^3$ is the quadrotor rotational acceleration with respect to the body frame orientation.

By setting $\boldsymbol{v} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$ and $\boldsymbol{\Omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}$ to rewrite Eq. (3-3) and Eq. (3-4) and by using the rotation matrices ${}^{\mathcal{I}}R_{\mathcal{B}}$ and ${}^{\mathcal{I}}R_{r,\mathcal{B}}$, we can construct the quadrotor model equations in six DOF (position and orientation are defined in $\mathcal{I}$, linear and angular velocities are defined in $\mathcal{B}$) [31]:

$$\begin{aligned}
\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} &= {}^{\mathcal{I}}R_{\mathcal{B}} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \\
\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} &= \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m}\boldsymbol{f} \\
\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} &= {}^{\mathcal{I}}R_{r,\mathcal{B}} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \\
\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} &= \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}}qr \\ \frac{I_{zz}-I_{xx}}{I_{yy}}pr \\ \frac{I_{xx}-I_{yy}}{I_{zz}}pq \end{bmatrix} + \begin{bmatrix} \frac{1}{I_{xx}} \\ \frac{1}{I_{yy}} \\ \frac{1}{i_{zz}} \end{bmatrix} \boldsymbol{\tau}
\end{aligned} \tag{3-5}$$

where g represents the gravitional acceleration and I_{xx}, I_{yy} and I_{zz} are the diagonal elements of the inertia matrix I.

Since $\boldsymbol{f}$ and $\boldsymbol{\tau}$ cannot be directly measured during flight, they cannot serve as model inputs. Therefore, the next section will describe how the model inputs relate to the the thrust and torque values using quadrotor dynamics.

3-2-2 Quadrotor dynamics

Figure 3-3 shows the quadrotor free-body diagram. It can be concluded that each rotor has its own contribution to the total force and torque acting on the quadrotor rigid-body. In order to derive the force and torque, the following assumptions are made [34], [35]:

1. Each rotor is oriented, such that the thrust it generates is directed along $\hat{\boldsymbol{b}}_z$.
2. Each rotor has exactly the same geometry (with mirrored geometry for rotors mounted next to each other).
3. There are no other air flow effects in the neighbourhood of the rotors than only the air flow generated by the rotors themselves.

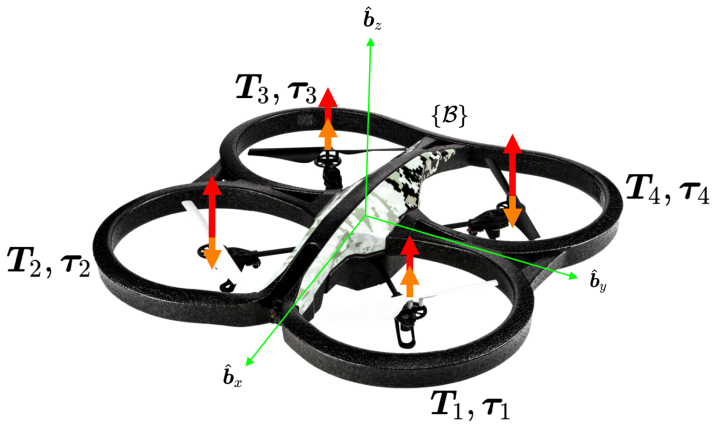


Figure 3-3: Quadrotor free body diagram: rotor 1 and 3 rotate in ClockWise (CW) direction and therefore generate an upward force (red) in $\hat{\boldsymbol{b}}_z$ direction and a torque (orange) causing a rotation in CounterClockWise (CCW) direction. Rotor 2 and 4 rotate in CCW direction and therefore generate an upward force in $\hat{\boldsymbol{b}}_z$ direction and a torque causing a rotation in CW direction. In general, the rotors generate a torque having a value being around a factor 40 smaller than the force value. Adapted from [26].

The total force in the body frame, caused by rotor rotations, is given by:

$$\boldsymbol{f} = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{3-6}$$

Theoretically, the thrust T is described by [36]:

$$T = \sum_{i=1}^{4} T_i = \sum_{i=1}^{4} c_T \omega_i^2 \tag{3-7}$$

where c_T is the so-called thrust coefficient with the purpose of creating a lumped parameter model of the thrust generation. For a specific quadrotor, this parameter can be identified to

give a proper relation between rotor velocity and generated thrust. This is one of the main contributions of Section 5-4.

The total torque is given by the following relation:

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{r1} \\ \tau_{r2} \\ \tau_{r3} \end{bmatrix} + \begin{bmatrix} \tau_{g1} \\ \tau_{g2} \\ \tau_{g3} \end{bmatrix} \tag{3-8}$$

First of all, the velocities may be configured in such a way that the quadrotor starts rolling, pitching and/or yawing. This is represented by the following torques for an X-type quadrotor [36], [37]:

$$\begin{bmatrix} \tau_{r1} \\ \tau_{r2} \\ \tau_{r3} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} l c_T(\omega_1^2 - \omega_2^2 - \omega_3^2 + \omega_4^2) \\ \frac{\sqrt{2}}{2} l c_T(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2) \\ c_Q(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \tag{3-9}$$

where l denotes the rotor arm length. c_Q is a lumped parameter used to model the torque τ_{r3} around the $\hat{\boldsymbol{b}}_z$ axis [38] and can be determined in a similar way as c_T.

Furthermore, a difference in rotational velocities also results in a gyroscopic effect. This effect actually entails a roll-inducing torque for nonzero angular velocity in pitch direction and a pitch-inducing torque for nonzero angular velocity in roll direction, which can be mathematically represented as [39]:

$$\begin{aligned} \begin{bmatrix} \tau_{g1} \\ \tau_{g2} \\ \tau_{g3} \end{bmatrix} &= -I_r \boldsymbol{\Omega} \times \hat{\boldsymbol{b}}_z(\omega_2 + \omega_4 - \omega_1 - \omega_3) \\ &= I_r \begin{bmatrix} -q \\ p \\ 0 \end{bmatrix} \omega \end{aligned} \tag{3-10}$$

where I_r represents the rotor inertia and $\omega = \omega_2 + \omega_4 - \omega_1 - \omega_3$ acts like a disturbance on the system. In this case, the generated torque is linearly dependent on the rotational velocities, instead of quadratically dependent as is the case for the torques given in (3-9).

As can be concluded, $\boldsymbol{f}$ and $\begin{bmatrix} \tau_{r1} \\ \tau_{r2} \\ \tau_{r3} \end{bmatrix}$ can be calculated using the quadratic rotor rotational velocities. However, as Section 4-1 will show, no sensors are available to measure the rotor rotational velocities during flight. Instead, the motor Pulse Width Modulation (PWM) values can be used as model inputs to relate to the rotor rotational velocities and thus to $\boldsymbol{f}$ and $\begin{bmatrix} \tau_{r1} \\ \tau_{r2} \\ \tau_{r3} \end{bmatrix}$.

As Section 5-4 will describe, the system identification experiments, used to relate motor PWM values to force and torque values, make use of a MATLAB/Simulink toolbox [40]. This

toolbox can directly control the motor PWM values. As a result, the rotors start rotating and their speed is measured using an optical Revolutions Per Minute (RPM) counter. This way, the rotor rotational velocity can be related to the motor PWM values as well as the generated thrust. However, the PWM scale of this toolbox is different from the scale used by the quadrotor software. Therefore, the rotor rotational velocity is given by:

$$\begin{aligned}\omega_i &= \boldsymbol{c}_M(1) \cdot \text{pwmM}_i + \boldsymbol{c}_M(2) \\ &= \boldsymbol{c}_M(1) \cdot \frac{\text{pwmA}_i}{2.55} + \boldsymbol{c}_M(2) \\ &= \boldsymbol{c}_A(1) \cdot \text{pwmA}_i + \boldsymbol{c}_A(2)\end{aligned} \tag{3-11}$$

where $\boldsymbol{c}_M$ and $\boldsymbol{c}_A$ are 2D constant vectors containing the coefficients to translate motor PWM values of the MATLAB/Simulink toolbox (pwmM) and the quadrotor software (pwmA) to rotor rotational velocity, respectively. The values for these coefficients are given in Section 5-4-1. Eq. (3-11) holds for $\text{pwmM}_i \geq 0.2$.

The rotational velocity squared is thus given by:

$$\begin{aligned}\omega_i^2 &= \left(\boldsymbol{c}_A(1) \cdot \text{pwmA}_i + \boldsymbol{c}_A(2)\right)^2 \\ &= \left(\boldsymbol{c}_A(1) \cdot \text{pwmA}_i + \boldsymbol{c}_A(2)\right)\left(\boldsymbol{c}_A(1) \cdot \text{pwmA}_i + \boldsymbol{c}_A(2)\right) \\ &= \boldsymbol{c}_A(1)^2 \cdot \text{pwmA}_i^2 + 2 \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) \cdot \text{pwmA}_i + \boldsymbol{c}_A(2)^2\end{aligned} \tag{3-12}$$

As Section 5-4-2 will describe, the thrust results are better explained by a 2nd-order polynomial without constant term ($\boldsymbol{c}_T$ is a 2D vector). This holds for the torque coefficients too.

The thrust can now be expressed in motor PWM values as follows:

$$\begin{aligned}T &= \boldsymbol{c}_T(1) \sum_{i=1}^{4} \omega_i^2 + \boldsymbol{c}_T(2) \sum_{i=1}^{4} \omega_i \\ &= \boldsymbol{c}_T(1) \sum_{i=1}^{4} \left(\boldsymbol{c}_A(1)^2 \cdot \text{pwmA}_i^2 + 2 \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) \cdot \text{pwmA}_i + \boldsymbol{c}_A(2)^2\right) \\ &\quad + \boldsymbol{c}_T(2) \sum_{i=1}^{4} \left(\boldsymbol{c}_A(1) \cdot \text{pwmA}_i + \boldsymbol{c}_A(2)\right) \\ &= \left(\boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1)^2\right) \sum_{i=1}^{4} \text{pwmA}_i^2 \\ &\quad + \left(2 \cdot \boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) + \boldsymbol{c}_T(2) \cdot \boldsymbol{c}_A(1)\right) \sum_{i=1}^{4} \text{pwmA}_i \\ &\quad + 4 \cdot \left(\boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(2)^2 + \boldsymbol{c}_T(2) \cdot \boldsymbol{c}_A(2)\right) \\ &= \boldsymbol{c}_{PT}(1) \sum_{i=1}^{4} \text{pwmA}_i^2 + \boldsymbol{c}_{PT}(2) \sum_{i=1}^{4} \text{pwmA}_i + \boldsymbol{c}_{PT}(3)\end{aligned} \tag{3-13}$$

The torque in roll direction can be expressed as:

$$\begin{aligned}
\tau_{r1} = \quad & \frac{\sqrt{2}}{2} l \left[\boldsymbol{c}_T(1) \sum_{i=1,i\neq 2,3}^{4} \omega_i^2 + \boldsymbol{c}_T(2) \sum_{i=1,i\neq 2,3}^{4} \omega_i - \boldsymbol{c}_T(1) \sum_{i=2}^{3} \omega_i^2 - \boldsymbol{c}_T(2) \sum_{i=2}^{3} \omega_i \right] \\
= \quad & \frac{\sqrt{2}}{2} l \left[\boldsymbol{c}_T(1) \sum_{i=1,i\neq 2,3}^{4} \left(\boldsymbol{c}_A(1)^2 \cdot \mathrm{pwmA}_i^2 + 2 \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) \cdot \mathrm{pwmA}_i + \boldsymbol{c}_A(2)^2 \right) \right. \\
& + \boldsymbol{c}_T(2) \sum_{i=1,i\neq 2,3}^{4} \left(\boldsymbol{c}_A(1) \cdot \mathrm{pwmA}_i + \boldsymbol{c}_A(2) \right) \\
& - \boldsymbol{c}_T(1) \sum_{i=2}^{3} \left(\boldsymbol{c}_A(1)^2 \cdot \mathrm{pwmA}_i^2 + 2 \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) \cdot \mathrm{pwmA}_i + \boldsymbol{c}_A(2)^2 \right) \\
& \left. - \boldsymbol{c}_T(2) \sum_{i=2}^{3} \left(\boldsymbol{c}_A(1) \cdot \mathrm{pwmA}_i + \boldsymbol{c}_A(2) \right) \right] \\
= \quad & \frac{\sqrt{2}}{2} l \left[\boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1)^2 \sum_{i=1,i\neq 2,3}^{4} \mathrm{pwmA}_i^2 + \left(2 \cdot \boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) + \boldsymbol{c}_T(2) \cdot \boldsymbol{c}_A(1) \right) \sum_{i=1,i\neq 2,3}^{4} \mathrm{pwmA}_i \right. \\
& \left. - \boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1)^2 \sum_{i=2}^{3} \mathrm{pwmA}_i^2 - \left(2 \cdot \boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) + \boldsymbol{c}_T(2) \cdot \boldsymbol{c}_A(1) \right) \sum_{i=2}^{3} \mathrm{pwmA}_i \right] \\
= \quad & \frac{\sqrt{2}}{2} l \left[\boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1)^2 \left(\sum_{i=1,i\neq 2,3}^{4} \mathrm{pwmA}_i^2 - \sum_{i=2}^{3} \mathrm{pwmA}_i^2 \right) \right. \\
& \left. + \left(2 \cdot \boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) + \boldsymbol{c}_T(2) \cdot \boldsymbol{c}_A(1) \right) \left(\sum_{i=1,i\neq 2,3}^{4} \mathrm{pwmA}_i - \sum_{i=2}^{3} \mathrm{pwmA}_i \right) \right] \\
= \quad & \boldsymbol{c}_{P\phi}(1) \left(\sum_{i=1,i\neq 2,3}^{4} \mathrm{pwmA}_i^2 - \sum_{i=2}^{3} \mathrm{pwmA}_i^2 \right) + \boldsymbol{c}_{P\phi}(2) \left(\sum_{i=1,i\neq 2,3}^{4} \mathrm{pwmA}_i - \sum_{i=2}^{3} \mathrm{pwmA}_i \right)
\end{aligned} \tag{3-14}$$

Similarly, the torque in pitch direction can be expressed as:

$$\begin{aligned}
\tau_{r2} = \quad & \frac{\sqrt{2}}{2} l \left[\boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1)^2 \left(\sum_{i=3}^{4} \mathrm{pwmA}_i^2 - \sum_{i=1}^{2} \mathrm{pwmA}_i^2 \right) \right. \\
& \left. + \left(2 \cdot \boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) + \boldsymbol{c}_T(2) \cdot \boldsymbol{c}_A(1) \right) \left(\sum_{i=3}^{4} \mathrm{pwmA}_i - \sum_{i=1}^{2} \mathrm{pwmA}_i \right) \right] \\
= \quad & \boldsymbol{c}_{P\theta}(1) \left(\sum_{i=3}^{4} \mathrm{pwmA}_i^2 - \sum_{i=1}^{2} \mathrm{pwmA}_i^2 \right) + \boldsymbol{c}_{P\theta}(2) \left(\sum_{i=3}^{4} \mathrm{pwmA}_i - \sum_{i=1}^{2} \mathrm{pwmA}_i \right)
\end{aligned} \tag{3-15}$$

Similarly, the torque in yaw direction can be expressed as:

$$\begin{aligned}\tau_{r3} = \quad & \boldsymbol{c}_Q(1)\cdot\boldsymbol{c}_A(1)^2\left(\sum_{i=1,i\neq 2}^{3}\text{pwmA}_i^2-\sum_{i=2,i\neq 3}^{4}\text{pwmA}_i^2\right)\\ & +\left(2\boldsymbol{c}_Q(1)\cdot\boldsymbol{c}_A(1)\cdot\boldsymbol{c}_A(2)+\boldsymbol{c}_Q(2)\cdot\boldsymbol{c}_A(1)\right)\left(\sum_{i=1,i\neq 2}^{3}\text{pwmA}_i-\sum_{i=2,i\neq 3}^{4}\text{pwmA}_i\right)\\ = \quad & \boldsymbol{c}_{P\psi}(1)\left(\sum_{i=1,i\neq 2}^{3}\text{pwmA}_i^2-\sum_{i=2,i\neq 3}^{4}\text{pwmA}_i^2\right)+\boldsymbol{c}_{P\psi}(2)\left(\sum_{i=1,i\neq 2}^{3}\text{pwmA}_i-\sum_{i=2,i\neq 3}^{4}\text{pwmA}_i\right)\end{aligned} \tag{3-16}$$

For the rest of this chapter, the model input is defined as $\begin{bmatrix}\text{pwmA}_1 & \text{pwmA}_2 & \text{pwmA}_3 & \text{pwmA}_4\end{bmatrix}^T$. However, in most equations T, τ_{r1}, τ_{r2} and τ_{r3} are used for notational convenience.

3-2-3 Nonlinear state-space model

Using derivations from the previous sections, we can rewrite Eq. (3-5) and express the complete nonlinear quadrotor state-space model as [31], [39]:

$$\begin{aligned}\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \dot{z}\end{bmatrix} &= {}^{\mathcal{I}}R_{\mathcal{B}}\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix}+\begin{bmatrix}0\\ 0\\ -g\end{bmatrix}\\ \begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix} &= \begin{bmatrix}rv-qw\\ pw-ru\\ qu-pv\end{bmatrix}+\begin{bmatrix}0\\ 0\\ \frac{1}{m}T\end{bmatrix}\\ \begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} &= {}^{\mathcal{I}}R_{r,\mathcal{B}}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix}\\ \begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} &= \begin{bmatrix}\frac{I_{yy}-I_{zz}}{I_{xx}}qr\\ \frac{I_{zz}-I_{xx}}{I_{yy}}pr\\ \frac{I_{xx}-I_{yy}}{I_{zz}}pq\end{bmatrix}+\begin{bmatrix}-\frac{I_r}{I_{xx}}q\\ \frac{I_r}{I_{yy}}p\\ 0\end{bmatrix}\omega+\begin{bmatrix}\frac{1}{I_{xx}}\tau_{r1}\\ \frac{1}{I_{yy}}\tau_{r2}\\ \frac{1}{I_{zz}}\tau_{r3}\end{bmatrix}\end{aligned} \tag{3-17}$$

3-3 Linear model

Before linearizing the nonlinear model, first some simplifications are made. These simplifications rely on the assumption that the quadrotor is following non-aggressive and smooth trajectories. This causes the quadrotor rigid-body angles as well as the rotational velocities to be small. Small angles imply that the body frame orientation is approximately equal to the inertial frame orientation. Therefore, ${}^{\mathcal{I}}R_{r,\mathcal{B}}$ becomes an identity matrix and the following relation holds:

$$\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} \approx \begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} \tag{3-18}$$

Furthermore, small angles imply that all rotors are rotating at approximately the same speed. Therefore, the disturbance term ω disappears. This means that the moment introduced by an individual rotating rotor is cancelled by the rotor next to it, so the role of rotor inertia vanishes. The torques acting on the quadrotor rigid-body can now be written as:

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{r1} \\ \tau_{r2} \\ \tau_{r3} \end{bmatrix} = \begin{bmatrix} \tau_{\phi} \\ \tau_{\theta} \\ \tau_{\psi} \end{bmatrix} \tag{3-19}$$

Small rotational velocities of the quadrotor imply negligible Coriolis terms, so these terms are also removed from the model equations.

These simplifications render Eq. (3-17) in the following form (all states are now expressed in the inertial frame):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ (c\phi s\theta c\psi + s\phi s\psi)\frac{1}{m}T \\ (c\phi s\theta s\psi - s\phi c\psi)\frac{1}{m}T \\ c\phi c\theta \frac{1}{m}T - g \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \frac{1}{I_{xx}}\tau_{\phi} \\ \frac{1}{I_{yy}}\tau_{\theta} \\ \frac{1}{I_{zz}}\tau_{\psi} \end{bmatrix} \tag{3-20}$$

The linear model is derived in Appendix A-3 and given by the following state equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m}c_{Bz} & \frac{1}{m}c_{Bz} & \frac{1}{m}c_{Bz} & \frac{1}{m}c_{Bz} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{I_{xx}}c_{B\phi} & -\frac{1}{I_{xx}}c_{B\phi} & -\frac{1}{I_{xx}}c_{B\phi} & \frac{1}{I_{xx}}c_{B\phi} \\ -\frac{1}{I_{yy}}c_{B\theta} & -\frac{1}{I_{yy}}c_{B\theta} & \frac{1}{I_{yy}}c_{B\theta} & \frac{1}{I_{yy}}c_{B\theta} \\ \frac{1}{I_{zz}}c_{B\psi} & -\frac{1}{I_{zz}}c_{B\psi} & \frac{1}{I_{zz}}c_{B\psi} & -\frac{1}{I_{zz}}c_{B\psi} \end{bmatrix} \begin{bmatrix} \text{pwmA}_1 \\ \text{pwmA}_2 \\ \text{pwmA}_3 \\ \text{pwmA}_4 \end{bmatrix} \tag{3-21}$$

where c_{Bz}, $c_{B\phi}$, $c_{B\theta}$ and $c_{B\psi}$ are given by Eq. (A-22) until (A-25) in Appendix A-3.

3-4 Reduced model

The goal of this section is to simplify the linear state-space model in Eq. (3-21) as much as possible to not unncessarily complicate the filtering results, but still be able to draw useful conclusions regarding filtering performance. Eq. (3-21) clearly indicates that the total state-space consists of four independent linear systems. These subsystems include the following states:

1. $\dot{\phi}$, ϕ, $\dot{y}$ and y
2. $\dot{\theta}$, θ, $\dot{x}$ and x
3. $\dot{z}$ and z
4. $\dot{\psi}$ and ψ

The experiment description in Section 4-2 indicates that the quadrotor will hover a few metres away from a wind source. The quadrotor is oriented in such a way that the roll rate (and therefore also the roll angle, linear y velocity and y position) is directly influenced by the air flow coming from the wind source. Therefore, subsystem 1 is selected.

This subsystem can be even further reduced by only including states $\dot{\phi}$ and ϕ. Therefore, the following model will be used in the filters:

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{I_{xx}} c_{B\phi} & -\frac{1}{I_{xx}} c_{B\phi} & -\frac{1}{I_{xx}} c_{B\phi} & \frac{1}{I_{xx}} c_{B\phi} \end{bmatrix} \begin{bmatrix} \text{pwmA}_1 \\ \text{pwmA}_2 \\ \text{pwmA}_3 \\ \text{pwmA}_4 \end{bmatrix} \tag{3-22}$$

To make the filter model complete, the following output equation is chosen:

$$\boldsymbol{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \text{pwmA}_1 \\ \text{pwmA}_2 \\ \text{pwmA}_3 \\ \text{pwmA}_4 \end{bmatrix} \tag{3-23}$$

The argument for choosing this output equation is twofold: first of all, we could choose to measure both states, because sensors are available for both quantities (see Section 4-1). However, estimating a hidden state is much more interesting than estimating an observed state from a filtering perspective. Therefore, it is chosen to measure one state. Secondly, choosing to measure $\dot{\phi}$ would mean that the system is not completely observable (the system has one unobservable mode: ϕ, since we try to construct the value of a quantity of which only the derivative is measured). This implies that we have to add an extra state to the system: $\dot{y}$. It is not preferred to include $\dot{y}$, because there is no sensor present that can properly measure $\dot{y}$ (it is only estimated on the quadrotor, see Section 4-1) and it makes the system more complex than necessary. Therefore, it is chosen to measure ϕ. This choice makes the system observable and the hidden state, $\dot{\phi}$, is influenced by the output as well as the inputs, both of which are present in the state update rule of the DEM filter. The filter should be able to come up with a proper state estimate, despite a mismatch in these measurements.

Chapter 4

Experimental setup

This chapter describes the experimental setup used to gather data for filter analysis. It does so by first giving an overview of the lab hardware and software setup in Section 4-1. The lab setup includes the OptiTrack MoCap system, the Parrot AR.Drone 2.0 quadrotor, a TUD desktop, a user-dependent laptop and two routers. The AR.Drone 2.0 and OptiTrack MoCap system are described in Sections 4-1-1 and 4-1-2, respectively. Furthermore, the software platform ROS is used to interconnect the different systems, as described in Section 4-1-3. Secondly, Section 4-2 describes how this hardware and software setup is used, together with a wind source to constitute the total setup used for the experiment considered in this thesis. Finally, Appendix C gives some extra details regarding the quadrotor simulator and the experiment plan corresponding to the content of this chapter.

4-1 Lab hardware and software setup

Figure 4-1 shows the Networked Embedded Robotics lab in Delft (NERDlab) within Delft Center for Systems and Control (DCSC) used to conduct all quadrotor flight experiments during this thesis. During the experiments, the lab consisted of several components. Figure 4-2 gives an overview of these components and how they are interconnected.

The lab room contains $9 \times 4 \times 3$ m free space [41], which is perfectly suited for conducting the experiment described in Section 4-2. To perform an experiment in a safe way, the lab provides the possibility to enclose the whole lab with a protective net cage, as can also be seen in Figure 4-1.

The lab is equipped with an OptiTrack Motion Capture (MoCap) system able to measure the quadrotor position and orientation in 3D space. As Figure 4-2 shows, this information is assembled in the Motive:Tracker program, installed on a TU Delft (TUD) desktop present in the lab. See Section 4-1-2 for more information about the OptiTrack system and Motive:Tracker. Via Motive:Tracker, the quadrotor position and orientation are multicasted via a Netgear Nighthawk AC1900 router. This router is connected with an Ethernet cable to a laptop running ROS packages that interpret the OptiTrack data and control the quadrotor. Section 4-1-3

Figure 4-1: NERDlab with protective netting and Parrot AR.Drone 2.0 placed inside.

gives more information about the ROS packages used. The control commands are converted to User Datagram Protocol (UDP) packets and sent via a Wireless Fidelity (Wi-Fi) network to the quadrotor (see Section 4-1-1 for more information about the quadrotor) [42]. This Wi-Fi network is established by connecting both laptop and quadrotor to another Netgear Nighthawk AC1900 router that is not connected to the World Wide Web (WWW) to avoid communication overhead. During the flight experiment, data is recorded in a ROS bag file. In post-processing stage, this data is read out in MATLAB and processed for proper analysis.

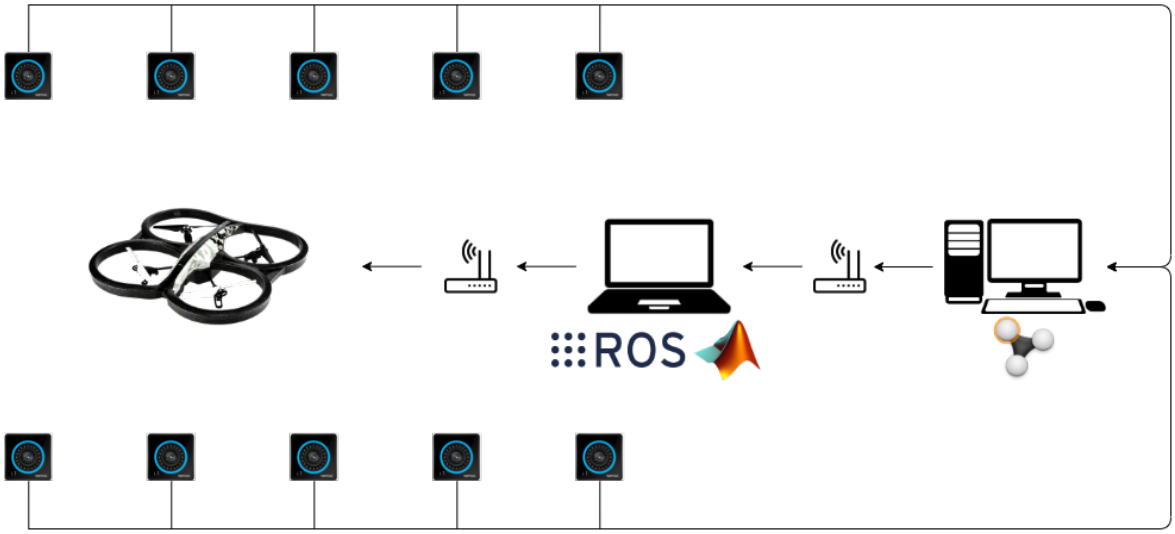


Figure 4-2: Symbolic overview of lab components, including 10 OptiTrack Prime 17W cameras, a TUD desktop computer (with Motive:Tracker installed), a user-dependent laptop (with ROS and MATLAB installed), two Netgear Nighthawk AC1900 routers and a Parrot AR.Drone 2.0. Images sources: [26], [43]-[49].

4-1-1 Parrot AR.Drone 2.0 quadrotor

For the experiment in this thesis, the Parrot AR.Drone 2.0 quadrotor (as shown in Figure 4-3) is chosen to be used. The Parrot AR.Drone 2.0 is a very popular quadrotor among researchers [50]. This is no surprise, since Parrot has been able to deliver a product with relatively low Time To Value (TTV) (i.e. the product owner quickly gets value out of the quadrotor). A low TTV is caused by the fact that the AR.Drone 2.0 is inexpensive, light-weight and contains internal control software to be able to hover, perform fast forward flight and ensure a safe flight [50], [51].

Figure 4-3: Parrot AR.Drone 2.0 with Parrot battery attached and indoor hull detached.

Parrot also offers a software development guide and Software Development Kit (SDK), which enables others to send high-level commands to the quadrotor from their own software package [42]. The availability of the SDK has caused the Autonomy Lab group of the Simon Fraser University (SFU) in Canada to make the AR.Drone 2.0 communication compatible with ROS by implementing the *ardrone_autonomy* package [52]. This package is used by the Technical University of Munich (TUM) vision group to create the *tum_ardrone* ROS package implementing a quadrotor position controller [53]. Besides the position controller, this package also offers a Graphical User Interface (GUI) giving the possibility to choose between controlling the quadrotor automatically (using the position controller), using keyboard or using joystick. The experiment in this thesis only considers joystick control (see Section 4-2).

The advantage of using ROS to control the quadrotor is the fact that the control software can remain the same for simulation as for physical flights: the *tum_ardrone* package is also compatible with the *tum_simulator* package. Therefore, the software development time is reduced. Although simulation is not considered in this thesis, time has been spent on getting the simulation environment running and therefore it will be the topic of Appendix C-1. This appendix also proposes another possible simulator and indicates the preferred one for future research.

Using the *ardrone_autonomy* package, the following sensor information can be received from the quadrotor [42]:

- Attitude (neither in $\mathcal{B}$, nor in $\mathcal{I}$, but in a quadrotor-specific frame out of scope)
- Angular velocities in $\mathcal{B}$

- Altitude in $\mathcal{I}$
- Linear velocities in $\mathcal{B}$
- Linear accelerations in $\mathcal{B}$
- Magnetic field intensity
- Pressure of the medium the drone is flying in
- Imagery of forward (and downward) view as seen from the drone
- Motor PWM values

This information can be received in real-time at two frequencies: either 15 Hz or 200 Hz [54]. 200 Hz is chosen as update frequency, because it provides more accurate information about the quadrotor movements.

4-1-2 OptiTrack MoCap system

As Figure 4-2 shows, the lab is equipped with 10 OptiTrack Prime 17W cameras. Each camera has a horizontal Field Of View (FOV) of 70° and a vertical FOV of 49° [55]. Together, these cameras span a volume of $9 \times 4 \times 2.2$ m in which they are able to detect the 3D position of reflective markers by making use of 850 nm InfraRed (IR) technology [41], [55]. A total of four reflective markers are attached to the quadrotor (see Figure 4-4), such that the OptiTrack system is able to construct a unique 3D quadrotor rigid-body representation at all time during flight. The marker positions are communicated from the OptiTrack cameras via Ethernet cables to the TUD desktop with running Motive:Tracker software [56]. The Motive:Tracker program shows the reflective markers in space, calculates the 3D rigid-body position and orientation and multicasts this information to the first Netgear router (as shown in Figure 4-2). With a camera frame rate of 360 Frames Per Second (FPS) and a latency of 2.8 ms [55], the complete OptiTrack system is able to provide the 3D rigid-body position and orientation at a rate of 120 Hz.

The *mocap_optitrack* ROS package [57] is developed, such that the 3D position and orientation information multicasted by the Motive:Tracker can be properly interpreted and recorded in a ROS bag file for post-processing.

Since the AR.Drone 2.0 update frequency is set to 200 Hz, the OptiTrack update frequency is the limiting factor in the system. Therefore, all data is (sub)sampled at 120 Hz using linear interpolation and the system equations are discretized with sampling time $\frac{1}{120}$ s.

4-1-3 Robot Operating System

ROS is a robot software platform designed to encourage collaborations in the robotics area between a significant amount of experts in different areas. By using standardized interfaces and libraries, ROS aims to bridge the gap between the expert groups with different expertise to allow for quickly building an interconnected and robust robotics software platform [58].

Figure 4-4: Parrot AR.Drone 2.0 with indoor hull and OptiTrack markers attached.

As mentioned before, ROS is used in this thesis, due to the compatibility of simulation and physical flight. The most important ROS packages that have been used to fly the AR.Drone 2.0 in the lab are listed below:

- *ardrone_autonomy*
- *tum_ardrone*
- *mocap_optitrack*

These packages are compatible with ROS Kinetic and Ubuntu 16.04 and run on the user-dependent laptop in Figure 4-2. Using the GUI from the *tum_ardrone* package and the AR.Drone 2.0 communication layer from *ardrone_autonomy*, the quadrotor is controlled using joystick. The data is recorded during flight using the *ardrone_autonomy* and *mocap_optitrack* packages to allow for post-processing. Figure 4-5 shows the most important ROS nodes and topics used to collect flight data.

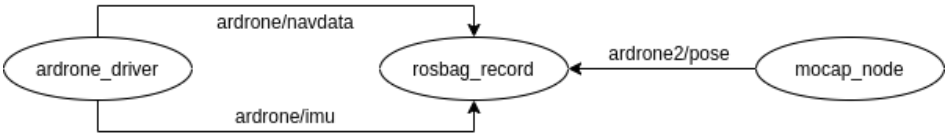


Figure 4-5: ROS node and topic interconnections to record data during flight.

The *ardrone2/pose* topic contains the quadrotor 3D position and orientation in the OptiTrack system coordinate frame, thereby including the system output as defined in Section 3-4 (i.e. the roll angle ϕ). The model inputs are given through the *ardrone/navdata* topic from the quadrotor. This includes the motor PWM values for each motor separately. Finally, the hidden state reference $\dot{\phi}$ (used to benchmark the DEM filter) is obtained by collecting the angular velocity data from the *ardrone/imu* topic.

4-2 Experiment description

Figure 4-6 shows the initial setup of the quadrotor experiment discussed in this thesis. The setup consists of the above described OptiTrack MoCap system and the Parrot AR.Drone 2.0, and a wind source [59]. The wind source is a drum fan used to create smooth air flows in the lab, such that the quadrotor movements will also be smooth and the resulting system noises (as discussed in Section 2-3-2) will be smooth too.

Before the experiment, the quadrotor was positioned at the origin of the OptiTrack MoCap system, indicated by the tape cross on the ground. The quadrotor was visually aligned with this cross after which the small OptiTrack position and orientation values were set to zero in Motive:Tracker. When aligned with the OptiTack axes, the quadrotor was turned on in order to have the same coordinate frame as the OptiTrack MoCap system.

After initialization, the software was run and the quadrotor was manually controlled with a joystick to hover at approximately 6 m from the wind source. The wind source was turned on and kept on for a period of time that allowed to take data for 30 s while the wind was constant.

The procedure described above was repeated a few times using different modes and orientations of the wind source. From these experiments, the one with the smoothest and cleanest data was taken to be analyzed in Chapters 6 and 7.

Appendix C-2 provides an extensive universal experiment plan, as well as the exact details of the experiment considered in this thesis.

During the experiment, it was observed that the quadrotor exhibited periodic behaviour as a result of the air flows present in the lab. These air flows were caused by the rotor blade movements of the quadrotor itself, as well as the air flows coming from the wind source. Walls and other objects in the lab space caused the air flows to circulate. The air flow circulation is the most probable cause of the periodic quadrotor movements. Section 6-2-2 shows that this affects the roll rate process noise and gives possible explanations for this result.

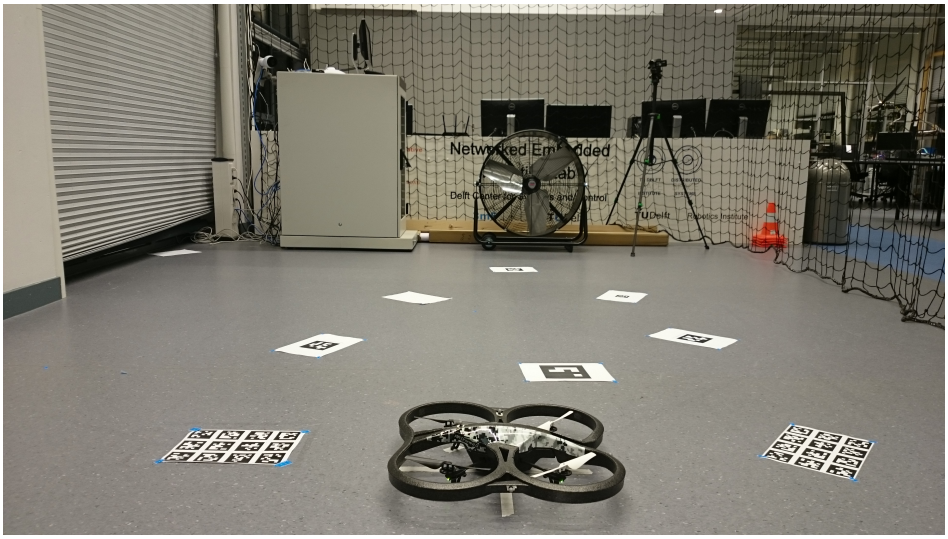


Figure 4-6: Initial setup of the experiment considered in this thesis.

Chapter 5

System identification

The main goal of this chapter is to elaborate on the experiments conducted to find the model parameter values corresponding to the Parrot AR.Drone 2.0 to use in the quadrotor model. These parameters include the mass (Section 5-1), arm length (Section 5-2), moment of inertia (Section 5-3) and thrust and torque coefficients (Section 5-4). Background information on the measurements is given in Appendix B.

5-1 Mass m

The mass to be used in the model consists of the mass of the quadrotor body with indoor hull and the battery attached to it. These masses are all determined by measuring the mass three times using a weighing scale with an accuracy of 1 g and averaging the values. The results are given in Table 5-1.

Table 5-1: Mass values of different quadrotor components.

Component	**Mass** (kg)
Quadrotor frame with indoor hull	0.362
Parrot battery	0.119
Akku-King battery	0.135

5-2 Rotor arm length l

The arm length of a rotor is the distance between rotor axle and quadrotor COM. It is determined by measuring the arm between two diagonally located rotors and dividing it by two. This yields an arm length of 0.178 m for all rotors. This length is validated by also measuring the distance between two neighbouring rotors and using the Pythagorean theorem.

5-3 Inertia matrix I

This section derives the values in the quadrotor inertia matrix. In general, this matrix is given by:

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{5-1}$$

Due to the assumption of a symmetric quadrotor, as explained in Section 3-2, all non-diagonal terms become zero and we are left with:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{5-2}$$

There are different methods to determine the mass moment of inertia around the three principle axes of the quadrotor. First of all, it can be determined geometrically by modelling all parts of the quadrotor as uniform mass distributions with their COM being located at a certain distance from the quadrotor COM (e.g. [60]-[62]). The disadvantage of these methods is the simplifying assumption of the mass distribution and shapes of the different quadrotor parts. A better approach may be to use Computer-Aided Design (CAD) tools that can automatically calculate these values (e.g. [35]). However, CAD models often face the same kind of simplifying assumptions. These assumptions can be overcome by doing an experiment, such as the bifilar pendulum experiment (e.g. [63]). The disadvantage of this method is the fact that it can be subject to measurement noise.

In this case, the bifilar pendulum experiment is chosen to be used, because it is a relatively easy way of obtaining the moment of inertia of the complete AR.Drone 2.0, including indoor hull. Furthermore, it is preferred not to decompose the quadrotor into all its separable parts and a CAD model was not available.

Figure 5-1 shows a schematic view of this experiment. The wires are separated by distance d_w and have a length l_w (in the figure indicated by D and h, respectively). In this case the quadrotor is the test object that needs to be attached to some horizontal frame using two parallel (thin) wires of equal length. The quadrotor is then rotated by an initial angle α_0 (in the figure indicated as θ) and released. This will introduce a damped quadrotor oscillation around the axis parallel to the wires. During this experiment, the rotational velocity of the quadrotor body is measured by collecting data from its own gyroscope using a MATLAB/Simulink toolbox specifically designed for the AR.Drone 2.0 [40]. After collecting this data, a simulation of the theoretical bifilar pendulum is run and the resulting rotational velocity is fitted to the measured rotational velocity by adjusting three variables, including the moment of inertia.

The resulting experiment values of the inertia components around the three axes are shown in Table 5-2.

The inertia values around the x- and y-axis do not differ much, since the mass distributions around these axes are approximately equal (except for the front camera and the slightly non-symmetric shape of the indoor hull). When regarding the z-axis, the quadrotor has more mass

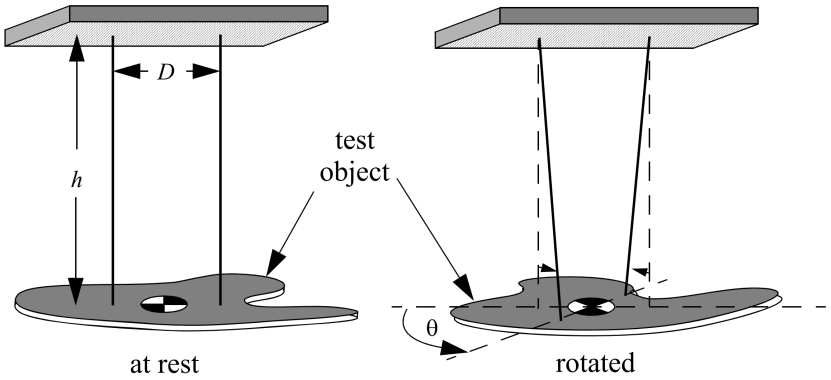


Figure 5-1: Schematic view of bifilar pendulum experiment.

Table 5-2: Results of mass moment of inertia experiments.

Inertia component	**Value** ($\mathrm{kg\,m^2}$) (batP)	**Value** ($\mathrm{kg\,m^2}$) (batA)
I_{xx}	$3.4 \cdot 10^{-3}$	$3.5 \cdot 10^{-3}$
I_{yy}	$4.0 \cdot 10^{-3}$	$4.1 \cdot 10^{-3}$
I_{zz}	$6.9 \cdot 10^{-3}$	$6.9 \cdot 10^{-3}$

distributed with a larger distance to the quadrotor COM, causing a higher inertia value. It should be noted that the Akku-King battery is slightly heavier than the Parrot battery, which introduces a slightly higher inertia value. See Appendix B-1 for more information about these experiments.

It turns out to be difficult to verify these values based on other literature, since the values found in the literature are varying (compare the values provided in [34], [64], [65] and [66]). For the experiments presented here, only [64] and [66] can be used to compare with, because [65] uses a simplified CAD model without indoor hull and [34] calculates the inertia values for the AR.Drone 2.0 with outdoor hull. [64] actually uses the same experiment setup, but the value for the moment of inertia is optimized using the MATLAB function `fminsearch`. This function needs a proper loss function that drives the solution in the right direction. [64] uses the L1 loss function, which calculates the absolute difference between two signals. However, this loss function will return similar errors for inertia values that may differ $1 \cdot 10^{-3}$ $\mathrm{kg\,m^2}$ from each other, which is a significant amount compared to the number given in Table 5-2. Therefore, the method as described in Appendix B-1-3 seems to better fit the experiment purpose. Furthermore, [66] provides similar inertia values as presented above, indicating that these values can be used.

5-4 Aerodynamic thrust and torque coefficients c_T and c_Q

This section derives the values of the thrust and torque coefficients. The coefficient values for the AR.Drone 2.0 have already been determined by others (see [34] and [64]). In [64], the experiments were performed by mounting the quadrotor on a torque and force sensor just

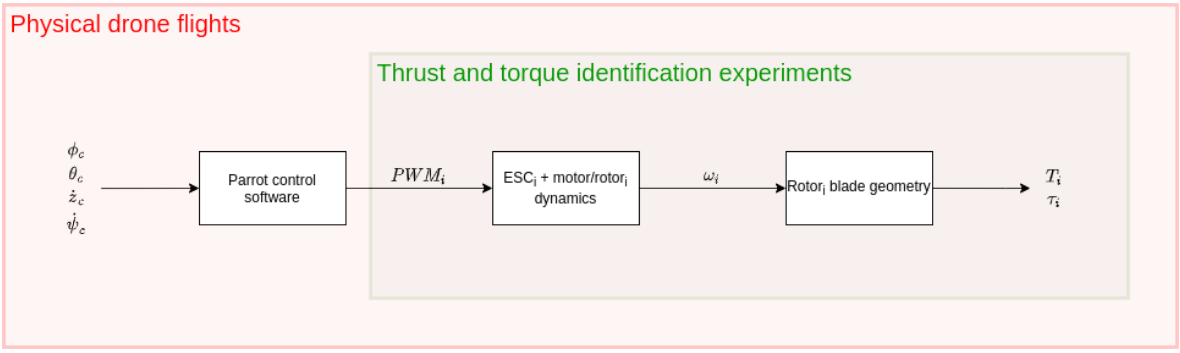


Figure 5-2: System overview controller inputs to rotor thrust and torque generated by rotor i.

above a table without indoor hull and by powering the quadrotor using a lab power supply. In [34], the experiments were performed by mounting the quadrotor without indoor hull to different setups that were attached to a weighing scale and by powering the quadrotor using the original Parrot batteries. Involving a weighing scale in the experiments possibly introduced fluctuating values, causing the authors to execute a significant number of experiments and average them. In conclusion, in both sources the quadrotor was located near a table surface without indoor hull, possibly influencing the air flow through the rotors. Furthermore, the voltage dependency of the rotor rotational velocity was not investigated. These aspects will be the main contributions of the results presented in this thesis.

In order to determine the aerodynamic coefficients, it is important to understand the system and flow of signals between high-level control inputs provided to the quadrotor and the resulting thrusts and torques introduced by the rotating rotors. This system is displayed in Figure 5-2. The figure shows that the quadrotor receives the commanded roll and pitch angle, ϕ_c and θ_c, vertical speed, $\dot{z}_c$, and yaw rate, $\dot{\psi}_c$. Depending on the current quadrotor state, these are translated into 4 PWM signals that are provided to the Electronic Speed Controller (ESC) of each motor-rotor combination. The ESC controls the power to the motor, causing the rotors to rotate with a certain rotational velocity ω_r that should come close to the desired rotational velocity. Due to the blade geometry, the rotors generate a certain thrust and torque, causing the quadrotor to move.

In the coefficient determination experiments, the Parrot control software is omitted, because the behaviour of this block is not completely known and can only be described by a high-level model. Instead, the aforementioned MATLAB/Simulink toolbox is used to immediately send a certain PWM value to each individual ESC, as illustrated in Figure 5-2.

It is thus important to determine the relation between PWM value and resulting thrust and torque. As the figure shows, this translation actually consists of two steps: PWM to rotational velocity and rotational velocity to generated thrust/torque (characterized by the thrust and torque coefficient, respectively). The first step will be described in Section 5-4-1. The second step is the main topic of Sections 5-4-2 and 5-4-3, in which two different setups are used to create the rotational velocity to thrust and torque characteristic, respectively. All datasets used to construct the results in this section are given in Appendices B-2, B-3 and B-4.

5-4-1 Relation between PWM and rotor rotational velocity

To characterize the relation between PWM value and rotor speed, the following aspects are taken into account:

1. In theory, the ESC-motor-rotor combinations are exactly the same, but in practice they will have slightly different characteristics.
2. The ESC-motor combination depends on the battery voltage, unless actively compensated for.
3. The PWM values used in the MATLAB/Simulink toolbox differ from the PWM values given by the AR.Drone 2.0.

It turns out that each ESC-motor-rotor combination is slightly different. The most deviating rotor speeds (from rotor 1 and rotor 2) still deviate less than 1% from each other. This reproduces very well at different PWM values, so this effect is ignored. However, this effect is taken into account in the experiments by measuring the rotational velocities of either rotor 3 or 4, thereby giving an approximate average rotor velocity value.

Figure 5-3 and 5-4 provide the rotational velocity of rotor 4 given different battery voltages and different PWM values (5-40 and 45-80, respectively). These figures are constructed using eight experiments with a gradually increasing PWM value with steps of 5 from 0 to 100 and back (in order to check on hysteresis effects). Each PWM value is kept constant for 10 s in order to be able to average the battery voltage and to have enough time to manually measure the rotational velocity of rotor 4. In total, this gives eight measurement points per rotational velocity. See Appendix B-2-1 for the corresponding datasets.

After collecting the data points, a 1st-order polynomial fit is constructed that optimally fits the measurement points per rotational velocity. The purpose of this fit is to show that the PWM-ω_r relation is battery-independent. Visually, these fits seem to be horizontal. However, this is not completely true. To prove this, Figure 5-5 shows the percentage of the difference between end and begin point of the fits with respect to their means per PWM setpoint. It turns out that these percentages are less than 0.25%, indicating that the battery-dependency of the rotational velocity for a certain PWM setpoint can be ignored. These experiments are conducted using two types of batteries that are also used in the lab experiments: the original Parrot batteries (indicated with batP) and somewhat bigger Akku-King batteries (indicated with batA). Appendix B-2-2 shows the same figures, but with a distinction in battery type. From these figures can be concluded that for each battery the fits can also be regarded horizontal, so battery-dependence is not an issue in this system. Apparently, the ESC actively controls the motor speed depending on the battery voltage. Furthermore, it can be concluded that no hysteresis effect is present in the battery-dependent control of the ESC.

The attentive reader may notice that these plots are only created for PWM setpoints 5-80, not for 5-100. This is caused by the fact that the rotational velocity measurements are highly variable for PWM setpoints of 85 and higher, as will be shown below in Figure 5-6. This will only introduce uncertainty in the measurements with very little practical usefulness, since the rotors will reach those rotational velocities only for approximately 0.1 s during takeoff and in case of aggressive manoeuvres. Both reach beyond the scope of this thesis.

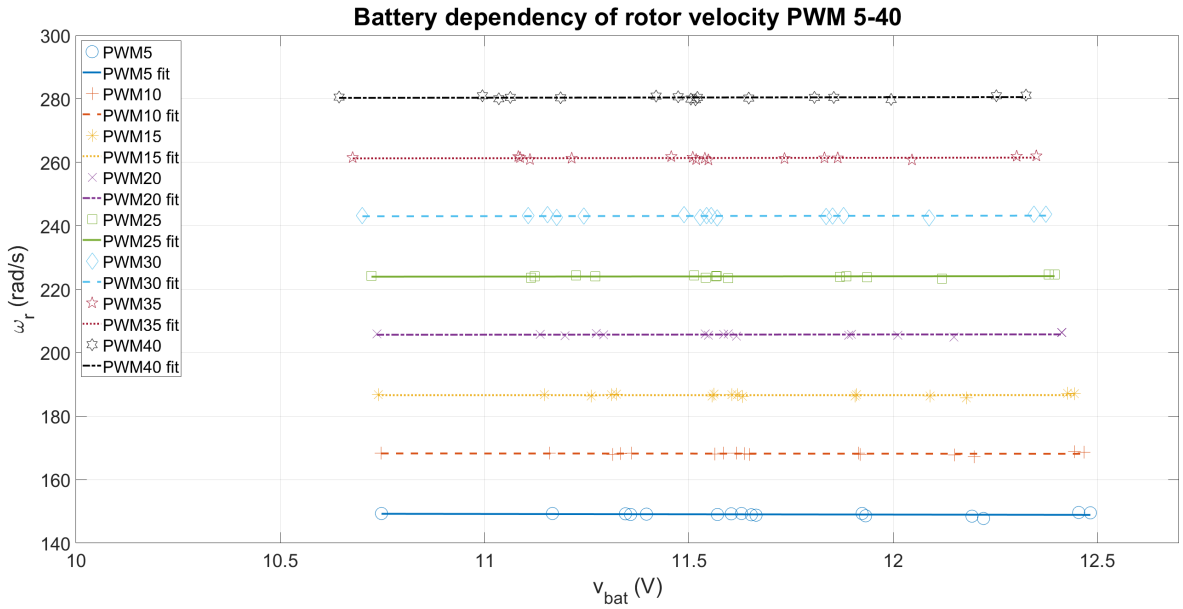


Figure 5-3: Battery dependency of rotor 4 velocity for PWM values ranging from 5 to 40.

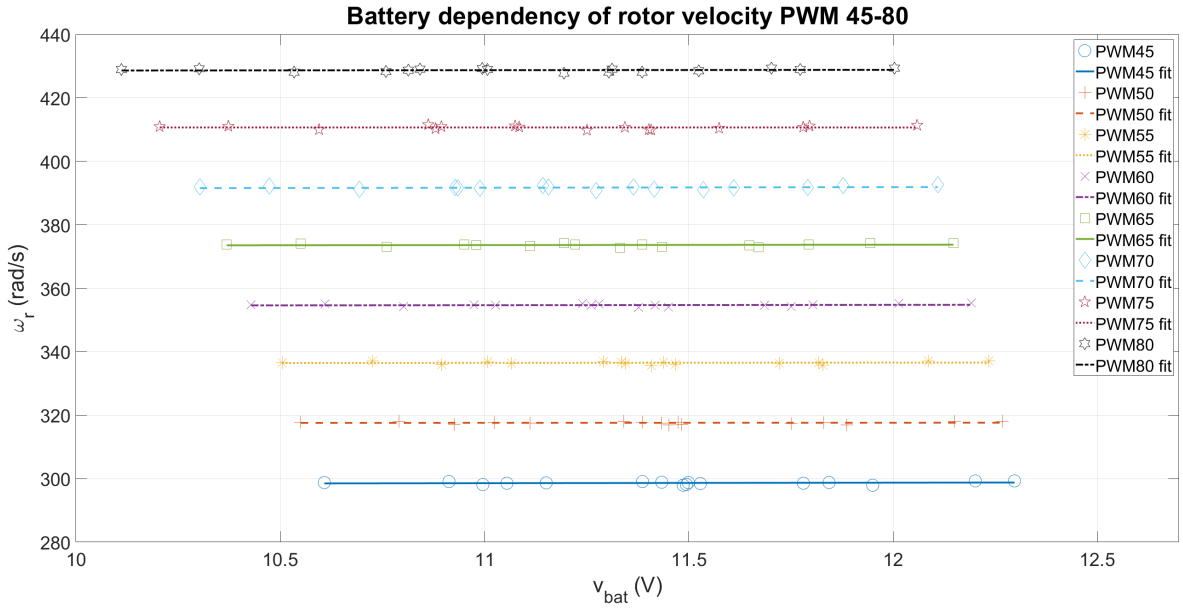


Figure 5-4: Battery dependency of rotor 4 velocity for PWM values ranging from 45 to 80.

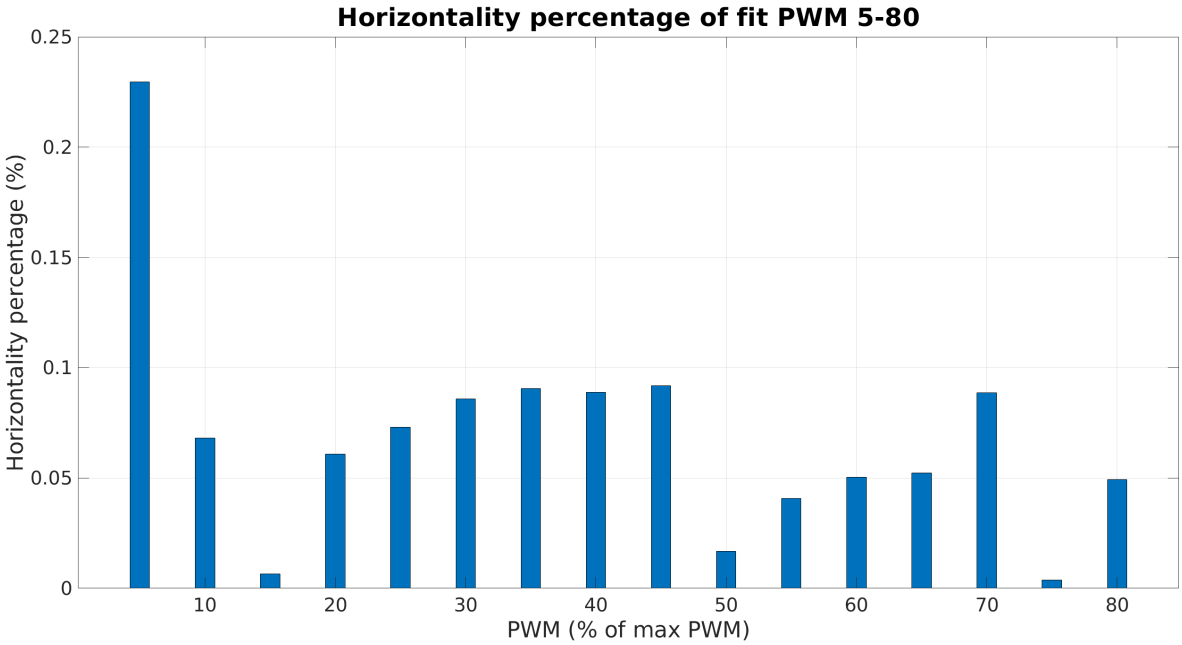


Figure 5-5: Horizontality indication of fits.

Since rotor rotational velocity is independent of battery voltage, the PWM-ω_r relation can be constructed using a single line, which is shown in Figure 5-6. The figure shows a clear linear relationship between PWM and rotational velocity. The relation for rotor i is given in Eq. (5-3). The rotors start turning from 0.2 PWM on, so the relation clearly holds from PWM 0.2 to 80. After 80, the rotational velocities are varying, as mentioned above. Figure 5-6 shows that the rotational velocity starts deviating from the linear fit from PWM 85 to 100. For these PWM setpoints the battery voltage dropped too much for the ESCs to compensate for it. In theory, the linear relationship should hold until PWM 100 with constant battery voltage. Given the unavailability of a lab power supply, a fully charged Akku-King battery is used to quickly go to PWM 100 and reduce to PWM 80 with steps of 5. This way, the battery is not loaded too much during the rest of the experiment and the battery is better able to keep a constant voltage at its output. The resulting measurements are called 'maxSpeed batA' and 'maxSpeed batA 2' in the figure and prove that the relationship between PWM and rotational velocity is linear for the whole PWM range when disregarding battery voltage. As mentioned, PWM values of 85 and higher are not important for the research goal in this project, so the fact that the battery voltage will drop for PWM values of 85 and higher does not require further consideration.

$$\omega_i = 3.7 \cdot \text{pwmM}_i + 130.9, \text{ for } \text{pwmM}_i \geq 0.2 \tag{5-3}$$

The previous results are all created using the aforementioned MATLAB/Simulink toolbox. The toolbox actually provides a percentage (0-100) of the maximum PWM value that can be given to the motors. It scales the percentage linearly to a range of 9 bits (0-511) being required by the ESC software. However, the AR.Drone 2.0 reads out and provides the motor PWM values on the *ardrone/navdata* topic as an 8-bit number (0-255). Given the fact that

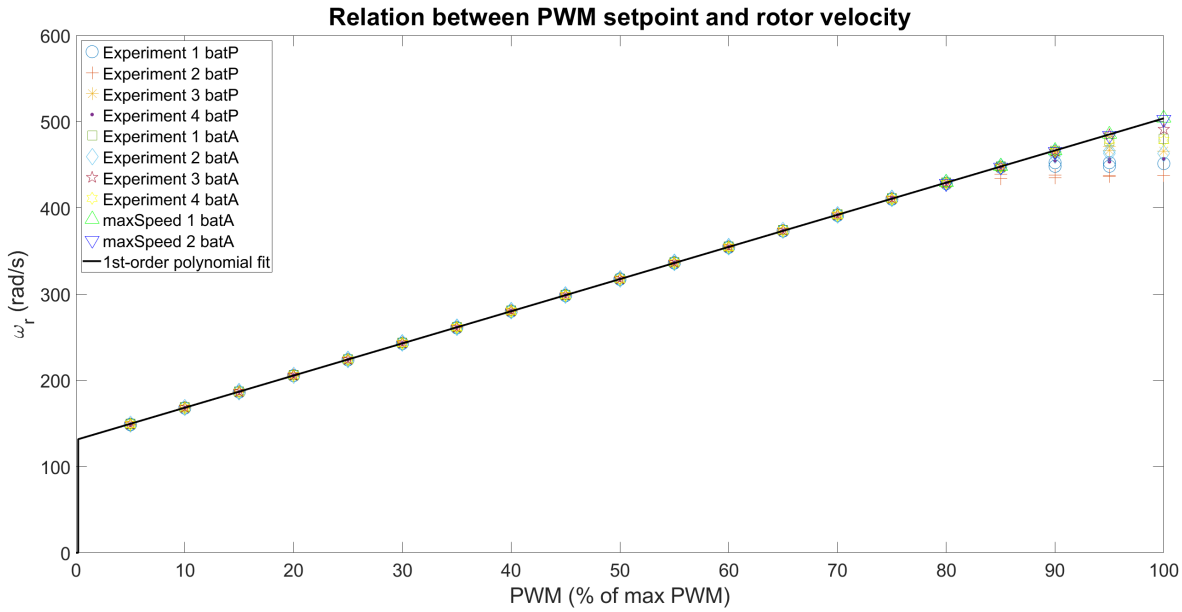


Figure 5-6: (Linear) relation between PWM setpoints and rotational velocity.

both methods use the same hardware, the assumption is that the provided PWM values are the eight most significant bits of the 9-bit control signal (giving half of the resolution) and thus the relation of PWM values would be a linear scaling. Using eight bits would make sense, because a *uint8* is a standard datatype and can easily be used to construct the message to be published on the *ardrone/navdata* topic.

Constructing the relation between both ways of representing the PWM value is performed by simultaneously measuring the rotational velocity of rotor 4 during hover flight and recording the PWM values on the *ardrone/navdata* topic. Using the experiments performed above, the rotor rotational velocity can be related to the PWM value the toolbox would give. This way, the PWM value on the *ardrone/navdata* topic is related to the toolbox PWM value. These measurements are performed on three points along the AR.Drone 2.0 PWM range by adding two times an equal mass to each of the four quadrotor landing gears. The measurements cover a range from 66% to 73% of maximum PWM. The region of operation in the considered experiment in this thesis is roughly between 50% and 80% PWM, of which the maximum and minimum values are very difficult to measure, because they rarely occur during non-aggressive flight. Figure 5-7 shows the theoretical relation between the two PWM values, together with the measured points. The points are located within 1% of the theoretical relation, so we can conclude that the PWM values are linearly related to each other and the assumption made above is valid.

The rotor rotational velocity can thus be obtained by recording the *ardrone/navdata* topic, reading out the motor PWM values and using the scaling in Eq. (5-4) to obtain the rotational velocity of rotor i.

$$\text{pwmA}_i = 2.55 \cdot \text{pwmM}_i \tag{5-4}$$

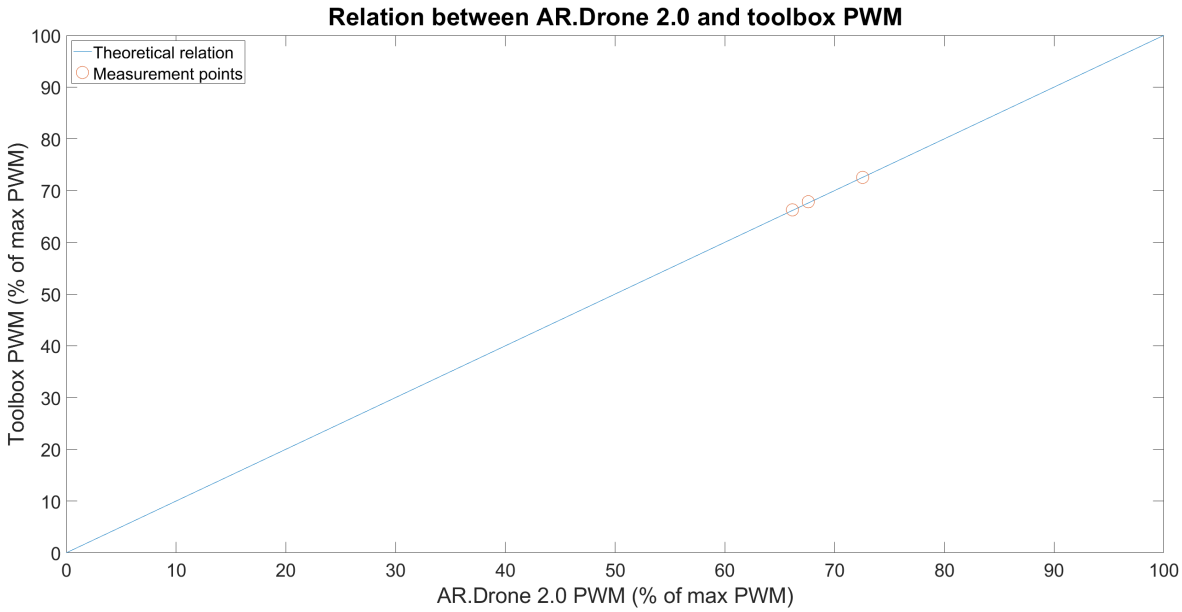


Figure 5-7: Relation between AR.Drone 2.0 and toolbox PWM values.

5-4-2 Thrust coefficient c_T

As given in Section 3-2-2, the relation between rotational velocity and generated thrust of rotor i can be described by the following quadratic relation:

$$T_i = c_T \cdot \omega_i^2 \tag{5-5}$$

Figure 5-8 shows the physical test setup used to quantize the thrust coefficient. As mentioned in [34], better methods exist to determine the coefficient. The same holds for the torque coefficient. However, due to limited resources, the setup in Figure 5-8 is built for this purpose. See Appendix B-3-1 for the setup design. To use this setup for thrust measurements, the quadrotor with indoor hull and fully charged battery was placed on the aluminium lever arm besides the table, such that the rotors can rotate in free air.

During the experiments, all motors receive increasing PWM setpoints from 0 to 100 and back in steps of 5. The rotational velocity of rotor 4 is read from the hand-tachometer in RPM and converted to $\mathrm{rad\,s^{-1}}$. The force sensor values are logged using a Data AcQuisition (DAQ) system and a Virtual Instrument (VI) in LabView 2018 and divided by four to obtain the generated thrust of one rotor. See Appendix B-3-2 for the complete measurement plan.

Using the experiment datasets given in Appendix B-3-3, Figure 5-9 is constructed. This figure shows the measured force values for the rotational velocities of rotor 4 belonging to PWM in the operating range 5-80 of three different experiments.

The figure indicates three different fits through the measurement points. The first fit is a quadratic fit, which is in accordance with the theory as shown above. However, we can clearly see the deviation from the experiment data. Therefore, a 2nd-order polynomial fit is created that better fits the data. However, this fit does not go through the point $(0, 0)$, which

Figure 5-8: Setup to determine thrust coefficient c_T.

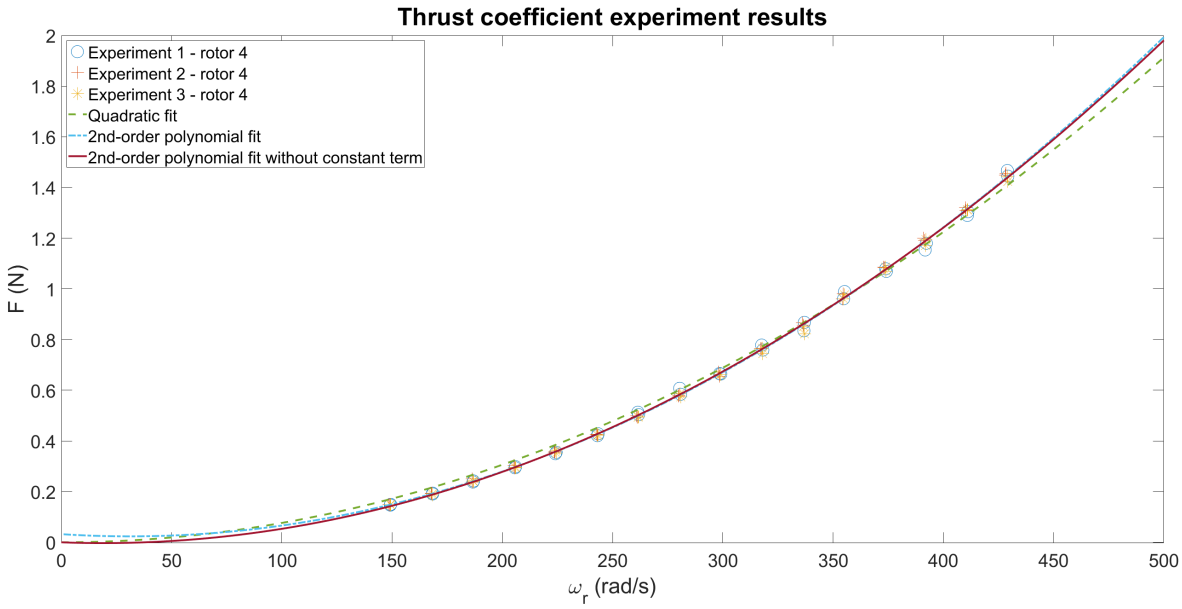


Figure 5-9: Relation between rotational velocity and thrust.

is more physically realistic, because the rotors will not generate thrust at all when standing still. Therefore, a third fit is constructed: a 2nd-order polynomial fit without constant term. This fit goes trough $(0, 0)$ and is still able to properly represent the experiment data.

The fits are given by Eq. (5-6) until (5-8), respectively:

$$T_{i,1} = 7.7 \cdot 10^{-6} \cdot \omega_i^2 \tag{5-6}$$

$$T_{i,2} = 9.0 \cdot 10^{-6} \cdot \omega_i^2 - 5.6 \cdot 10^{-4} \cdot \omega_i + 3.2 \cdot 10^{-2} \tag{5-7}$$

$$T_{i,3} = 8.6 \cdot 10^{-6} \cdot \omega_i^2 - 3.2 \cdot 10^{-4} \cdot \omega_i \tag{5-8}$$

Table 5-3 provides the Mean Squared Error (MSE) of the three fits. The 2nd-order polynomial fits indeed better represent the experiment data with respect to the quadratic fit. The MSE of the 2nd-order polynomial fit without constant term is a bit higher than the 2nd-order polynomial fit, but it makes physically much more sense, so this fit is chosen to be used. It should be noted that the thrust is thus not described using a single thrust coefficient, but using two coefficients that constitute a 2nd-order polynomial fit without constant term: Eq. (5-8).

Table 5-3: MSE of thrust fit 1, 2 and 3.

Fit	MSE
1	$5.8 \cdot 10^{-4}$
2	$1.2 \cdot 10^{-4}$
3	$1.3 \cdot 10^{-4}$

Validation

As can be concluded from Section 3-2-2, the ω_i-thrust relation in Eq. (5-8) can be rewritten as a 2nd-order polynomial PWM-thrust relation given in Eq. (3-13). By hovering the quadrotor several times and recording the PWM values for the Parrot as well as the Akku-King batteries, two operating points (caused by the different masses of each battery) can be constructed.

Figure 5-10 shows the PWM-thrust hovering operating points for both Parrot and Akku-King battery, as well as the PWM-thrust relation of this work and the work presented in [34] and [64]. The figure clearly indicates that none of the experimentally-determined PWM-thrust relations goes through the operating points. The deviation of the relation in this work is probably caused by the fact that the sensors were calibrated one month before the experiments took place, due to COVID-19 circumstances. The limitations of the relations from [34] and [64] have already been discussed above.

To overcome the deviations in the hovering operating points, the PWM-thrust relation is estimated, based on the two operating points. The fourth relation in Figure 5-10 represents the corresponding fit. Since this fit goes exactly through both operating points, while having a similar form of PWM-thrust relation as given before, this fit is going to be used to construct the noise and filter results in Chapters 6 and 7.

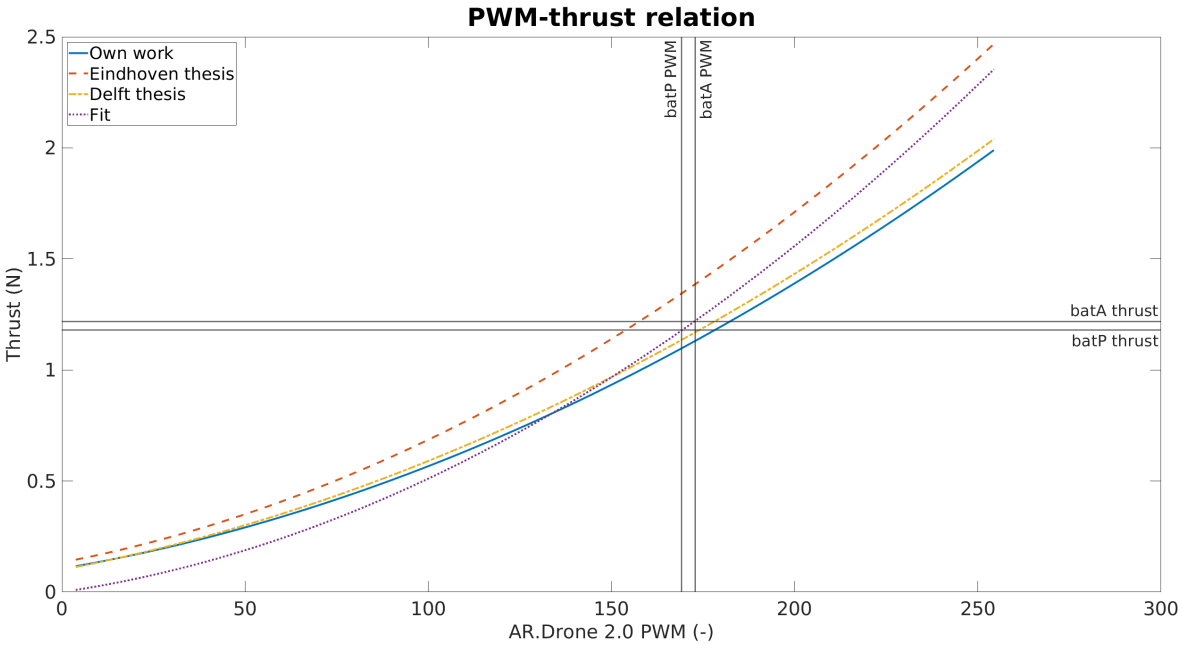


Figure 5-10: PWM-thrust hovering operating points and PWM-thrust relation from this section (Own work), [34] (Delft thesis), [64] (Eindhoven thesis) and fitted (Fit) using both operating points. The data used to construct this figure is presented in Appendix B-3-4. This appendix also shows the same figure with linearizations of these relations around the operating points.

5-4-3 Torque coefficient c_Q

As can be concluded from Section 3-4, the torque coefficient is not necessary for analyzing the noise and filter results in Chapters 6 and 7. However, for the sake of completeness and the possibility to use this value in future research, this section describes the determination of the torque coefficient.

The relation between rotational velocity and generated torque of rotor i can be described by the following quadratic relation:

$$\tau_i = c_Q \cdot \omega_i^2 \tag{5-9}$$

For determining the torque coefficient a similar approach is taken as for determining the thrust coefficient. Figure 5-11 shows the setup used to estimate the torque coefficient. The design of this setup is explained in Appendix B-4-1. Using this setup, the quadrotor is also placed above free air.

For the AR.Drone 2.0 holds that rotor 1 and 3 (diagonally located rotors) produce a torque in the same direction, being opposite to the torque produced by rotor 2 and 4. Therefore, separate torque experiments are conducted for rotor 1 and 3 and for rotor 2 and 4. In these experiments, two rotors turn at the same time in order to let the total torque be directed around the theoretical COM of the quadrotor, which is desired with this torque setup. Both motors receive the same PWM setpoints as in case of the thrust setup (i.e. 0 to 100 and back in steps of 5). The data is logged in the same way as for determining the thrust coefficient,

Figure 5-11: Setup to determine torque coefficient c_Q.

except that the measured force value is now divided by two, not by four, and multiplied by the arm length to get the corresponding torque value. The complete measurement plan is given in Appendix B-4-2.

Using the data in Table B-7, Figure 5-12 is created. This figure contains exactly the same content as Figure 5-9, but now for the torque coefficient using four different experiments: two using rotor 1 and 3 and two using rotor 2 and 4.

Again, three fits are constructed: a quadratic fit (given by Eq. (5-10)), a 2nd-order polynomial fit (given by Eq. (5-11)) and a 2nd-order polynomial fit without constant term (given by Eq. (5-12)). Their respective MSE values are reported in Table 5-4. In this case, the quadratic fit represents the data well too and the 2nd-order polynomial fit without constant term even has a lower MSE value than the 2nd-order polynomial fit. Eq. (5-12) is thus chosen to be used to describe the torque production of the rotors.

$$\tau_{i,1} = 2.2 \cdot 10^{-7} \cdot \omega_i^2 \tag{5-10}$$

$$\tau_{i,2} = 2.6 \cdot 10^{-7} \cdot \omega_i^2 - 2.3 \cdot 10^{-5} \cdot \omega_i + 1.8 \cdot 10^{-3} \tag{5-11}$$

$$\tau_{i,3} = 2.4 \cdot 10^{-7} \cdot \omega_i^2 - 9.9 \cdot 10^{-6} \omega_i \tag{5-12}$$

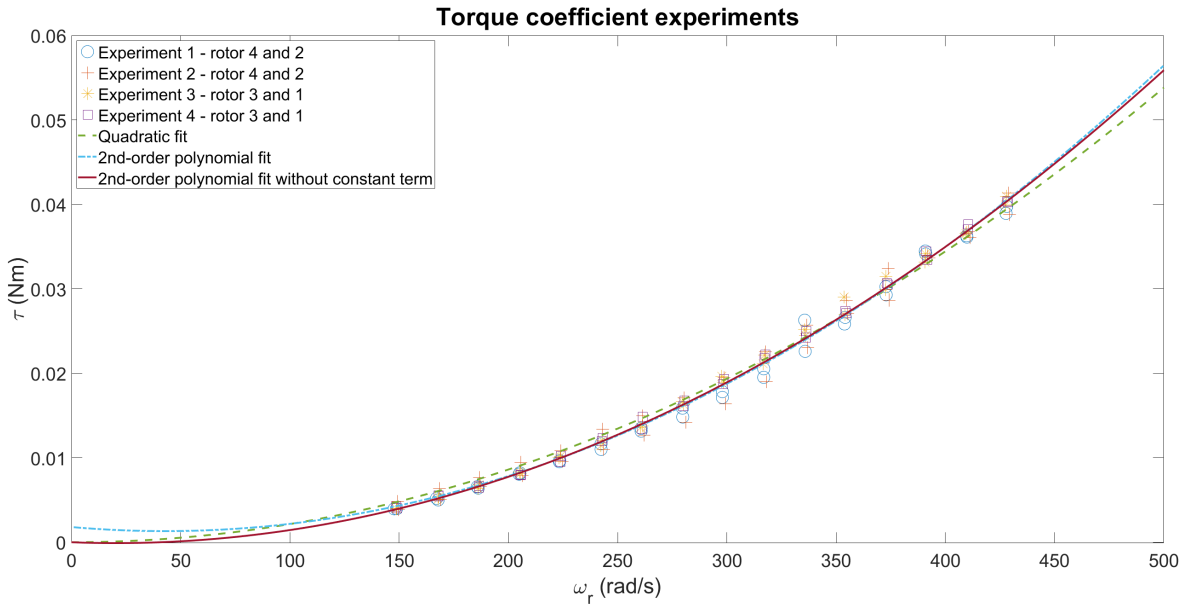


Figure 5-12: Relation between rotational velocity and torque.

Table 5-4: MSE of torque fit 1, 2 and 3.

Fit	MSE
1	$1.1 \cdot 10^{-6}$
2	$7.3 \cdot 10^{-7}$
3	$7.2 \cdot 10^{-7}$

Chapter 6

Noise analysis

As discussed in Chapter 2, the nature of the noises plays an important role in the performance of the DEM filter. So before presenting the filter results in the next chapter, this chapter analyzes the noises to check whether the coloured noise assumption made in Chapter 2 is correct. This section firstly gives the definition of process noise and measurement noise in Section 6-1. Secondly, in order to determine the coloured noise properties (i.e. the covariance matrix and smoothness) of the noises, the noise signals should be known. However, it will be shown that in practice these signals cannot be determined. Therefore, an assumption on the measurement noise is made in Section 6-2. Thirdly, the coloured noise properties of both measurement and process noise are constructed in Sections 6-2-1 and 6-2-2, respectively. These sections also validate the Laplace approximation from Chapter 2 by showing that the noise signals are Gaussian distributed. Fourthly, Section 6-3 discusses the assumption that coloured noise is produced by filtering a white noise sequence with a Gaussian filter. Finally, the idea and validation of another smoothness estimation method is presented. This method will shortly be described in Section 6-4 and needs further research.

6-1 Noise definition

The general LTI dynamic model description is given by Eq. (2-34) and (2-35) in Section 2-3-1 in which $\boldsymbol{w}$ and $\boldsymbol{z}$ represent the process and measurement noise, respectively. In discrete form, these equations yield:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= A_d \boldsymbol{x}_k + B_d \boldsymbol{u}_k + \boldsymbol{w}_k \\ \boldsymbol{y}_k &= C_d \boldsymbol{x}_k + \boldsymbol{z}_k \end{aligned} \tag{6-1}$$

where $\boldsymbol{x}_k$, $\boldsymbol{u}_k$, $\boldsymbol{y}_k$, $\boldsymbol{w}_k$ and $\boldsymbol{z}_k$ represent the state, input, output, process noise and measurement noise values at time instance k, and A_d, B_d and C_d denote the discretized state, input and output matrices, respectively. As mentioned in Section 4-1-2, the sampling frequency for discretization is chosen to be 120 Hz.

The process noise accounts for the error made in modelling the states. For a discretized LTI model in general, this error consists of the following elements [67]:

- *Mis-modelled system dynamics*
  For the model considered in this thesis, the mis-modelled dynamics include the non-ideal relation between PWM value and generated thrust as well as other (nonlinear) effects that are not captured in the LTI model.
- *The existence of a hidden state that is not captured in the model*
  The model considered in this thesis does not capture motor dynamics and their influence on rotor dynamics. The dynamics of brushless Direct Current (DC) motors are often modelled by a 1st-order system [68], thereby adding an extra state to the system.
- *The approximations made to simplify the model (e.g. using a Taylor expansion to describe the dynamics in a linear form)*
  Several approximations are used to construct the LTI model from Chapter 3. These include neglecting Coriolis terms and the disturbance term and setting the rotation matrix for rotational dynamics equal to identity (by the small angle assumption).
- *The discretization of time*
  Time discretization introduces a finite accuracy to the state and input values and thus to the process noise value.

The measurement noise accounts for mis-modelled measurement dynamics (i.e. sensor inaccuracies and electrical non-idealities on data transport wires).

In the characterization of the noises in the next section, these individual aspects are not taken into account. They are listed here for the sake of completeness and to understand what components the noises consist of.

6-2 Noise characterization

In contrast to the information provided to the filters, enough sensor information is available to calculate the noises. Given an initial state and writing out the discrete state equation for the first few samples, we obtain:

$$\boldsymbol{x}_1 = A_d\boldsymbol{x}_0 + B_d\boldsymbol{u} + \boldsymbol{w}_0 \tag{6-2}$$
$$\boldsymbol{x}_2 = A_d\boldsymbol{x}_1 + B_d\boldsymbol{u}_1 + \boldsymbol{w}_1 \tag{6-3}$$
$$= A_d^2\boldsymbol{x}_0 + A_dB_d\boldsymbol{u}_0 + A_d\boldsymbol{w}_0 + B_d\boldsymbol{u}_1 + \boldsymbol{w}_1 \tag{6-4}$$
$$\boldsymbol{x}_3 = A_d\boldsymbol{x}_2 + B_d\boldsymbol{u}_2 + \boldsymbol{w}_2 \tag{6-5}$$
$$= A_d^3\boldsymbol{x}_0 + A_d^2B_d\boldsymbol{u}_0 + A_d^2\boldsymbol{w}_0 + A_dB_d\boldsymbol{u}_1 + A_d\boldsymbol{w}_1 + B_d\boldsymbol{u}_2 + \boldsymbol{w}_2 \tag{6-6}$$
$$\vdots$$

The previous expression can be written in compact form as:

$$\boldsymbol{x}_k = A_d^k \boldsymbol{x}_0 + \sum_{i=0}^{k-1} A_d^{k-i-1} B_d \boldsymbol{u}_i + \sum_{i=0}^{k-1} A_d^{k-i-1} \boldsymbol{w}_i \tag{6-7}$$

The compact form of the discrete state equation can be trivially converted to the compact, discrete form of the output equation:

$$\boldsymbol{y}_k = C_d A_d^k \boldsymbol{x}_0 + C_d \sum_{i=0}^{k-1} A_d^{k-i-1} B_d \boldsymbol{u}_i + C_d \sum_{i=0}^{k-1} A_d^{k-i-1} \boldsymbol{w}_i + \boldsymbol{z}_k \tag{6-8}$$

Given the fact that the initial state is known and the evolving state and input values over time can be calculated using the model, the first two terms in Eq. (6-8) are known and we are left with the noise dynamics. The complete observed noise dynamics $\boldsymbol{y}_n$ can be derived from the above equation and equals:

$$\boldsymbol{y}_{n,k} = C_d \sum_{i=0}^{k-1} A_d^{k-i-1} \boldsymbol{w}_i + \boldsymbol{z}_k \tag{6-9}$$

Writing out above equation for the first five samples in matrix form, one obtains:

$$\begin{bmatrix} \boldsymbol{y}_{n,1} \\ \boldsymbol{y}_{n,2} \\ \boldsymbol{y}_{n,3} \\ \boldsymbol{y}_{n,4} \\ \boldsymbol{y}_{n,5} \end{bmatrix} = \begin{bmatrix} C_d & 0 & 0 & 0 & 0 \\ C_d A_d & C_d & 0 & 0 & 0 \\ C_d A_d^2 & C_d A_d & C_d & 0 & 0 \\ C_d A_d^3 & C_d A_d^2 & C_d A_d & C_d & 0 \\ C_d A_d^4 & C_d A_d^3 & C_d A_d^2 & C_d A_d & C_d \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_0 \\ \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \boldsymbol{w}_3 \\ \boldsymbol{w}_4 \end{bmatrix} + \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \\ \boldsymbol{z}_3 \\ \boldsymbol{z}_4 \\ \boldsymbol{z}_5 \end{bmatrix} \tag{6-10}$$

As can be seen, the above equation does not allow to trivially retrieve $\boldsymbol{w}$ and $\boldsymbol{z}$ as separate noise signals, since it is an underdetermined set of equations.

In literature, there are existing methods to estimate the the noise covariance matrices (see e.g. [69] and [70]). However, in this case we also need a smoothness value representing the colouredness of the noises. Since we have little experience with the smoothness value in our research group, we want to gain more insight in this value. Therefore, the next sections will take another approach to construct the process and measurement noise covariance matrices and derive a smoothness value for each of the noise signals. These smoothness values will be discussed in Chapter 7.

6-2-1 Measurement noise

The measurement noise characteristics can be obtained relatively easily by measuring the output signal when the quadrotor is standing on the ground. Here, we make use of the assumption that the measurement noise characteristics are the same for the case where the quadrotor is standing on the ground and the case where the quadrotor is flying in the air.

Since the output is defined as the roll angle measured by the OptiTrack system, the assumption is verified by looking at the error in marker position estimation by the system as reported in

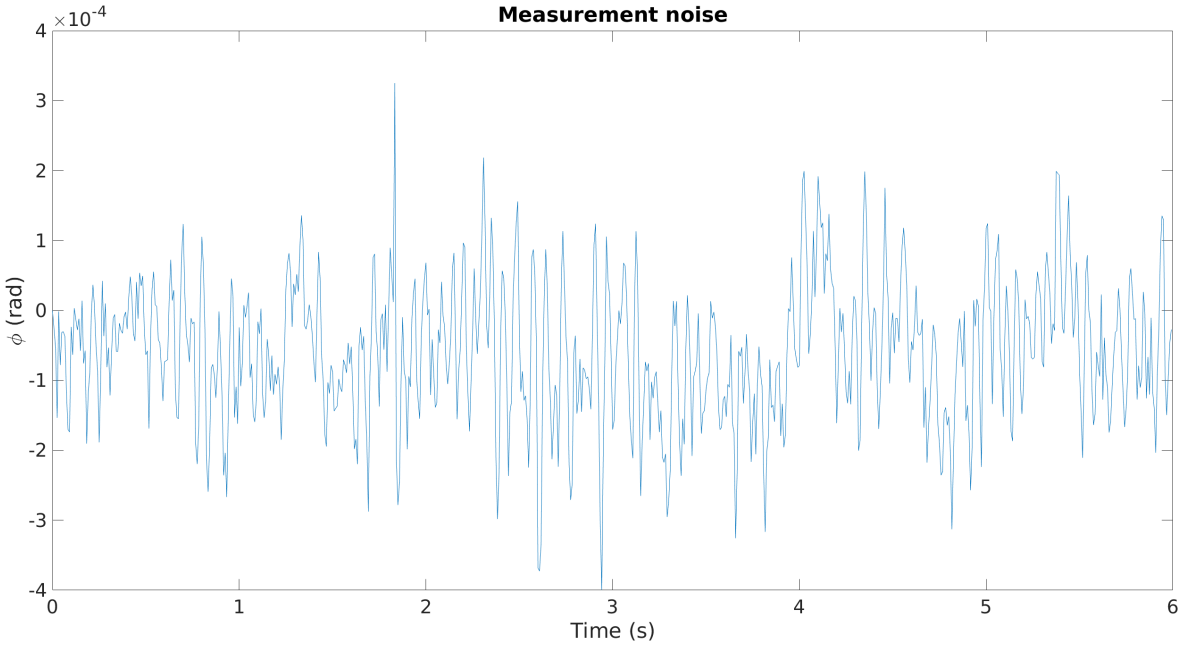


Figure 6-1: Measurement noise, recorded during first six seconds of experiment. This signal has a standard deviation of $\sigma_z = 9.92 \cdot 10^{-5}$ rad.

Motive:Tracker. This error stays the same in aforementioned cases, so we can assume that the characteristics of the noise present on the roll angle data also stay the same in both cases.

In order to improve the measurement noise characteristics calculation, the experiment is designed in such a way that the quadrotor was first standing a few seconds on the ground before taking off. This way, the roll angle data could be recorded using exactly the same experiment setup as used to record the flight data used in the next section. The characteristics of the measurement noise thus properly represent the measurement noise characteristics of the output as used in the next section.

Figure 6-1 shows the roll angle data during the first six seconds of the experiment. Take care of the scaling of amplitude values. Using the MATLAB function `std` the standard deviation of this signal is determined to be $9.92 \cdot 10^{-5}$ rad. The measurement noise covariance matrix equals the measurement variance and thus is given by:

$$\Sigma_z = \sigma_z^2 = 9.83 \cdot 10^{-9} \tag{6-11}$$

The measurement noise is approximately three orders of magnitude smaller than the peak values of the output itself (which are in the order of 10^{-1} rad). This fact will be used in the next section to derive the process noise characteristics.

Figure 6-2 shows the measurement noise distribution. Furthermore, it shows a Gaussian fit on this distribution. The same is displayed for the 1st- and 2nd-order derivatives of the measurement noise. From this figure can be concluded that the measurement noise and its derivatives are Gaussian distributed, thereby validating the Laplace approximation in Section 2-2-2. Appendix D-1 shows that the same holds for the higher-order derivatives until at least order 6 (as used in the DEM filter in the next chapter).

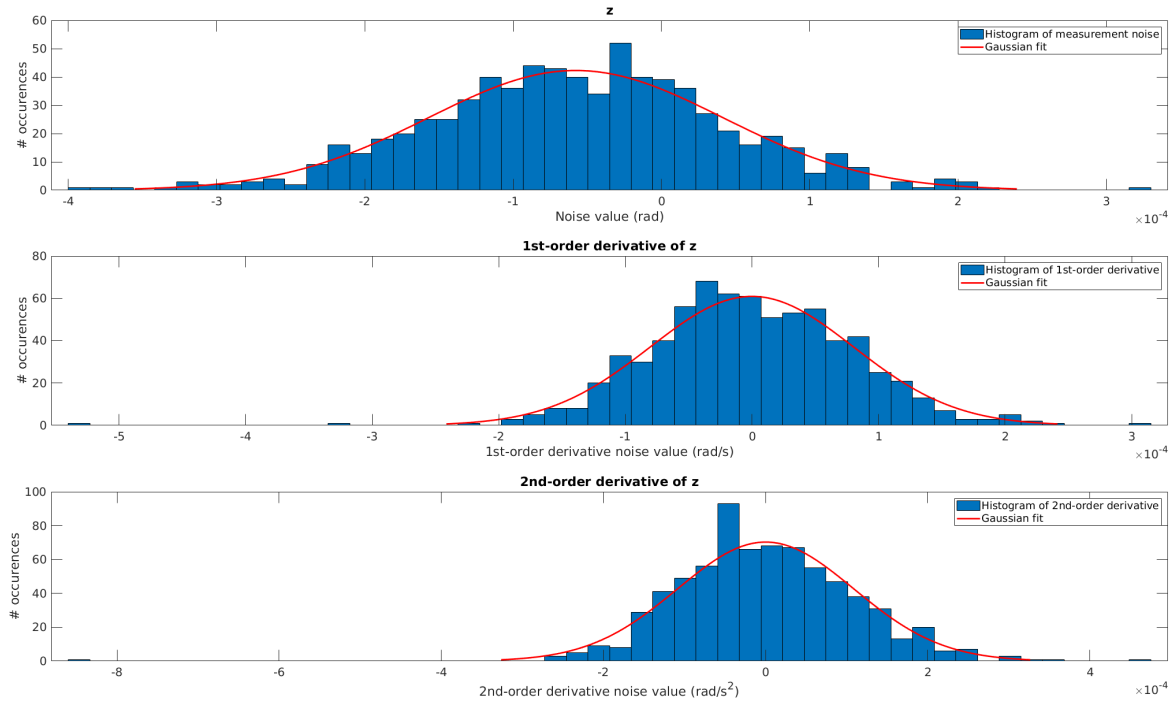


Figure 6-2: Distributions of z and its first two derivatives, together with their corresponding Gaussian fit.

As both process and measurement noise are described as coloured noise, we want to know the smoothness value of the measurement noise. As explained in Section 2-3-2, the smoothness value can be obtained by fitting the Gaussian filter autocorrelation to the measurement noise autocorrelation. Figure 6-3a shows the autocorrelation function of the measurement noise with a fitted Gaussian filter autocorrelation. The smoothness, or kernel width, of the Gaussian filter that makes the Gaussian filter autocorrelation approximate the measurement noise autocorrelation the best is shown in Figure 6-3b and equals:

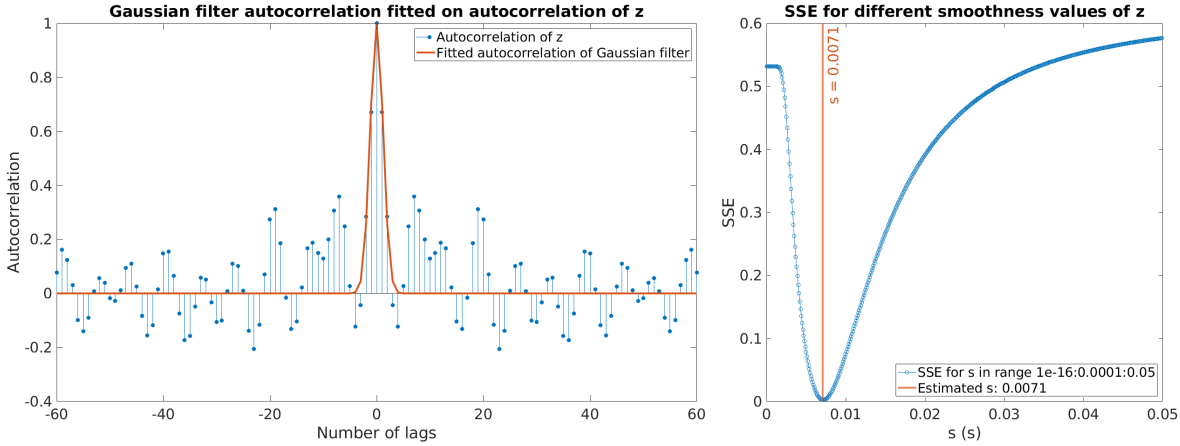
$$s_z = 7.10 \cdot 10^{-3} \tag{6-12}$$

As indicated in Section 2-3-2, the lower the smoothness value, the more the Gaussian filter autocorrelation approximates a delta function, the less correlation can be found in the measurement noise and thus the less coloured the noise is.

6-2-2 Process noise

In contrast to the measurement noise, the process noise only takes non-zero values if the quadrotor is actually moving. Therefore, we cannot take a zero measurement to derive its characteristics. Furthermore, as explained in Section 6-2, we cannot directly compute the process noise from the output samples we get during flight. Therefore, this section will show that we can calculate the process noise using the following equation:

$$\boldsymbol{w}_k = \boldsymbol{x}_{k+1} - A_d \boldsymbol{x}_k - B_d \boldsymbol{u}_k \tag{6-13}$$



(a) Autocorrelation **(b)** SSE

Figure 6-3: The optimal Gaussian filter of which the autocorrelation fits the autocorrelation of z in terms of SSE has kernel width $s_z = 7.10 \cdot 10^{-3}$ s.

So in order to calculate $\boldsymbol{w}$, we need state values $\boldsymbol{x}_{k+1}$ and $\boldsymbol{x}_k$ and input values $\boldsymbol{u}_k$. The inputs are directly measured. For the states, we can use the best sensor sources in our experimental setup. The most reliable roll angle source is the OptiTrack system. The only source for the roll rate is the quadrotor gyroscope. It will be shown that the noise of both sensors is very small compared to the state values, such that we can use these signals to calculate the process noise.

The noise of the measured roll angle has already been discussed in the previous section. The measurement noise is three orders of magnitude smaller than the measured output during the experiment: $z \ll y$. It means that we can take the output to represent the roll angle. Since we are mainly interested in the hidden state estimate of the filters, we can safely use this value measurement to derive the process noise characteristics.

The noise of the roll rate measured by the quadrotor gyroscope is given in Figure 6-4. We can draw a similar conclusion as for the roll angle measurement, based on data recorded when the quadrotor is standing on the ground. The measured values during the experiment are in the order of 10^0, whereas the noise values are in the order of 10^{-4}, meaning that also in this case the noise is three orders of magnitude smaller than the measured values during the experiment. Therefore, we will use the roll rate measurement of the quadrotor gyroscope to derive the process noise characteristics.

Figure 6-5 shows the process noise signals. The figure shows that $w_1 = w_\phi$ has a relatively low value with respect to the roll angle values in the experiment, while $w_2 = w_{\dot{\phi}}$ is given in the same order of magnitude as the roll rate during the experiment. The process noise covariance matrix is calculated using the MATLAB command `cov` and equals:

$$\Sigma_w = \begin{bmatrix} \sigma^2_{w_1} & \sigma^2_{w_1 w_2} \\ \sigma^2_{w_2 w_1} & \sigma^2_{w_2} \end{bmatrix} = \begin{bmatrix} 9.10 \cdot 10^{-7} & 3.62 \cdot 10^{-5} \\ 3.62 \cdot 10^{-5} & 1.43 \cdot 10^{-2} \end{bmatrix} \tag{6-14}$$

Figure 6-6 and 6-7 show the distribution of w_1 and w_2 and their 1st- and 2nd-order derivatives. The Gaussian fit in the figures shows that w_1 and its 1st- and 2nd-order derivatives and

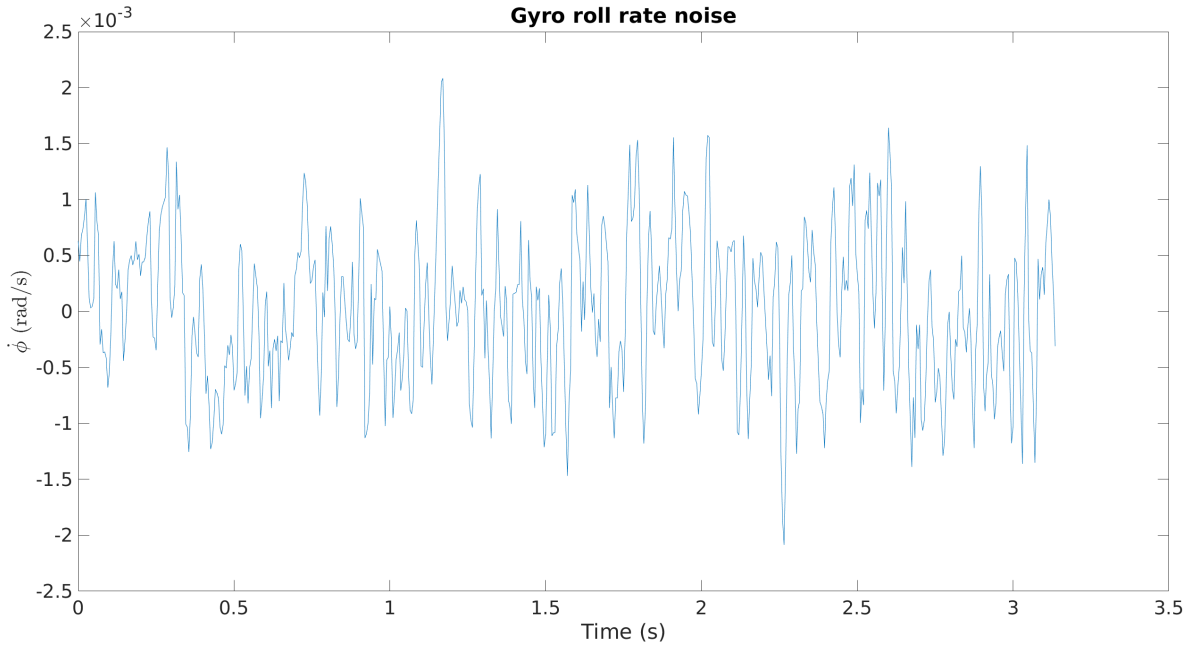


Figure 6-4: AR.Drone 2.0 gyroscope roll rate noise, recorded when drone was standing on the ground in the experiment. This signal has a standard deviation of $\sigma_{\dot{\phi}} = 6.73 \cdot 10^{-4}$ rad s^{-1}.

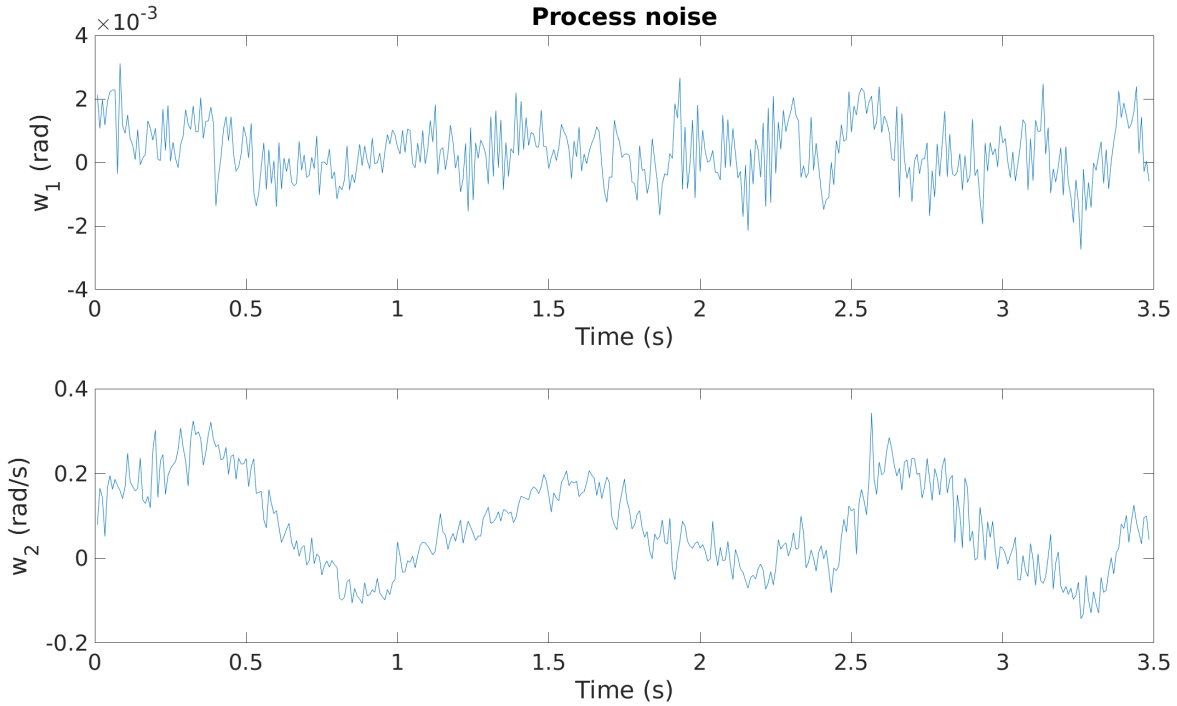


Figure 6-5: Process noise, calculated using OptiTrack roll angle, AR.Drone 2.0 gyroscope roll rate and AR.Drone 2.0 PWM values recorded during the experiment.

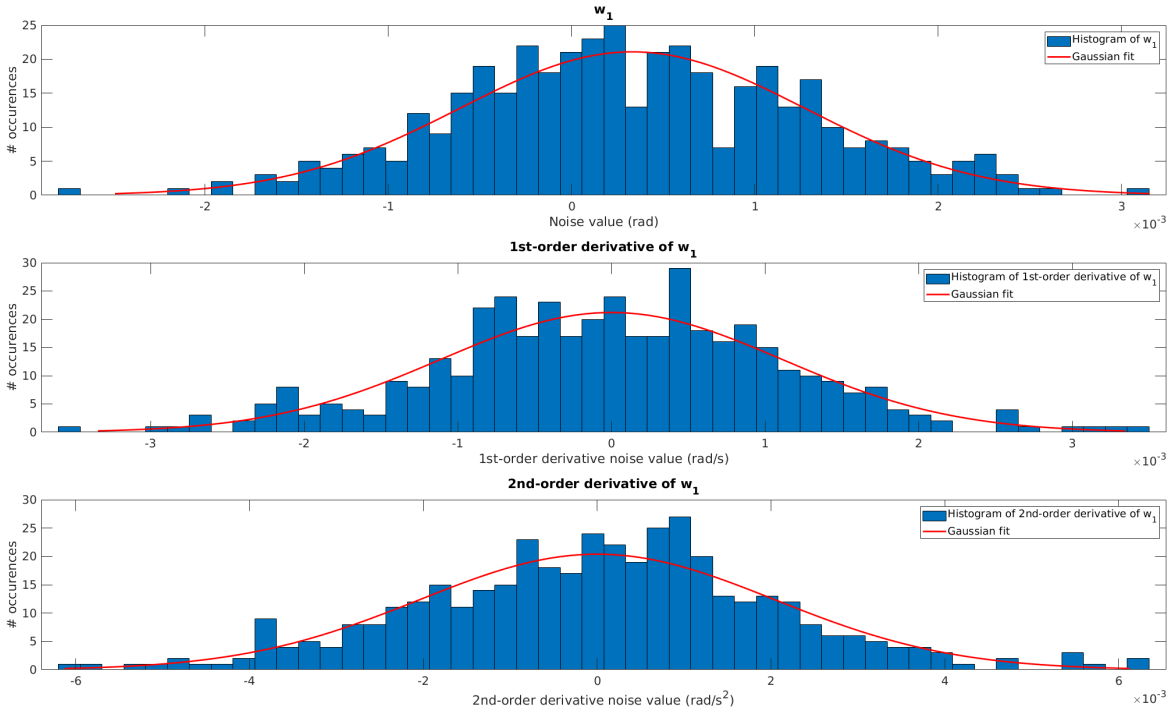


Figure 6-6: Distributions of w_1 and its first two derivatives, together with their corresponding Gaussian fit.

the 1st- and 2nd-order derivatives of w_2 are Gaussian distributed, thereby validating the Laplace approximation made in Section 2-2-2. However, w_2 itself does not follow a Gaussian distribution and seems to have two peak values around $w_2 = 0.04$ rad s^{-1} and $w_2 = 0.18$ rad s^{-1}. The first peak value is introduced by the fact that w_2 does not follow a perfect sine wave. This is caused by the fact that, instead of a minimum, a small peak occurs around $t = 2.3$ s. Furthermore, w_2 stays at that value for a while at the end. The second peak in the distribution is caused by the fact that the second peak in w_2 is relatively low and approximately equal to 0.18 rad s^{-1}. Appendix D-2 shows that the Laplace approximation is also valid for the higher-order derivatives up to and including at least order 6.

The periodic behaviour in w_2 can be related to different causes. Since w_2 is calculated using the gyroscope roll rate and the motor PWM values, it can be concluded that the motor PWM values do not properly reflect the thrust generated by each rotor to induce the measured roll rate. This mismatch is caused by the air flows around the quadrotor. The air flows around the quadrotor are produced by rotor blade movements as well as the external wind source and influenced by the lab environment (including walls and other objects). When looking at the experiment video, the air flows cause the quadrotor to make relatively big movements (including movements in positive and negative roll direction) with a periodicity of approximately 1 s. This corresponds to the periodicity of the process noise in Figure 6-5.

The effect of these air flows can be explained by four different phenomena. Firstly, air flows perpendicular to the plane in which each rotor is rotating have impact on the generated thrust by that rotor [71]. This is not accounted for in the system model. Secondly, a PWM value does not directly represent a rotor rotational velocity. Therefore, the system model would be

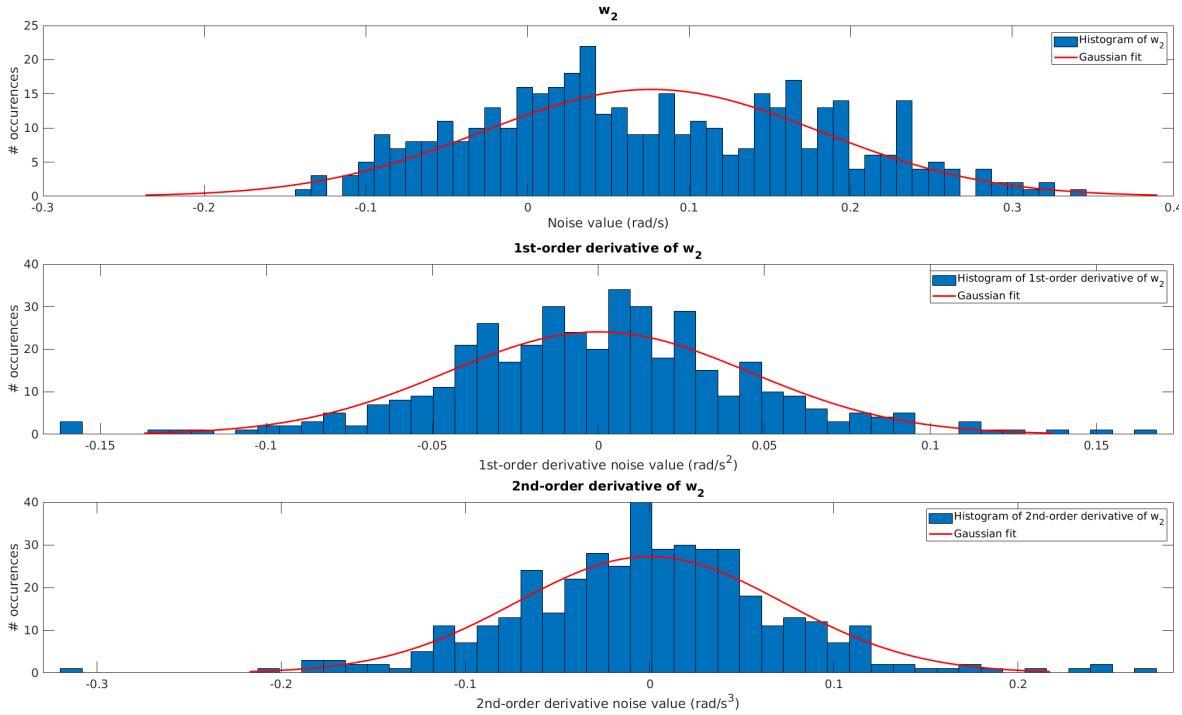


Figure 6-7: Distributions of w_2 and its first two derivatives, together with their corresponding Gaussian fit.

more accurate when including the motor and rotor dynamics. In theory, this should be done for each motor-rotor combination separately, since each combination has slightly different characteristics. Thirdly, the rotor blades are not exactly the same, so it would be better to include a thrust coefficient for each rotor separately. Finally, system linearization causes the PWM-thrust curve to lie below the nonlinear PWM-thrust curve. The error between the two curves is bigger for the lowest PWM values than for the highest PWM values measured, as shown in Figure B-10 in Appendix B-3-4. To track down the influence of each phenomenon described above would be a complex task, which is outside the scope of this thesis. It is therefore considered as future work.

In order to determine the smoothness of w_2, the noise aspect described above is not desired, since the kind of smoothness defined above is acting on a whole different time frame than the smoothness leveraged by the DEM filter. Therefore, we would like to remove this behaviour by fitting a function to this 'global' behaviour and calculating the residual. This is shown in Figure 6-8.

Using w_1 and the residual of w_2, $w_{2,res}$, we can fit a Gaussian filter autocorrelation on the process noise autocorrelations and derive their smoothness values by minimizing the SSE between the autocorrelations.

Figures 6-9a and 6-9b show the result for w_1. The smoothness for which the Gaussian filter autocorrelation best fits the autocorrelation of w_1 is given by:

$$s_{w_1} = 9.30 \cdot 10^{-3} \tag{6-15}$$

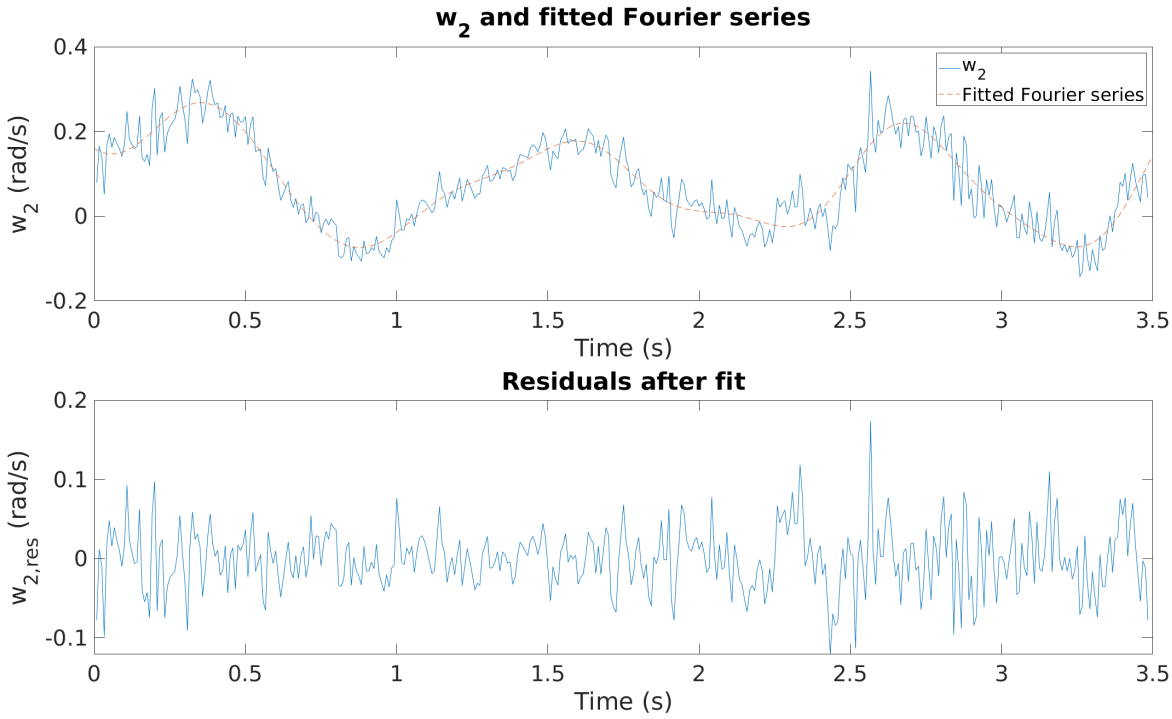
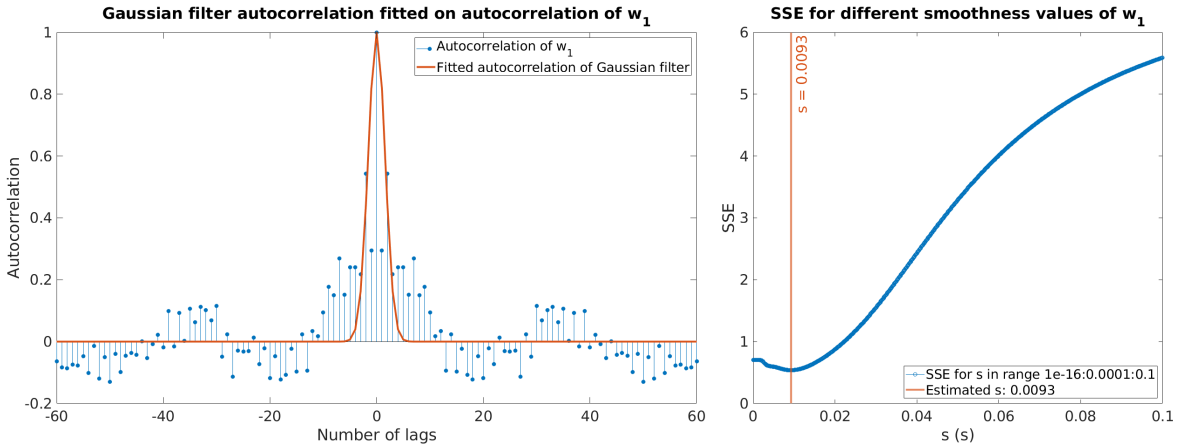
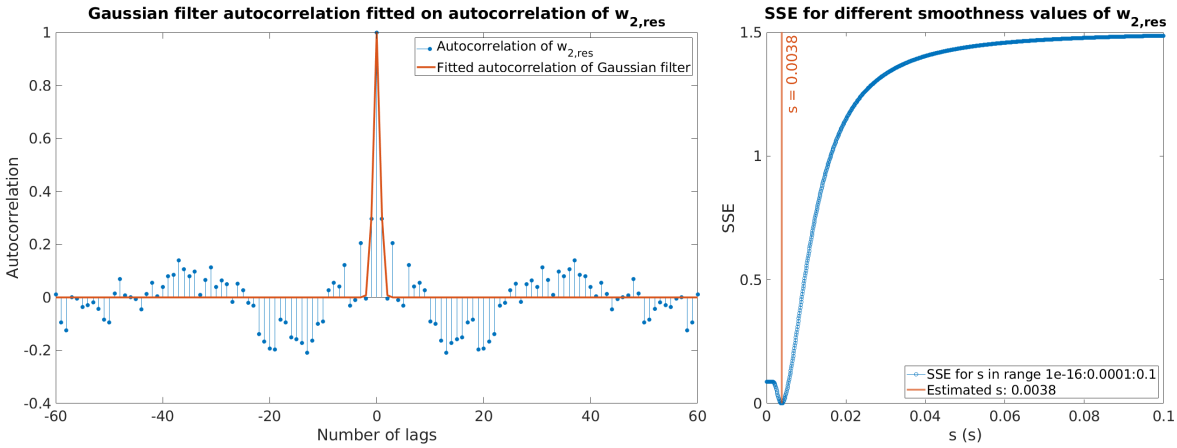


Figure 6-8: Process noise w_2, together with its Fourier fit and residuals after fitting.



(a) Autocorrelation **(b)** SSE

Figure 6-9: The optimal Gaussian filter of which the autocorrelation fits the autocorrelation of w_1 in terms of SSE has kernel width $s_{w_1} = 9.30 \cdot 10^{-3}$ s.



(a) Autocorrelation **(b)** SSE

Figure 6-10: The optimal Gaussian filter of which the autocorrelation fits the autocorrelation of $w_{2,res}$ in terms of SSE has kernel width $s_{w_{2,res}} = 3.80 \cdot 10^{-3}$ s.

The result for $w_{2,res}$ is shown in Figures 6-10a and 6-10b. The smoothness value for $w_{2,res}$ is thus given by:

$$s_{w_{2,res}} = 3.80 \cdot 10^{-3} \tag{6-16}$$

As mentioned before, we would expect the process noise (containing unmodelled system dynamics) to have higher smoothness values than the measurement noise (only containing sensor noise). However, it can be concluded that w_1 and $w_{2,res}$ have smoothness values in the same range as z. w_1 is defined as the system output and therefore contains a very limited amount of unmodelled system dynamics (mostly related to linearization and discretization of time). w_2 contains a lot of smoothness, but this smoothness is acting on a larger timescale and not giving useful information about the noise derivatives and can therefore be ignored. It is thus recommended to design an experiment in which w_2 has smoothness on a timescale that can be leveraged by the DEM filter.

Despite these facts, the next chapter proves that higher-order derivatives exist and contain information that can be used to construct a proper state estimate by showing the DEM filter results.

6-3 Gaussian filter validity

The goal of this section is to draw a conclusion on the validity of the Gaussian filter assumption (i.e. coloured noise is generated by convoluting a white noise sequence with a Gaussian filter).

Figures 6-11, 6-12 and 6-13 show the measurement and process noises, together with a white noise signal generated using the same mean and variance of the original noise, and a coloured noise signal generated by convoluting the white noise signal with a Gaussian filter with the smoothness values found above as kernel widths. The figures indicate that all noises are more comparable to the generated coloured noise than to the white noise signal. The figures

also indicate that z and $w_{2,res}$ are comparable in smoothness to the corresponding generated coloured noise signals. w_1 is only comparable in smoothness in limited time frames. Section 7-2 will confirm these observations by showing that the DEM filter always outperforms Kalman for smoothness values corresponding to z and $w_{2,res}$, while it requires tuning for the smoothness value corresponding to w_1.

The figures also indicate that the smoother the noise, the more stable the smoothness in the generated coloured noise signal over the whole time frame is. However, this effect is not visible in the experiment data.

In conclusion, the Gaussian filter validity depends on the purpose. If we want to have an accurate description of the measurement and process noise during an experiment, the Gaussian filter is too optimistic regarding stability of the smoothness over time and should not be used. On the other hand, if one is interested in estimating the smoothness value, the Gaussian filter approach will result in proper values. However, care has to be taken for relatively high smoothness values, as indicated by the filter results for the smoothness value obtained for w_1.

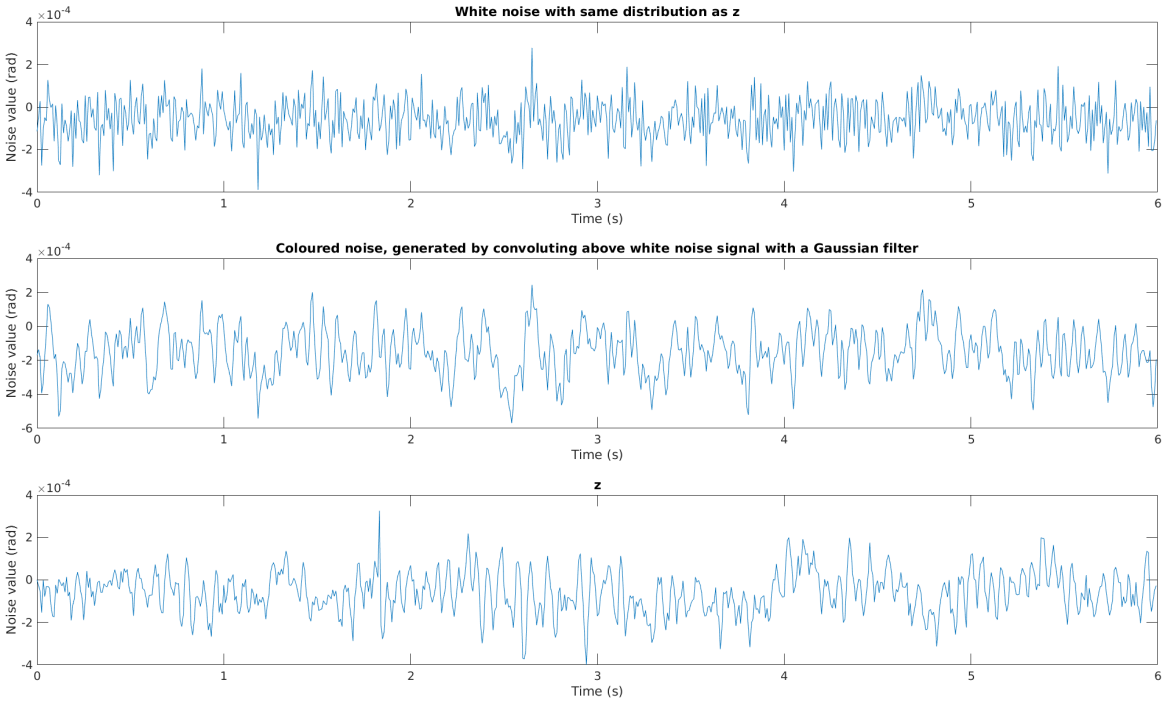


Figure 6-11: From top to bottom: white noise with same distribution as z, coloured noise generated by convoluting the white noise with a Gaussian filter with kernel width equal to s_z, and z itself.

6-4 Another smoothness estimation method

Until now, this chapter has focused on estimating the smoothness values s_z, s_{w_1} and $s_{w_{2,res}}$ using the Gaussian filter assumption. These smoothness values are meant to be an educated guess for the single smoothness value giving the best DEM filter results with respect to using other smoothness values. As described in Chapter 2, the smoothness value is used in the

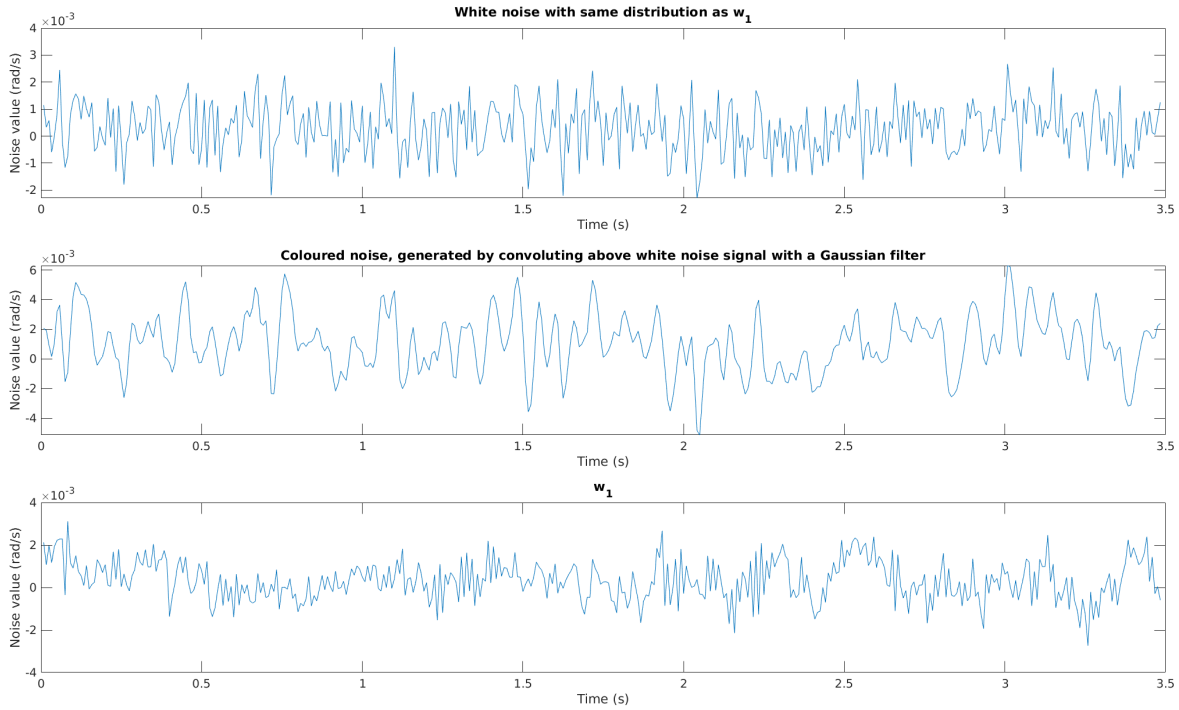


Figure 6-12: From top to bottom: white noise with same distribution as w_1, coloured noise generated by convoluting the white noise with a Gaussian filter with kernel width equal to s_{w_1}, and w_1 itself.

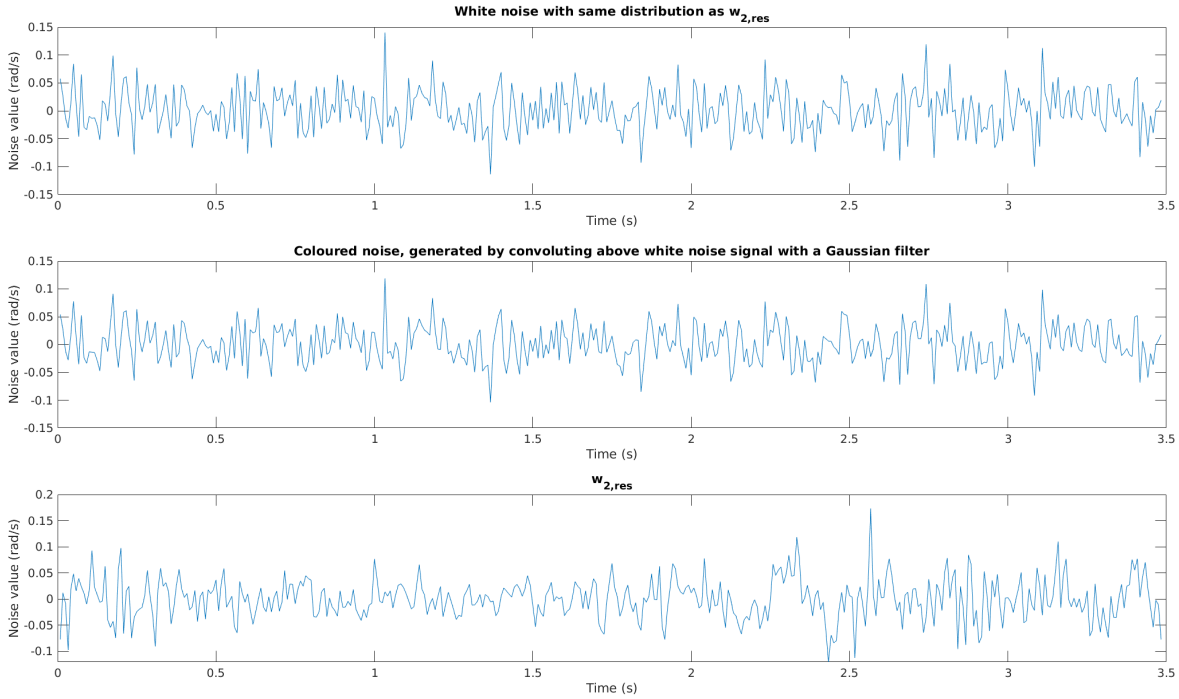


Figure 6-13: From top to bottom: white noise with same distribution as $w_{2,res}$, coloured noise generated by convoluting the white noise with a Gaussian filter with kernel width equal to $s_{w_{2,res}}$, and $w_{2,res}$ itself.

theoretical representation of the covariance matrix of generalized motions. This would mean that the covariance matrix of the generalized system output should have the same form as the theoretical covariance matrix of a generalized signal in case the generalized system output does not contain an error and perfectly represents the roll angle measurement dynamics. So by calculating the covariance matrix of the generalized system output and relating the elements to the elements in theoretical covariance matrix, the smoothness value can be calculated.

The theoretical covariance matrix of $\tilde{\boldsymbol{z}}$ (with $p = 6$) is given by [23]:

$$\tilde{\Sigma}_z = \begin{bmatrix} \sigma_z^2 & 0 & -\frac{\sigma_z^2}{2s^2} & 0 & \frac{3\sigma_z^2}{4s^4} & 0 & -\frac{15\sigma_z^2}{8s^6} \\ 0 & \frac{\sigma_z^2}{2s^2} & 0 & -\frac{3\sigma_z^2}{4s^4} & 0 & \frac{15\sigma_z^2}{8s^6} & 0 \\ -\frac{\sigma_z^2}{2s^2} & 0 & \frac{3\sigma_z^2}{4s^4} & 0 & -\frac{15\sigma_z^2}{8s^6} & 0 & \frac{105\sigma_z^2}{16s^8} \\ 0 & -\frac{3\sigma_z^2}{4s^4} & 0 & \frac{15\sigma_z^2}{8s^6} & 0 & -\frac{105\sigma_z^2}{16s^8} & 0 \\ \frac{3\sigma_z^2}{4s^4} & 0 & -\frac{15\sigma_z^2}{8s^6} & 0 & \frac{105\sigma_z^2}{16s^8} & 0 & -\frac{945\sigma_z^2}{32s^{10}} \\ 0 & \frac{15\sigma_z^2}{8s^6} & 0 & -\frac{105\sigma_z^2}{16s^8} & 0 & \frac{945\sigma_z^2}{32s^{10}} & 0 \\ -\frac{15\sigma_z^2}{8s^6} & 0 & \frac{105\sigma_z^2}{16s^8} & 0 & -\frac{945\sigma_z^2}{32s^{10}} & 0 & \frac{10{,}395\sigma_z^2}{64s^{12}} \end{bmatrix} \tag{6-17}$$

This matrix contains six different expressions involving s: $\frac{\sigma_z^2}{2s^2}$, $\frac{3\sigma_z^2}{4s^4}$, $\frac{15\sigma_z^2}{8s^6}$, $\frac{105\sigma_z^2}{16s^8}$, $\frac{945\sigma_z^2}{32s^{10}}$ and $\frac{10{,}395\sigma_z^2}{64s^{12}}$. By equating each element involving one of these expressions to the corresponding element in the covariance matrix of the generalized output, different values for the smoothness can be calculated. The result is shown in Table 6-1.

Table 6-1: Smoothness values calculated using each element in $\tilde{\Sigma}_z$ that should theoretically be calculated with an expression involving s. The first value corresponds to calculation using elements on the diagonal, the next terms correspond to a calculation using elements further away from the diagonal.

$\frac{\sigma_z^2}{2s^2}$	$\frac{3\sigma_z^2}{4s^4}$	$\frac{15\sigma_z^2}{8s^6}$	$\frac{105\sigma_z^2}{16s^8}$	$\frac{945\sigma_z^2}{32s^{10}}$	$\frac{10{,}395\sigma_z^2}{64s^{12}}$
$1.22 \cdot 10^{-4}$	$1.94 \cdot 10^{-3}$	$4.26 \cdot 10^{-3}$	$3.74 \cdot 10^{-3}$	$6.59 \cdot 10^{-3}$	$5.55 \cdot 10^{-3}$
$1.15 \cdot 10^{-4}$	$2.86 \cdot 10^{-3}$	$2.94 \cdot 10^{-3}$	$5.53 \cdot 10^{-3}$	$4.70 \cdot 10^{-3}$	-
$1.15 \cdot 10^{-4}$	$2.86 \cdot 10^{-3}$	$2.94 \cdot 10^{-3}$	$5.53 \cdot 10^{-3}$	$4.70 \cdot 10^{-3}$	-
-	$2.09 \cdot 10^{-3}$	$4.61 \cdot 10^{-3}$	$4.03 \cdot 10^{-3}$	-	-
-	$2.09 \cdot 10^{-3}$	$4.61 \cdot 10^{-3}$	$4.03 \cdot 10^{-3}$	-	-
-	-	$3.43 \cdot 10^{-3}$	-	-	-
-	-	$3.43 \cdot 10^{-3}$	-	-	-

In theory, all calculated smoothness values should be same. In practice, they differ from each other, as becomes clear in Table 6-1. This is caused by the fact that the generalized output covariance matrix does not match the theoretical covariance matrix exactly. Despite the different smoothness values, the smoothness values calculated using the higher-order elements are in the same region as calculated earlier in this chapter. As the next section will show, the smoothness values are also in the range of the 'optimal' smoothness for which DEM outperforms Kalman. Given this promising result, calculating the smoothness values this way is certainly worth further investigation and will therefore be considered future work.

Chapter 7

Filter results

This chapter shows the DEM filter performance results on recorded experimental quadrotor flight data. First of all, Section 7-1 will show the usefulness of generalized coordinates (for states, output and inputs) for state estimation with DEM. Secondly, Section 7-2 will prove that DEM outperforms the Kalman filter, as well as show the values for tuning parameters p, d and s for which the outperformance of Kalman holds. Appendix E provides the complete filter results dataset, based on which the figures in this chapter are constructed.

7-1 The impact of generalized motions on state estimation

Chapter 2 has shown that the main power of the DEM filter is the fact that it leverages the information contained in (smooth) noises for state estimation. This information comes in the form of different orders of derivatives of the states, outputs and inputs, weighted by precision matrices. This section will show the importance of the derivatives for the DEM state estimation performance on recorded quadrotor flight data.

Figure 7-1 shows the DEM state estimation results with and without generalized states. When using generalized states, there is a very slight increase in error for the observed state, but this is orders of magnitude lower than the error decrease on the hidden state. Clearly, using generalized states has a positive influence on the DEM filter state estimation performance.

Figure 7-2 shows the DEM state estimation results with and without generalized output. Similar to generalized states, generalized output introduces a very slight error increase for the observed state and orders of magnitude higher error decrease for the hidden state. Therefore, including generalized output also has a positive influence on the DEM filter state estimation performance.

Figure 7-3 shows the DEM state estimation results with and without generalized inputs. In contrast to generalized states and generalized output, generalized inputs do not have a significant impact on the observed as well as hidden state estimation error.

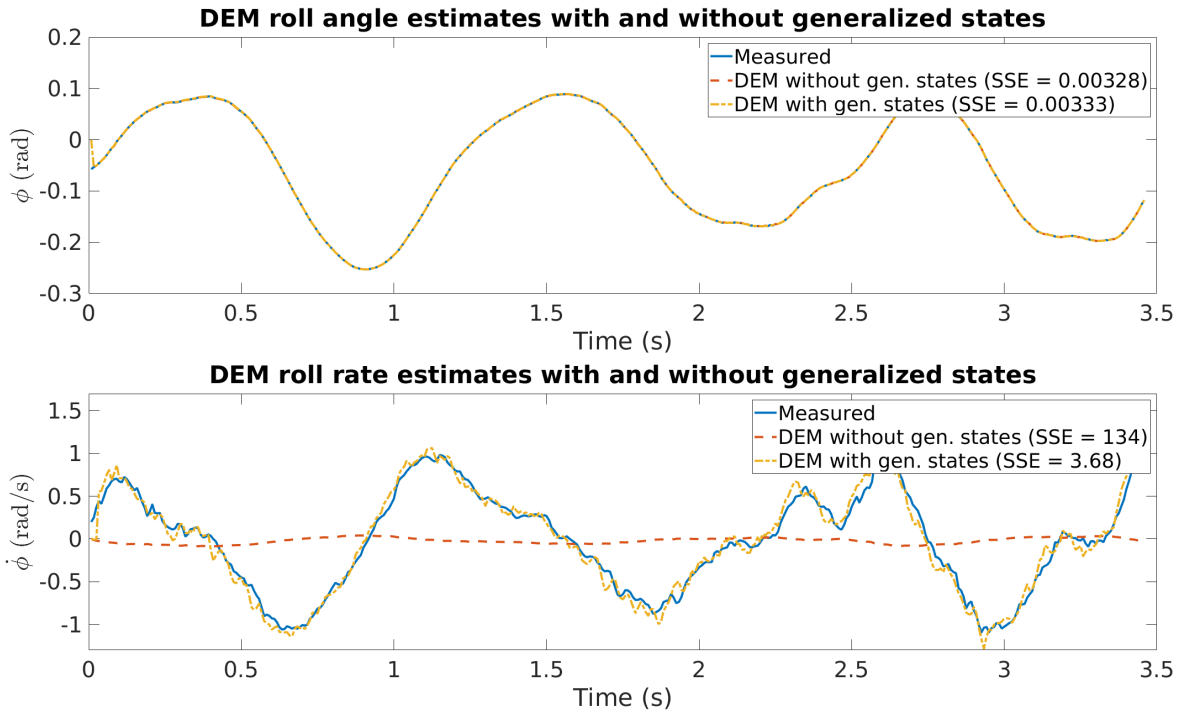


Figure 7-1: Impact of using generalized states on DEM state estimation. The following parameter values are used: $p \in \{0, 2\}$, $d = 2$ and $s = 0.005$ s.

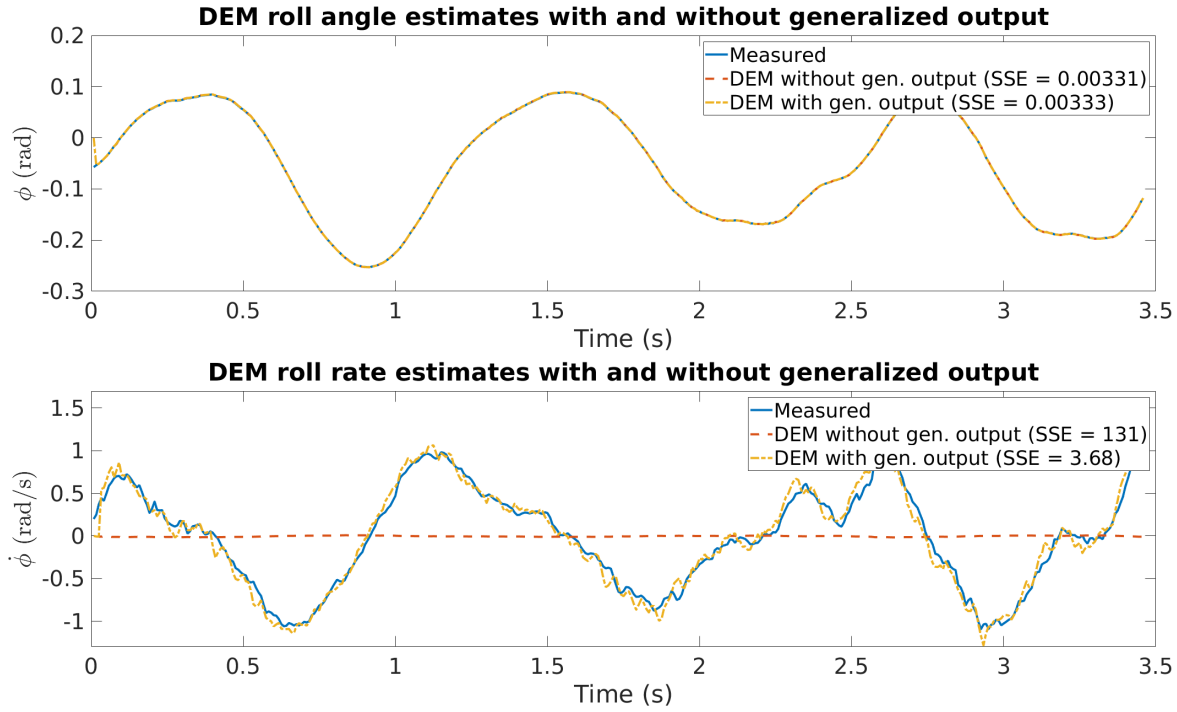


Figure 7-2: Impact of using generalized output on DEM state estimation. The following parameter values are used: $p = 2$, $d = 2$ and $s = 0.005$ s. $p = 2$ is used to have access to generalized states, but all output derivatives are set to zero.

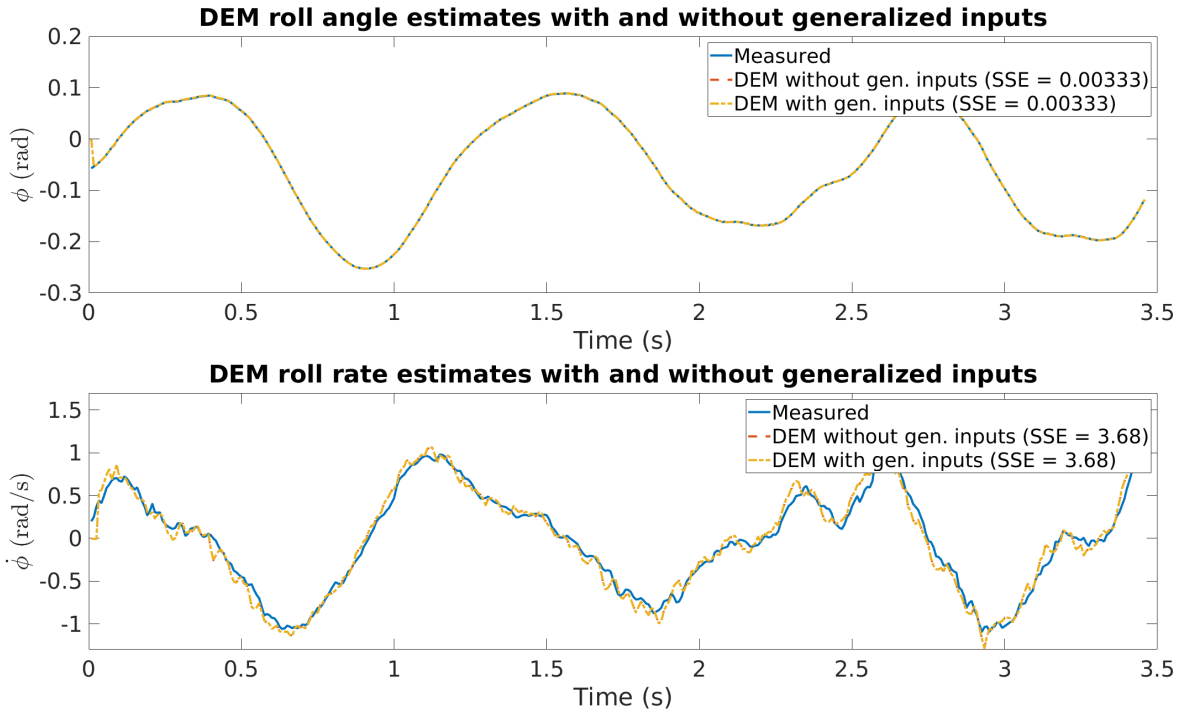


Figure 7-3: Impact of using generalized inputs on DEM state estimation. Take care that both DEM estimates are overlapping. The following parameter values are used: $p = 2$, $d \in \{0, 2\}$ and $s = 0.005$ s.

7-2 Benchmarking DEM

Figure 7-4 shows one of the main results in this thesis: the DEM filter is able to outperform the Kalman filter in estimating the hidden state as the SSE of the DEM estimate is lower than the SSE of the Kalman estimate.

This result is valid for specific values of tuning parameters p, d and s. However, the result can be extended to bigger tuning spaces of p, d and s values. Figure 7-5 shows the varying SSE values of the DEM filter for all combinations of p and d in the range 1 to 6 per smoothness value as well as the constant SSE for the Kalman filter. Since the error bars indicate minimum and maximum performance of all combinations of p and d values, the figure clearly shows that there is a big design space of parameters p, d and s in which DEM will always outperform Kalman. This holds for smoothness values $s = 9 \cdot 10^{-4}$ s up to and including $s = 8 \cdot 10^{-3}$ s.

If the noise analysis, similar to the work in the previous chapter, returns smoothness values in the range of $9 \cdot 10^{-4}$ s, the left-hand side of the figure tells us that, on an average for all p and d combinations, a small adjustment in s will significantly change the filter results. It is still possible to outperform Kalman, but it requires more tuning of p and d. Therefore, s becomes a critical tuning parameter in this case. This is expected, since a low smoothness value corresponds to less smooth noise, meaning that derivatives provide less information and the advantage of the DEM filter with respect to Kalman disappears. The right-hand side of the figure indicates that for bigger s values, it is still possible to outperform Kalman, but it differs per combination of p and d. In this case p and d become the critical tuning parameters.

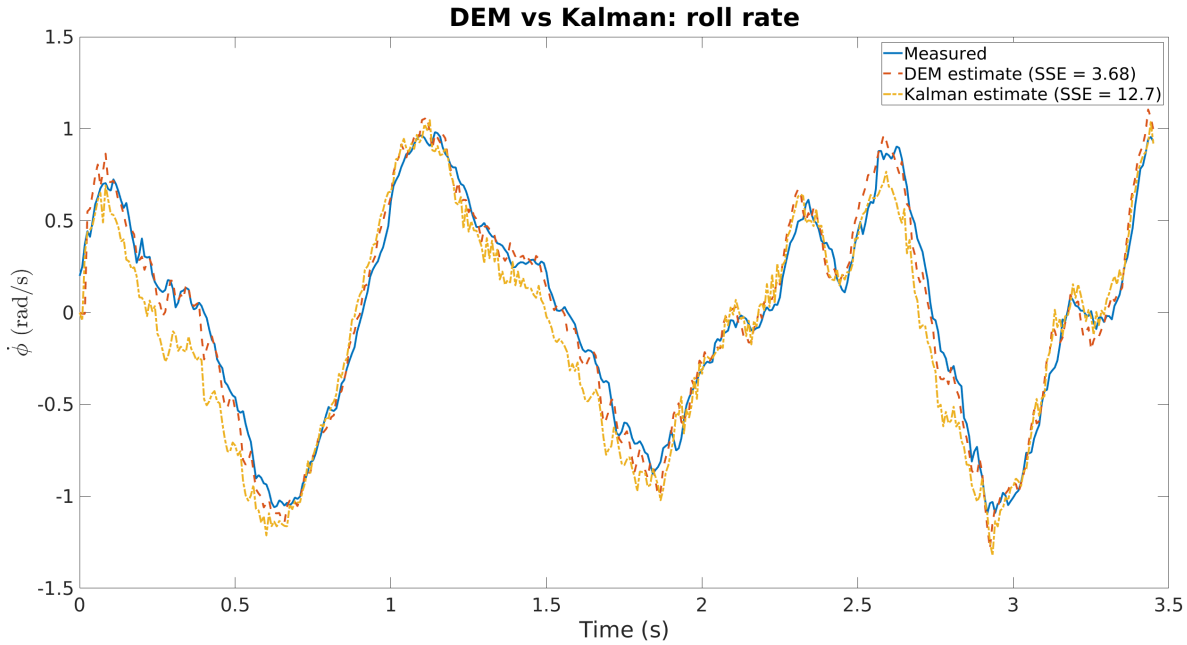


Figure 7-4: Comparison of measured roll rate and DEM and Kalman roll rate estimates for $p = 2$, $d = 2$ and $s = 0.005$ s.

The figure also displays the estimated smoothness values from the previous section. These smoothness values are located in the tuning subspace that results in outperformance of the Kalman filter in almost all cases. Only a few combinations of p and d for $s = 9.0 \cdot 10^{-3}$ s result in a higher SSE value than the SSE of Kalman. This result indicates that the Gaussian filter assumption provides us with proper smoothness values that, regardless of the p and d values, will almost always result in outperformance of the Kalman filter.

As is stated in [4], embedding orders of $p > 6$ and $d > 2$ will not contribute much to the filter results. The dataset provided in Appendix E-1 confirms this statement by showing that for embedding orders $p = 7$ and $d \in \{3, 4, 5, 6, 7\}$ and non-critical smoothness values the results are comparable to the results using lower embedding orders. However, care must be taken when using $p = 7$ and/or $d = 7$ for smoothness values of $9 \cdot 10^{-3}$ and higher, since the SSE starts increasing relatively fast at that point. Therefore, especially when considering smooth noise signals, taking a maximum embedding order of 6 for both p and d is considered a proper practical rule of thumb.

As displayed in Section 7-1, the SSE of the DEM filter roll angle estimate with generalized states and output is slightly higher than the SSE without generalized states and output. This phenomenon is caused by the usage of non-ideal smoothness and derivative information. The higher the smoothness, the more the derivatives are weighted, the more the errors in non-ideal derivatives are weighted, the bigger the error becomes. This error has the same order of magnitude as the Kalman error up to and including $s = 9 \cdot 10^{-3}$ s. Thereafter, the error starts increasing up to the order of 10^{-2} for $s = 1.5 \cdot 10^{-2}$ s and most embedding orders. It can be concluded that the DEM filter allows to make a trade-off in state estimation for the error of observed and hidden states. However, this trade-off is very limited, for the observed state error is orders of magnitude lower than the hidden state error. However, to ensure the

total error is taken into account, above analysis is based on the sum of observed and hidden state SSE.

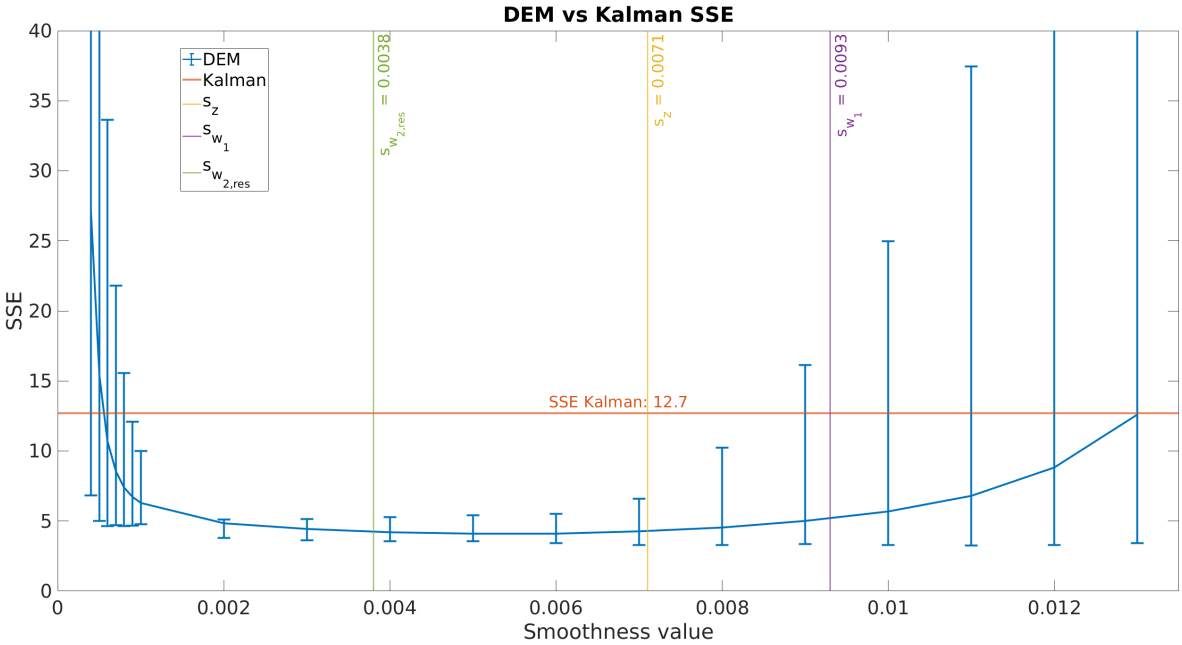


Figure 7-5: Comparison of DEM and Kalman filter SSE values for $p, d \in \{1, 2, 3, 4, 5, 6\}$ and s ranging from $4 \cdot 10^{-4}$ until 10^{-3} in steps of 10^{-4} and from 10^{-3} until $13 \cdot 10^{-3}$ in steps of 10^{-3}. For each smoothness value, the mean SSE of the results of all p and d combinations is given. The error bars indicate the minimum and maximum SSE of all p and d combinations per smoothness value.

Chapter 8

Conclusions

This conclusive chapter will provide a summary of the research findings in this thesis in Section 8-1. Based on this information, Section 8-2 gives an answer to the main research question. Furthermore, to provide an overview of the work done, Section 8-3 will list all thesis contributions. Finally, Section 8-4 provides a discussion of the quadrotor modelling approach, experimental setup and filter results and addresses the main limitations of this work and corresponding future work.

8-1 Summary

This thesis is part of the research aiming to provide an elaborate discussion on the performance of DEM. DEM is a neuroscientific parameter estimation algorithm that should be capable of solving the filtering and system identification problems in the control engineering domain. In this thesis, the performance of the filtering, or state estimation, part of DEM is evaluated using experimental quadrotor flight data.

In order to be able to assess the performance of the DEM filter, the filter needs a dynamic system model. A quadrotor model that involves the quadrotor roll angle and rate as states and the roll angle as output is chosen. The system inputs are defined as motor PWM values. The parameters in the model, including mass, arm length, mass moment of inertia and thrust coefficients, are identified using several identification experiments. The data used to run the filter is obtained by hovering the quadrotor at a few meters away from a wind source. The quadrotor blade rotations and the wind source will produce air flows in the lab environment causing the drone to move smoothly.

The advantage of state estimation using DEM is the fact that it can derive extra information from the derivatives of the system inputs and outputs to better infer the states, given that the process and measurement noises are coloured. The smooth motions of the quadrotor cause the system inputs and output to be smooth signals, such that their derivatives exist. Furthermore, since a quadrotor is a highly nonlinear system and it is modelled using an LTI model, unmodelled system dynamics, among other effects, are present and appear in the

process noise. By definition, the unmodelled system dynamics are not changing infinitely fast, thereby introducing coloured noise.

One of the main research contributions of this thesis is the analysis of the measurement and process noises. In general, coloured noise is characterized by a covariance matrix and a smoothness value. These values are derived for both process and measurement noise. Furthermore, it has been shown that the noises are Gaussian distributed, thereby validating the Laplace approximation used to derive the DEM filter. The Gaussian filter assumption, commonly made to construct coloured noise in simulation, is validated as an assumption used to derive the noise smoothness values. However, it is shown that the coloured noise created by a Gaussian filter looks different than the derived noise signals using the experimental data. Finally, another smoothness estimation method is proposed that seems to provide reasonable smoothness estimates. Further research is needed to investigate the strengths and pitfalls of this method and to ensure a consistent smoothness estimate.

Another main research contribution is the DEM filter performance evaluation. It is shown that the DEM filter heavily relies on the generalized motions of states, outputs and inputs. The DEM filter is compared with the conventional Kalman filter to benchmark its performance. It can be concluded that DEM is able to outperform Kalman. Furthermore, it is shown that this result holds for a relatively big subspace of the tuning parameters p, d and s.

8-2 Answering the main research question

Based on the information above, we are able to answer the main research question in this thesis:

How does the DEM filter perform on experimental data of a quadrotor flight?

To give a direct and quantitative answer: DEM is able to outperform the Kalman filter with an SSE value of 3.68 against 12.7 by setting $p = 2$, $d = 2$ and $s = 0.005$ s. These parameter values are not the only ones for which DEM outperforms Kalman. The result can be extended to a relatively big subspace of parameters values ($1 \leq p \leq 6$, $1 \leq d \leq 6$ and $9 \cdot 10^{-4} \leq s \leq 8 \cdot 10^{-3}$ s). Other combinations of p, d and s will also work, but this requires parameter tuning. Using this information we can also conclude that the DEM filter fits the purpose of state estimation on quadrotor UAVs.

8-3 Thesis contributions

This thesis contributed by:

1. Giving a mathematical and intuitive explanation of the principles underlying DEM and how they are used to derive the DEM state update equation.
2. Selecting a quadrotor dynamic model used in the DEM and Kalman filter to perform state estimation, based on the information available in the hardware/software platform.
3. Designing the experimental setup by selecting a hardware/software platform suitable to record data, based on which the performance of DEM can be evaluated.

4. Performing system identification experiments to determine the model parameters values specific to the Parrot AR.Drone 2.0 quadrotor (including mass, arm length, mass moment of inertia, thrust and torque coefficients).
5. Analyzing the noises:
   (a) Deriving the covariance matrices.
   (b) Deriving the smoothness values.
   (c) Validating the Laplace approximation used to derive the DEM filter equations.
   (d) Discussing the Gaussian filter assumption used to generate coloured noise using white noise.
   (e) Introducing a new method to estimate the noise smoothness.
6. Analyzing the filter results:
   (a) Showing the importance of including generalized motions for states, outputs and inputs.
   (b) Comparing the DEM filter with the conventional Kalman filter.
   (c) Showing in which tuning parameter subspace the DEM filter outperforms the Kalman filter.

8-4 Future work and recommendations

The research conducted in this thesis has several limitations. These limitations can be divided into four categories: one concerning the statistical significance of the presented results, a second one concerning the quadrotor modelling approach and lab setup, a third one concerning DEM-specific features and a fourth one concerning the filter benchmarking. These limitations are discussed below and corresponding future research directions are pointed out.

Statistical significance
The results presented in this thesis are based on a single quadrotor flight experiment. This single experiment immediately indicates the potential of DEM. However, it is strongly recommended to make the result more significant from a statistical point of view. This means conducting the same experiment several times and showing how DEM performs with respect to Kalman for each experiment.

Quadrotor modelling approach and lab setup
Besides increasing the amount of experiments (quantitative improvement), the data collected during the experiment could also be improved (qualitative improvement). As explained in Section 6-2-2, the smoothness of w_2 arises in a different time frame than useful for the DEM filter. It is recommended to conduct an experiment that results in process noise w_2 having smoothness in a time frame that can be leveraged by the DEM filter. Before designing such an experiment, it is recommended to first reveal the effect of each of the four phenomena (as described in Section 6-2-2) by conducting experiments with reduced air flow circulation and/or with linear quadrotor movements without external wind source, for example.

Another qualitative improvement related to the quadrotor modelling is including more (hidden) states to be estimated. For proper quadrotor control, only estimating the roll rate is not enough. Future work might consider the filter results for other hidden states as well. By including more hidden states, care has to be taken regarding the observability of the system.

Improving DEM filter
Other possible qualitative improvements relate to specific features of DEM.

First of all, the process noises are derived using the assumption that we can perfectly measure the states (roll angle and rate). Although the sensor noise is relatively low compared to the measured values, it is unclear what impact this assumption has on the final results. The goal of this assumption was to be able to calculate the process noise in order to derive the precision matrix and smoothness values. To avoid making this assumption, the process noise precision matrix can be tuned manually, which could potentially lead to even better DEM filter results. Therefore, given the current results, future work could focus on the range of precision values for which the results still hold.

Secondly, as pointed out in Section 6-4, besides the Gaussian filter approach, the smoothness values of the noises can possibly be determined by evaluating specific elements in the covariance matrix of the generalized output. It is shown that for higher embedding orders, the calculated smoothness values using this approach come close to the region of smoothness values for which the DEM filter outperforms Kalman. This thesis has only indicated the potential and leaves a more rigid understanding as future work.

Thirdly, the current implementation of DEM uses one smoothness value for all noises. Although the smoothness values are shown to have the same order of magnitude, future experiment (as indicated above) might give higher process noise smoothness values. Therefore, it would be interesting to see the impact of using different smoothness values for different noises on the filter results.

Fourthly, the results presented in this thesis are based on recorded data and post-processing. In general, however, filtering is used to construct a proper state estimate to improve control, which should be performed real-time. Therefore, another possible research direction is to run the filter online. The problem of generating the generalized motions can be solved by subsampling the sensor signals for each filter update, such that intermediate sensor values can be used to construct the derivatives.

Benchmarking DEM filter
In this thesis the conventional Kalman filter has been used to evaluate the performance of the DEM filter. However, the Kalman filter assumes white noise, while the DEM filter is built to derive extra information from coloured noise. In this respect, the comparison is a bit unfair. Therefore, an important part of future work should consider a 'fair' comparison between DEM and another type of filter able to handle coloured noises.

Appendix A

Quadrotor model

Sections A-1 and A-2 below give the derivation of the rotation matrix used to translate coordinates from $\mathcal{B}$ into $\mathcal{I}$ and vice versa, as well as the derivation of the rotation matrix used to translate body frame angular velocities into Euler angle rates and vice versa. Section A-3 describes the quadrotor model linearization.

A-1 Derivation of rotation matrix for translational dynamics

The rotation matrix for translational dynamics is constructed using ZYX Euler angles. These Euler angles indicate that one obtains the body frame $\mathcal{B}$ coordinate axes by first rotating the inertial frame $\mathcal{I}$ axes with yaw angle ψ around the z-axis to obtain the so-called vehicle-1 frame. The vehicle-1 frame is rotated with pitch angle θ around the y-axis of this frame to get the vehicle-2 frame. Finally, the vehicle-2 frame is rotated with roll angle ϕ around the x-axis of this frame to arrive at frame $\mathcal{B}$ [31].

The rotations around the z-, y- and x-axis, respectively, are given by the following 3D rotation matrices:

$$ {}^{\mathcal{I}}R_{Z,\mathcal{B}} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A-1} $$

$$ {}^{\mathcal{I}}R_{Y,\mathcal{B}} = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \tag{A-2} $$

$$ {}^{\mathcal{I}}R_{X,\mathcal{B}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \tag{A-3} $$

where $c\phi = cos(\phi)$ and $s\phi = sin(\phi)$.

The complete rotation matrix is constructed as follows:

$$\begin{aligned} {}^{\mathcal{I}}R_{\mathcal{B}} &= {}^{\mathcal{I}}R_{ZYX,\mathcal{B}} = {}^{\mathcal{I}}R_{Z,\mathcal{B}}\,{}^{\mathcal{I}}R_{Y,\mathcal{B}}\,{}^{\mathcal{I}}R_{X,\mathcal{B}} \\ &= \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \\ &= \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \end{aligned} \tag{A-4}$$

This matrix allows to translate coordinates expressed in $\mathcal{B}$ into coordinates expressed in $\mathcal{I}$. Vice versa, ${}^{\mathcal{I}}R_{\mathcal{B}}^{T} = {}^{\mathcal{B}}R_{\mathcal{I}}$ can be used to translate coordinates expressed in $\mathcal{I}$ into coordinates expressed in $\mathcal{B}$. Its time derivative is given by the following equation [72]:

$$ {}^{\mathcal{I}}\dot{R}_{\mathcal{B}} = {}^{\mathcal{I}}R_{\mathcal{B}}\boldsymbol{\Omega}_{\times} \tag{A-5}$$

A-2 Derivation of rotation matrix for rotational dynamics

The rotation matrix for rotational dynamics can be constructed in a similar fashion. Since the roll angle ϕ is used to convert the vehicle-2 frame to $\mathcal{B}$, its time derivative is equal to the angular velocity p in $\mathcal{B}$. However, the pitch angle is defined in the vehicle-2 frame, not in $\mathcal{B}$. Therefore, its time derivative has to be translated from the vehicle-2 frame into $\mathcal{B}$. Since ${}^{\mathcal{I}}R_{X,\mathcal{B}}$ can convert coordinates expressed in $\mathcal{B}$ to coordinates expressed in the vehicle-2 frame, its inverse changes coordinates the other way around. Given that ${}^{\mathcal{I}}R_{X,\mathcal{B}}$ is part of the $SO(3)$ group, its inverse is equal to its transpose. ${}^{\mathcal{I}}R_{X,\mathcal{B}}^{T}$ is thus used in the conversion from pitch angle rate $\dot{\theta}$ to angular velocity q. Finally, in a similar way, the angular velocity r is obtained by transforming the yaw angle rate $\dot{\psi}$ to the vehicle-2 frame and $\mathcal{B}$ by pre-multiplying with ${}^{\mathcal{I}}R_{Y,\mathcal{B}}^{T}$ and ${}^{\mathcal{I}}R_{X,\mathcal{B}}^{T}$, respectively. This results in the following conversion from Euler angle rates to body frame angular velocities [31]:

$$\begin{aligned} \begin{bmatrix} p \\ q \\ r \end{bmatrix} &= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + {}^{\mathcal{I}}R_{X,\mathcal{B}}^{T} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + {}^{\mathcal{I}}R_{X,\mathcal{B}}^{T}\,{}^{\mathcal{I}}R_{Y,\mathcal{B}}^{T} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \\ &= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi s\phi & \\ 0 & -s\phi c\phi & \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} \\ &\quad + \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi s\phi & \\ 0 & -s\phi c\phi & \end{bmatrix} \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \end{aligned} \tag{A-6}$$

In general, however, quadrotors often contain an Inertial Measurement Unit (IMU) sensor with a gyroscope measuring p, q and r, not Euler angle rates $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$. Reversing this equation (and thus taking the inverse of the rotation matrix) results in [31]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi sec\theta & c\phi sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{A-7}$$

where $t\theta = tan(\theta)$ and $sec\theta = sec(\theta) = \frac{1}{cos(\theta)}$. This matrix will be referred to as ${}^{\mathcal{I}}R_{r,\mathcal{B}}$. The gimbal lock, as mentioned in Section 3-1 becomes clear from this equation, because the matrix is non-invertible for $\theta = \frac{\pi}{2}$, caused by the $sec\theta$ terms.

If ϕ and θ are small enough, Equation (A-7) can be written as [31]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{A-8}$$

A-3 Model linearization

The goal of linearization is to derive the state equation in the following form:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} \tag{A-9}$$

where $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$.

This linear form is obtained by taking the Taylor expansion of the state equation up to and including the 1st-order derivative term, which can be written as:

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{x}_e, \boldsymbol{u}_e) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)}(\boldsymbol{x} - \boldsymbol{x}_e) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)}(\boldsymbol{u} - \boldsymbol{u}_e) \tag{A-10}$$

where:

$$A = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} = \begin{bmatrix} \frac{\partial f_1}{\partial \boldsymbol{x_1}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \frac{\partial f_1}{\partial \boldsymbol{x_2}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \cdots & \frac{\partial f_1}{\partial \boldsymbol{x_{12}}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} \\ \frac{\partial f_2}{\partial \boldsymbol{x_1}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \frac{\partial f_2}{\partial \boldsymbol{x_2}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \cdots & \frac{\partial f_2}{\partial \boldsymbol{x_{12}}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{12}}{\partial \boldsymbol{x_1}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \frac{\partial f_{12}}{\partial \boldsymbol{x_2}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \cdots & \frac{\partial f_{12}}{\partial \boldsymbol{x_{12}}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} \end{bmatrix} \tag{A-11}$$

$$B = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} = \begin{bmatrix} \frac{\partial f_1}{\partial \boldsymbol{u_1}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \frac{\partial f_1}{\partial \boldsymbol{u_2}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \cdots & \frac{\partial f_1}{\partial \boldsymbol{u_4}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} \\ \frac{\partial f_2}{\partial \boldsymbol{u_1}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \frac{\partial f_2}{\partial \boldsymbol{u_2}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \cdots & \frac{\partial f_2}{\partial \boldsymbol{u_4}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{12}}{\partial \boldsymbol{u_1}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \frac{\partial f_{12}}{\partial \boldsymbol{u_2}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} & \cdots & \frac{\partial f_{12}}{\partial \boldsymbol{u_4}}|_{(\boldsymbol{x}=\boldsymbol{x}_e, \boldsymbol{u}=\boldsymbol{u}_e)} \end{bmatrix} \tag{A-12}$$

In this case the model is linearized around the hovering equilibrium point, meaning that $\boldsymbol{x}_e$ and $\boldsymbol{u}_e$ equal:

$$\begin{aligned}\boldsymbol{x}_e &= \begin{bmatrix} x_e & y_e & z_e & \dot{x}_e & \dot{y}_e & \dot{z}_e & \phi_e & \theta_e & \psi_e & \dot{\phi}_e & \dot{\theta}_e & \dot{\psi}_e \end{bmatrix}^T \\ &= \begin{bmatrix} 0 & 0 & z_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \end{aligned} \tag{A-13}$$

$$\boldsymbol{u}_e = \begin{bmatrix} \text{pwmA}_{eq} \\ \text{pwmA}_{eq} \\ \text{pwmA}_{eq} \\ \text{pwmA}_{eq} \end{bmatrix} \tag{A-14}$$

where z_0 indicates the height of the quadrotor while hovering. It is only listed for the sake of completeness, because it does not appear in the linearized state-space equations. pwmA_{eq} is determined using hovering experiments and equals 169.1 (using the Parrot battery). More information about the hovering experiments can be found in Section 5-4-2 and Appendix B-3-4.

Assuming that the total thrust equals mg when the rotor at PWM value pwmA_{eq}, the A matrix is not influenced by the input and becomes:

$$\begin{aligned} A &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g[-s0s0c0 + c0s0] & g[c0c0c0] & g[c0s0s0 + s0c0] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g[-s0s0s0 - c0c0] & g[c0c0s0] & g[c0s0c0 + s0s0] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g[-s0c0] & g[-c0s0] & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{A-15}$$

The B matrix depends on the inputs and is therefore given by:

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{c_1}{m}\frac{\partial T}{\partial \text{pwmA}_1}\Big|_{\substack{\text{pwmA}_1=\\ \text{pwmA}_{eq}}} & \frac{c_1}{m}\frac{\partial T}{\partial \text{pwmA}_2}\Big|_{\substack{\text{pwmA}_2=\\ \text{pwmA}_{eq}}} & \frac{c_1}{m}\frac{\partial T}{\partial \text{pwmA}_3}\Big|_{\substack{\text{pwmA}_3=\\ \text{pwmA}_{eq}}} & \frac{c_1}{m}\frac{\partial T}{\partial \text{pwmA}_4}\Big|_{\substack{\text{pwmA}_4=\\ \text{pwmA}_{eq}}} \\ \frac{c_2}{m}\frac{\partial T}{\partial \text{pwmA}_1}\Big|_{\substack{\text{pwmA}_1=\\ \text{pwmA}_{eq}}} & \frac{c_2}{m}\frac{\partial T}{\partial \text{pwmA}_2}\Big|_{\substack{\text{pwmA}_2=\\ \text{pwmA}_{eq}}} & \frac{c_2}{m}\frac{\partial T}{\partial \text{pwmA}_3}\Big|_{\substack{\text{pwmA}_3=\\ \text{pwmA}_{eq}}} & \frac{c_2}{m}\frac{\partial T}{\partial \text{pwmA}_4}\Big|_{\substack{\text{pwmA}_4=\\ \text{pwmA}_{eq}}} \\ \frac{c_3}{m}\frac{\partial T}{\partial \text{pwmA}_1}\Big|_{\substack{\text{pwmA}_1=\\ \text{pwmA}_{eq}}} & \frac{c_3}{m}\frac{\partial T}{\partial \text{pwmA}_2}\Big|_{\substack{\text{pwmA}_2=\\ \text{pwmA}_{eq}}} & \frac{c_3}{m}\frac{\partial T}{\partial \text{pwmA}_3}\Big|_{\substack{\text{pwmA}_3=\\ \text{pwmA}_{eq}}} & \frac{c_3}{m}\frac{\partial T}{\partial \text{pwmA}_4}\Big|_{\substack{\text{pwmA}_4=\\ \text{pwmA}_{eq}}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{I_{xx}}\frac{\partial \tau_\phi}{\partial \text{pwmA}_1}\Big|_{\substack{\text{pwmA}_1=\\ \text{pwmA}_{eq}}} & \frac{1}{I_{xx}}\frac{\partial \tau_\phi}{\partial \text{pwmA}_2}\Big|_{\substack{\text{pwmA}_2=\\ \text{pwmA}_{eq}}} & \frac{1}{I_{xx}}\frac{\partial \tau_\phi}{\partial \text{pwmA}_3}\Big|_{\substack{\text{pwmA}_3=\\ \text{pwmA}_{eq}}} & \frac{1}{I_{xx}}\frac{\partial \tau_\phi}{\partial \text{pwmA}_4}\Big|_{\substack{\text{pwmA}_4=\\ \text{pwmA}_{eq}}} \\ \frac{1}{I_{yy}}\frac{\partial \tau_\theta}{\partial \text{pwmA}_1}\Big|_{\substack{\text{pwmA}_1=\\ \text{pwmA}_{eq}}} & \frac{1}{I_{yy}}\frac{\partial \tau_\theta}{\partial \text{pwmA}_2}\Big|_{\substack{\text{pwmA}_2=\\ \text{pwmA}_{eq}}} & \frac{1}{I_{yy}}\frac{\partial \tau_\theta}{\partial \text{pwmA}_3}\Big|_{\substack{\text{pwmA}_3=\\ \text{pwmA}_{eq}}} & \frac{1}{I_{yy}}\frac{\partial \tau_\theta}{\partial \text{pwmA}_4}\Big|_{\substack{\text{pwmA}_4=\\ \text{pwmA}_{eq}}} \\ \frac{1}{I_{zz}}\frac{\partial \tau_\psi}{\partial \text{pwmA}_1}\Big|_{\substack{\text{pwmA}_1=\\ \text{pwmA}_{eq}}} & \frac{1}{I_{zz}}\frac{\partial \tau_\psi}{\partial \text{pwmA}_2}\Big|_{\substack{\text{pwmA}_2=\\ \text{pwmA}_{eq}}} & \frac{1}{I_{zz}}\frac{\partial \tau_\psi}{\partial \text{pwmA}_3}\Big|_{\substack{\text{pwmA}_3=\\ \text{pwmA}_{eq}}} & \frac{1}{I_{zz}}\frac{\partial \tau_\psi}{\partial \text{pwmA}_4}\Big|_{\substack{\text{pwmA}_4=\\ \text{pwmA}_{eq}}} \end{bmatrix} \tag{A-16}$$

In this expression, $c_1 = c0s0c0 + s0s0 = 0$, $c_2 = c0s0s0 - s0c0 = 0$ and $c_3 = c0c0 = 1$. Furthermore, we have to take derivatives of the thrust and torque values with respect to each individual motor PWM value. Using the final expressions in Eq. (3-13) until (3-16), this gives:

$$\frac{\partial T}{\partial \text{pwmA}_i} = \; 2 \cdot \boldsymbol{c}_{PT}(1) \cdot \text{pwmA}_i + \boldsymbol{c}_{PT}(2) \quad \forall i \in \{1,2,3,4\} \tag{A-17}$$

$$\frac{\partial \tau_\phi}{\partial \text{pwmA}_i} = \begin{cases} 2 \cdot c_{P\phi}(1) \cdot \text{pwmA}_i + c_{P\phi}(2) & \forall i \in \{1,4\} \\ -2 \cdot c_{P\phi}(1) \cdot \text{pwmA}_i - c_{P\phi}(2) & \forall i \in \{2,3\} \end{cases} \tag{A-18}$$

$$\frac{\partial \tau_\theta}{\partial \text{pwmA}_i} = \begin{cases} 2 \cdot c_{P\theta}(1) \cdot \text{pwmA}_i + c_{P\theta}(2) & \forall i \in \{3,4\} \\ -2 \cdot c_{P\theta}(1) \cdot \text{pwmA}_i - c_{P\theta}(2) & \forall i \in \{1,2\} \end{cases} \tag{A-19}$$

$$\frac{\partial \tau_\psi}{\partial \text{pwmA}_i} = \begin{cases} 2 \cdot c_{P\psi}(1) \cdot \text{pwmA}_i + c_{P\psi}(2) & \forall i \in \{1,3\} \\ -2 \cdot c_{P\psi}(1) \cdot \text{pwmA}_i - c_{P\psi}(2) & \forall i \in \{2,4\} \end{cases} \tag{A-20}$$

The B matrix now becomes:

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m}c_{Bz} & \frac{1}{m}c_{Bz} & \frac{1}{m}c_{Bz} & \frac{1}{m}c_{Bz} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{I_{xx}}c_{B\phi} & -\frac{1}{I_{xx}}c_{B\phi} & -\frac{1}{I_{xx}}c_{B\phi} & \frac{1}{I_{xx}}c_{B\phi} \\ -\frac{1}{I_{yy}}c_{B\theta} & -\frac{1}{I_{yy}}c_{B\theta} & \frac{1}{I_{yy}}c_{B\theta} & \frac{1}{I_{yy}}c_{B\theta} \\ \frac{1}{I_{zz}}c_{B\psi} & -\frac{1}{I_{zz}}c_{B\psi} & \frac{1}{I_{zz}}c_{B\psi} & -\frac{1}{I_{zz}}c_{B\psi} \end{bmatrix} \tag{A-21}$$

where:

$$c_{Bz} = 2 \cdot \boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1)^2 \cdot \mathrm{pwmA}_{eq} + 2 \cdot \boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) + \boldsymbol{c}_T(2) \cdot \boldsymbol{c}_A(1) \tag{A-22}$$

$$c_{B\phi} = \frac{\sqrt{2}}{2} \cdot l \cdot \Big(2 \cdot \boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1)^2 \cdot \mathrm{pwmA}_{eq} + 2 \cdot \boldsymbol{c}_T(1) \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) + \boldsymbol{c}_T(2) \cdot \boldsymbol{c}_A(1)\Big) \tag{A-23}$$

$$c_{B\theta} = c_{B\phi} \tag{A-24}$$

$$c_{B\psi} = 2 \cdot \boldsymbol{c}_Q(1) \cdot \boldsymbol{c}_A(1)^2 \cdot \mathrm{pwmA}_{eq} + 2 \cdot \boldsymbol{c}_Q(1) \cdot \boldsymbol{c}_A(1) \cdot \boldsymbol{c}_A(2) + \boldsymbol{c}_Q(2) \cdot \boldsymbol{c}_A(1) \tag{A-25}$$

Appendix B

System identification

This appendix provides the additional background information used to obtain the system identification results in Chapter 5.

B-1 Inertia matrix I

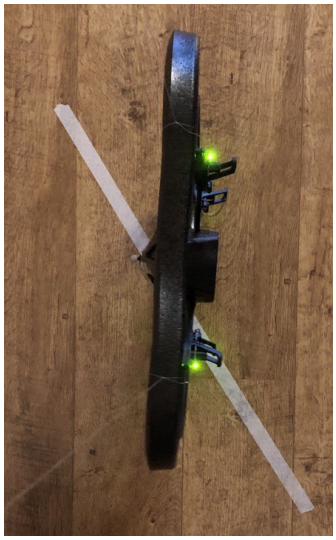
This section provides the design of the bifilar pendulum experiment (as described in Section 5-3) in Section B-1-1 as well as the measurement plan in Section B-1-2 and the experiment datasets in Section B-1-3.

B-1-1 Bifilar pendulum setup

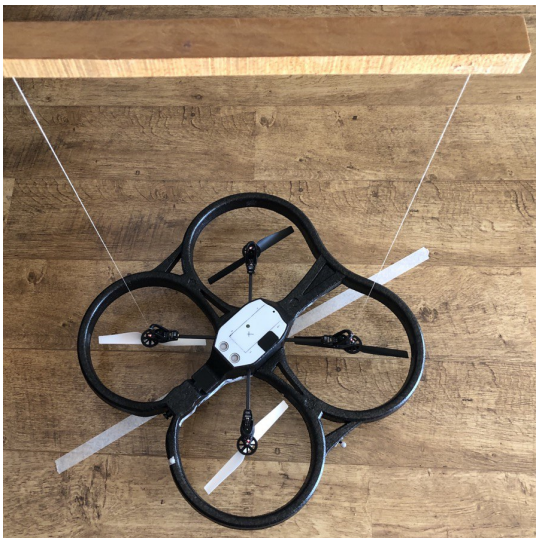
Figure 5-1 shows the theoretical setup that is used for this experiment. This setup is implemented as given in Figure B-1. It is important to note that the quadrotor as well as the frame to which it is attached should be horizontal in order to get realistic gyroscope data around the desired axis. This is checked using a spirit level. Furthermore, it is practical to use relatively long and thin wires. Long wires ensure a stable rotation, because the quadrotor does not rise much when initially turning it, and therefore has less chance to introduce movement in translational x- and y-direction (which would cause less accurate yaw rate values). Thin wires are more flexible and thus have less impact on the rotational dynamics of the quadrotor. Finally, a stroke of tape is stuck on the ground to align the quadrotor with and ensure a certain initial angle. In this case, the angle is chosen to be 32°. As we will see later, the inertia value is mostly determined by the period of oscillation, not by the initial angle.

B-1-2 Measurement plan

The following steps need to be taken in order to be able to perform moment of inertia experiments:



(a) Quadrotor is rotating around roll axis.



(b) Quadrotor attached to frame for rotations around yaw axis.

Figure B-1: Bifilar pendulum experiment setup for rotations around two different axes.

1. Attach wooden frame to table with glue clamp.
2. Put soft surface below AR.Drone 2.0 to prevent damage.
3. Place fully charged battery in AR.Drone 2.0.
4. Connect with laptop to AR.Drone 2.0 Wi-Fi network.
5. Check connection with AR.Drone 2.0 in MATLAB.
6. Put indoor hull around basic AR.Drone 2.0 frame.
7. Attach long, thin wire to one of the rotors (yaw)/rotor hulls (roll/pitch) and to one of the attachment points on the wooden frame.
8. Measure wire length.
9. Ensure the other wire (made of exactly the same material as the first wire) is also attached to both AR.Drone 2.0 and frame with exactly the same length.
10. Slowly remove soft surface below AR.Drone 2.0.
11. Build desired Simulink model (read out IMU data) and connect to target.
12. Start running the Simulink model in External mode.
13. Repeat three times:
    (a) Rotate AR.Drone 2.0 to initial angle α_0 and take care that the COM position only moves in vertical direction.
    (b) Release AR.Drone 2.0 and read simulation time in Simulink.
    (c) Let AR.Drone 2.0 execute damped oscillation for at least 60 s.

14. Stop running the Simulink model in External mode.
15. Process and save logged IMU data in a .mat file.
16. Replace the battery to determine the inertia value for different batteries and start from step 3.
17. Change the configuration to determine the inertia value around another axis and start from step 7.

B-1-3 Datasets

Table B-1 gives the moment of inertia values around the three principle axes of the quadrotor for the two types of batteries. The experiments to determine I_{xx} and I_{yy} are performed with a distance $d_w = 0.252$ m between the wires and a wire length $l_w = 0.668$ m. I_{zz} is determined using $d_w = 0.340$ m and $l_w = 0.655$ m. These values are based on 60 s of gyroscope (roll, pitch or yaw rate) data per experiment.

The values in this table are determined using two different methods. The first method entails calculating the average period of the damped oscillation and using it to calculate the mass moment of inertia around the corresponding principle axis [73], [74]:

$$I = \frac{mgd_w^2 T_o^2}{16h\pi^2} \tag{B-1}$$

where:

- I is the mass moment of inertia around the corresponding principle axis.
- m is the quadrotor mass, including indoor hull and either Parrot or Akku-King battery.
- g is the gravitational acceleration.
- d_w is the distance between the wires.
- T_o is the period of the damped oscillation around the corresponding principle axis.
- l_w is the wire length.

The second method entails simulating the theoretical response of the test object in the bifilar pendulum experiment using the following set of differential equations [63], [64]:

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -\left(\frac{D_1}{I}x_2 + \frac{D_2}{I}\right) - \frac{mgd_w^2}{4Il_w}\frac{sin(x_1)}{\sqrt{1 - \frac{1}{2}\left(\frac{d_w}{l_w}\right)(1 - cos(x_1))}} \end{aligned} \tag{B-2}$$

where:

- x_1 is the angle of rotation of the quadrotor.

Table B-1: Mass moment of inertia values around three principal axes with two batteries in three different experiments.

		Inertia component					
		$\boldsymbol{I_{xx}}$ ($\mathrm{kg\,m^2}$)		$\boldsymbol{I_{yy}}$ ($\mathrm{kg\,m^2}$)		$\boldsymbol{I_{zz}}$ ($\mathrm{kg\,m^2}$)	
		batP	batA	batP	batA	batP	batA
Experiment number	**1**	$3.4 \cdot 10^{-3}$	$3.5 \cdot 10^{-3}$	$4.0 \cdot 10^{-3}$	$4.1 \cdot 10^{-3}$	$6.9 \cdot 10^{-3}$	$6.9 \cdot 10^{-3}$
	2	$3.4 \cdot 10^{-3}$	$3.5 \cdot 10^{-3}$	$4.0 \cdot 10^{-3}$	$4.1 \cdot 10^{-3}$	$6.9 \cdot 10^{-3}$	$6.9 \cdot 10^{-3}$
	3	$3.4 \cdot 10^{-3}$	$3.5 \cdot 10^{-3}$	$4.0 \cdot 10^{-3}$	$4.1 \cdot 10^{-3}$	$6.9 \cdot 10^{-3}$	$6.9 \cdot 10^{-3}$
	Average	$\mathbf{3.4 \cdot 10^{-3}}$	$\mathbf{3.5 \cdot 10^{-3}}$	$\mathbf{4.0 \cdot 10^{-3}}$	$\mathbf{4.1 \cdot 10^{-3}}$	$\mathbf{6.9 \cdot 10^{-3}}$	$\mathbf{6.9 \cdot 10^{-3}}$

- x_2 is the quadrotor rotational velocity.
- D_1 is a damping parameter.
- D_2 is another damping parameter.

The simulated rotational velocity is plotted against the measured rotational velocity over time and the free parameters (D_1, D_2 and I) are tuned manually by taking an initial value for I given by the first method. Both methods result in the same values with one decimal accuracy. One decimal accuracy is chosen, because a test object mass difference of 1 g results in a different value using two decimals, while the weighing scale used to measure the quadrotor has an accuracy of 1 g.

It can thus be concluded that the mass moment of inertia value is approximately the same for both methods. Furthermore, the inertia values reproduce very well over the three different experiments. Moreover, a validation experiment for I_{zz} with a Parrot battery is executed in which l_w differs in value. The result lies within 1% of the indicated averaged value in Table B-1, thereby validating the values in Table B-1.

B-2 Relation between PWM and rotor rotational velocity

This section provides the datasets in Section B-2-1 that are used to visualize the results in Section 5-4-1 as well as additional figures giving extra information about the experiments.

B-2-1 Datasets

Table B-2 provides the battery (type and voltage) and rotational velocity data per PWM setpoint for eight experiments as displayed in Figures 5-3 and 5-4. Each PWM setpoint contains two sets of battery voltage and rotational velocity values, because this data is coming from thrust and torque experiments in which the PWM was varied from 0 to 100 and back, of which only the PWM values 5-80 are taken, as explained in Section 5-4-1.

Table B-2: Battery dependency of rotational velocity for PWM setpoints 5-80 in eight different experiments.

		Experiment number															
		1 (batP)		**2** (batP)		**3** (batP)		**4** (batP)		**5** (batA)		**6** (batA)		**7** (batA)		**8** (batA)	
		v_{bat} (V)	ω_r (RPM)	v_{bat} (V)	ω_r (RPM)	v_{bat} (V)	ω_r (RPM)	v_{bat} (V)	ω_r (RPM)	v_{bat} (V)	ω_r (RPM)	v_{bat} (V)	ω_r (RPM)	v_{bat} (V)	ω_r (RPM)	v_{bat} (V)	ω_r (RPM)
PWM	**5**	12.19	1418	11.40	1424	12.22	1411	11.65	1422	12.48	1428	11.60	1425	12.45	1429	11.93	1419
		11.36	1423	10.75	1426	11.66	1421	11.35	1425	11.63	1426	11.17	1426	11.92	1426	11.57	1423
	10	12.15	1602	11.36	1608	12.20	1596	11.64	1607	12.47	1611	11.59	1608	12.44	1612	11.92	1603
		11.31	1604	10.75	1608	11.65	1603	11.33	1607	11.62	1608	11.16	1608	11.92	1608	11.56	1605
	15	12.09	1779	11.32	1785	12.18	1774	11.62	1783	12.44	1787	11.56	1785	12.43	1788	11.91	1780
		11.26	1779	10.74	1784	11.63	1778	11.31	1783	11.61	1783	11.15	1783	11.91	1783	11.56	1780
	20	12.01	1962	12.27	1968	12.15	1957	11.60	1966	12.41	1970	11.54	1967	12.41	1970	11.89	1963
		11.20	1961	10.74	1966	11.62	1960	11.29	1964	11.59	1965	11.14	1965	11.90	1965	11.55	1962
	25	11.94	2137	11.22	2143	12.12	2133	11.57	2141	12.38	2145	11.51	2143	12.40	2145	11.87	2138
		11.11	2136	10.72	2141	11.60	2134	11.27	2139	11.57	2140	11.12	2140	11.89	2140	11.54	2136
	30	11.84	2320	11.15	2325	12.09	2315	11.54	2323	12.34	2327	11.49	2325	12.37	2327	11.85	2321
		11.18	2317	10.70	2323	11.57	2315	11.24	2321	11.56	2322	11.11	2322	11.88	2322	11.53	2317
	35	11.73	2494	11.08	2500	12.05	2490	11.51	2498	12.30	2501	11.46	2500	12.35	2502	11.83	2496
		11.11	2491	10.68	2497	11.55	2490	11.21	2495	11.54	2495	11.09	2497	11.86	2496	11.52	2491
	40	11.65	2676	10.99	2683	12.00	2672	11.48	2680	12.25	2683	11.42	2682	12.33	2685	11.81	2678
		11.04	2674	10.64	2679	11.52	2671	11.19	2677	11.52	2678	11.06	2678	11.86	2678	11.51	2673
	45	11.53	2850	10.91	2856	11.95	2845	11.43	2854	12.20	2858	11.39	2856	12.30	2858	11.78	2851
		11.00	2847	10.61	2853	11.49	2845	11.15	2852	11.50	2853	11.05	2853	11.84	2853	11.49	2847
	50	11.43	3032	10.79	3038	11.89	3026	11.39	3034	12.15	3038	11.34	3038	12.27	3038	11.75	3031
		10.93	3028	10.55	3035	11.45	3026	11.11	3032	11.47	3035	11.02	3033	11.83	3033	11.48	3028
	55	11.35	3212	10.73	3219	11.83	3207	11.34	3215	12.09	3219	11.29	3218	12.24	3219	11.72	3212
		10.90	3208	10.50	3214	11.41	3205	11.07	3212	11.44	3215	11.01	3216	11.82	3214	11.47	3209
	60	11.26	3386	10.61	3391	11.75	3382	11.28	3390	12.01	3392	11.24	3392	12.19	3393	11.69	3386
		10.80	3382	10.43	3388	11.38	3379	11.03	3387	11.42	3387	10.97	3388	11.80	3388	11.45	3381
	65	11.11	3565	10.55	3572	11.67	3561	11.22	3570	11.94	3574	11.19	3573	12.15	3574	11.65	3566
		10.76	3562	10.37	3569	11.33	3560	10.98	3568	11.39	3570	10.95	3570	11.79	3569	11.43	3563
	70	10.99	3738	10.47	3745	11.54	3734	11.16	3742	11.88	3747	11.14	3746	12.11	3749	11.61	3740
		10.69	3735	10.30	3742	11.27	3732	10.93	3740	11.36	3743	10.93	3741	11.79	3741	11.42	3735
	75	10.88	3918	10.37	3925	11.41	3914	11.08	3923	11.80	3926	11.07	3926	12.06	3928	11.57	3919
		10.59	3915	10.20	3924	11.25	3913	10.86	3920	11.34	3922	10.89	3924	11.78	3922	11.40	3915
	80	10.76	4090	10.30	4098	11.30	4088	11.01	4096	11.70	4100	10.99	4099	12.00	4100	11.52	4092
		10.53	4088	10.11	4096	11.19	4085	10.81	4094	11.31	4097	10.84	4097	11.77	4096	11.39	4088

The data in the Table B-2, together with the data in Table B-3 is used to construct the PWM-rotational velocity relationship indicated in Figure 5-6. The data in Table B-3 is generated by only controlling rotor 4 with high PWM setpoints, such that the battery voltage is high enough to let the rotor reach the high rotational velocities.

B-2-2 Additional figures

Figures B-2 and B-3 show the battery-dependency of the rotational velocity per PWM setpoint in the range of 5-80. In general, the Parrot batteries operate at a lower voltage than the Akku-King batteries. However, since the ESC of each motor is actively controlling the battery voltage, the difference in battery voltage operating region does not introduce a significant

Table B-3: Data of rotational velocities for high PWM setpoints in two different experiments.

		Experiment number	
		1 (batA)	**2** (batA)
		$\boldsymbol{\omega_4}$ (RPM)	$\boldsymbol{\omega_4}$ (RPM)
PWM	**100**	4813	4798
	95	4634	4620
	90	4454	4444
	85	4281	4272
	80	4100	4091

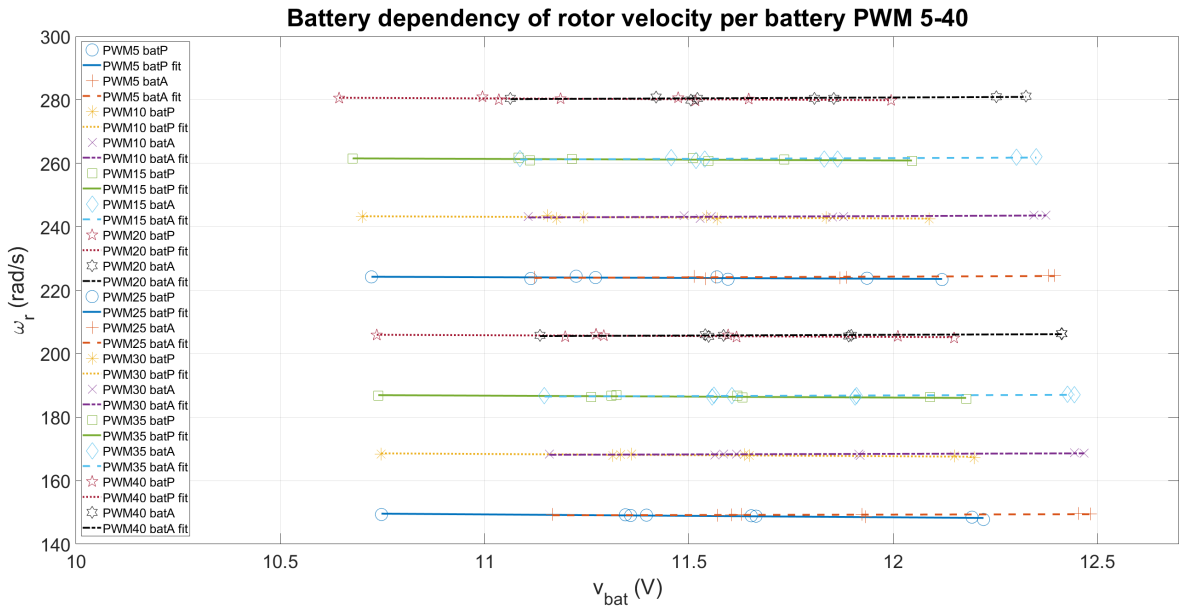


Figure B-2: Battery dependency of rotor 4 velocity for PWM values ranging from 5 to 40.

battery-dependency of the rotational velocity. The error values of the fits, as explained in Section 5-4-1 and shown in Figure B-4, remain below 1% and are therefore ignored.

B-3 Thrust coefficient c_T

This section provides the design of the thrust setup in Section B-3-1, together with the measurement plan in Section B-3-2, used to obtain the datasets in Section B-3-3 that are used to visualize the thrust coefficient results in Section 5-4-2. Finally, Section B-3-4 provides the validation data for the thrust coefficient.

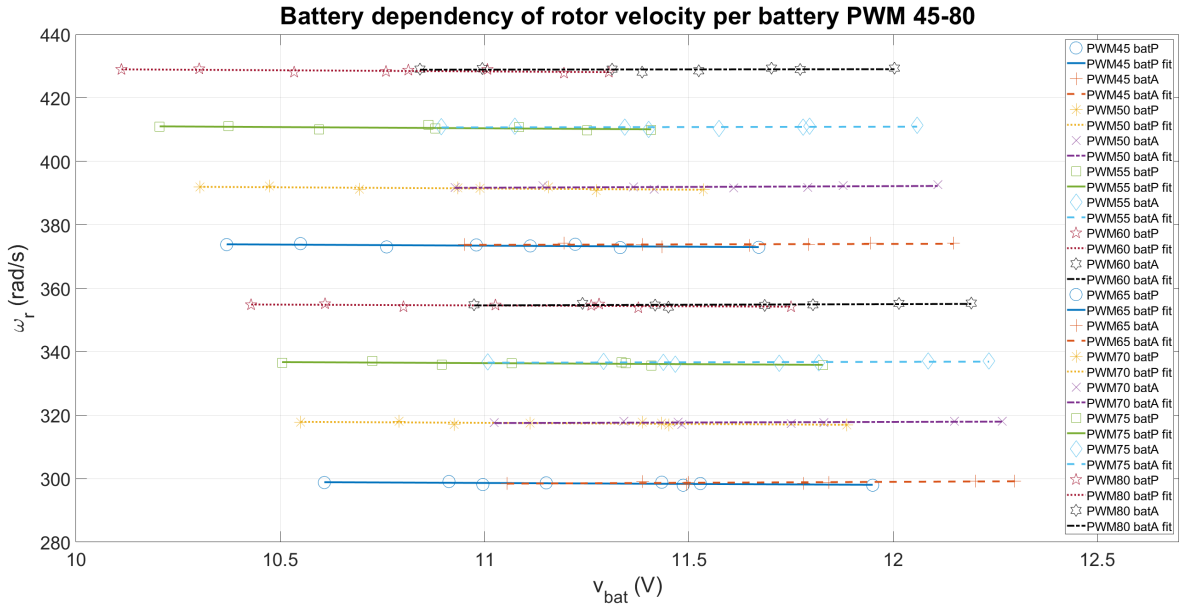


Figure B-3: Battery dependency of rotor 4 velocity for PWM values ranging from 45 to 80.

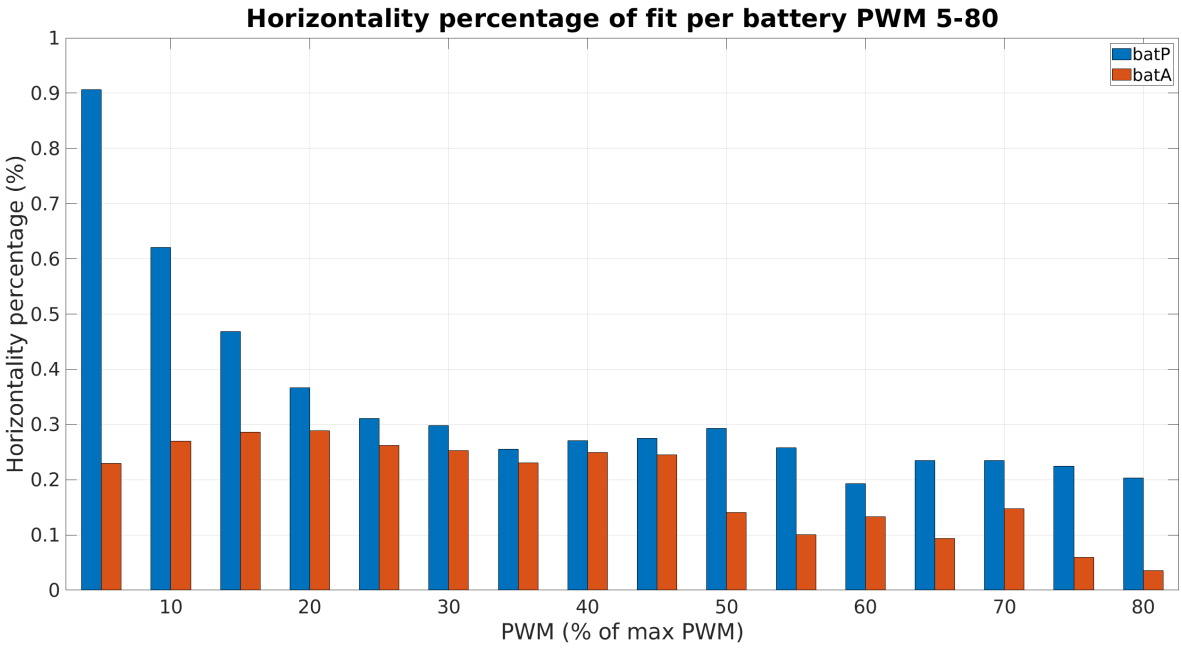


Figure B-4: Horizontality indication of fits per battery type.

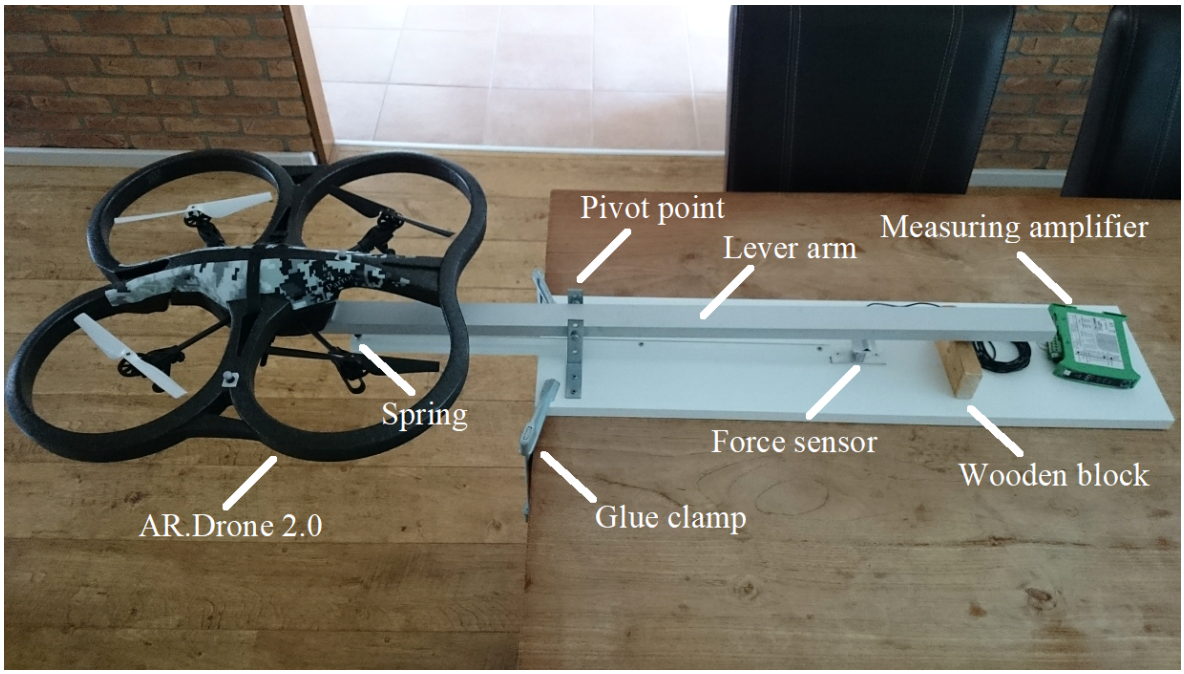


Figure B-5: Thrust setup with important elements highlighted.

B-3-1 Thrust coefficient setup design

Figure B-5 is a copy of Figure 5-8 and shows the thrust setup with important elements in the setup highlighted, which will be elaborated on below.

The basis of the thrust setup consists of a wooden shelf. This shelf is attached to a solid table using glue clamps in order to make sure that the setup cannot move during the experiments.

One of the contributions of these experiments with respect to the experiments performed by [34] and [64] is the fact that the quadrotor is placed in free air. Therefore, a lever arm is used with enough length to let the quadrotor be placed without overlap with the table, including some extra space. The lever arm is constructed using a hollow, rectangular, aluminium profile that, together with the pivot point, ensures stiffness in all directions, except for the rotating direction being leveraged during the experiments. The pivot point splits the complete lever arm into two arms of equal length. The upward force generated by the quadrotor thus equals the downward force exerted on the force sensor.

A disadvantage of constructing the pivot point this way is the extra space needed in the L-profile holes to be able to push a bolt through it and let it rotate with negligible friction. This extra space introduces clearance in vertical direction, resulting in nonlinear behaviour of the force sensor data. Therefore, an extra spring is inserted between pivot point and quadrotor mounting point. This spring ensures that the lever arm is always in the most upward part of the L-profile holes, even when the quadrotor is attached to the lever arm. An extra function of this spring is to ensure that the force on the sensor is always pointing downward, thereby avoiding the need to create a setup that is able to measure positive as well as negative forces. The spring is mounted on another hollow, rectangular, aluminium profile that counteracts the downward force the spring is exposed to when pushing against the lever arm.

(a) COM and lever markers on backside of AR.Drone 2.0.

(b) AR.Drone 2.0 mounted on the lever arm using cable ties with aluminium bar in between.

Figure B-6: Images to show what is needed to properly mount the quadrotor on the lever arm.

With these experiments it is important that the quadrotor is exactly aligned with the lever arm. Therefore, the theoretical COM (determined by taking the crossing of the rotor arms) and the resulting lever arm form are marked on the backside of the quadrotor as shown in Figure B-6a. Using these markers it is possible to mount the quadrotor accurately on the lever arm. The mounting is done using reusable cable ties that turned out to fix the quadrotor enough and avoid quadrotor movement on the arm when the rotors were rotating at relatively high velocities. Because of the fact that the AR.Drone 2.0 backside is a bit compressible, a flat aluminium bar (with a length somewhat larger than the quadrotor backside) is used between the arm and the quadrotor in order to be able to mount the quadrotor horizontally. Figure B-6b shows the quadrotor mounting in a close-up image.

Next to the quadrotor mounting, the most important setup part is the force sensor. The sensor used is an ME KD24s ±100 N force sensor. It is calibrated using calibrated weights at 40 N, which is a proper value compared to the measured force values in Table B-4. Using the rule of thumb saying that the sensor resolution equals the calibrated force divided by 1000, the sensor resolution boils down to a value of 0.04 N. When comparing this value to the force values given in Table B-4, it can be concluded that it is enough resolution to construct a proper relation between rotor rotational velocity and generated thrust. The force sensor data signal goes through Scaime's CPJ measuring amplifier and National Instrument's NI USB-6008 DAQ device to arrive at the VI created in LabView 2018 where it can be read out and saved to a text file (see Figure B-7).

Furthermore, it is important to have axial loading of the sensor, meaning that the force should ideally only have a vertical component and it should be exerted on the middle of the sensor. Due to the relatively long lever arm, the force can be considered vertical. The force is exerted on the sensor centre by making use of 2 triangular aluminium profiles that are oriented 90° with respect to each other (see Figure B-8a).

Moreover, the force range limit of the sensor (before it breaks mechanically) is 200 N [75]. To prevent the setup from accidentally being loaded too much, a wooden block, being higher than the force sensor, is placed besides the sensor to ensure that the lever arm will never

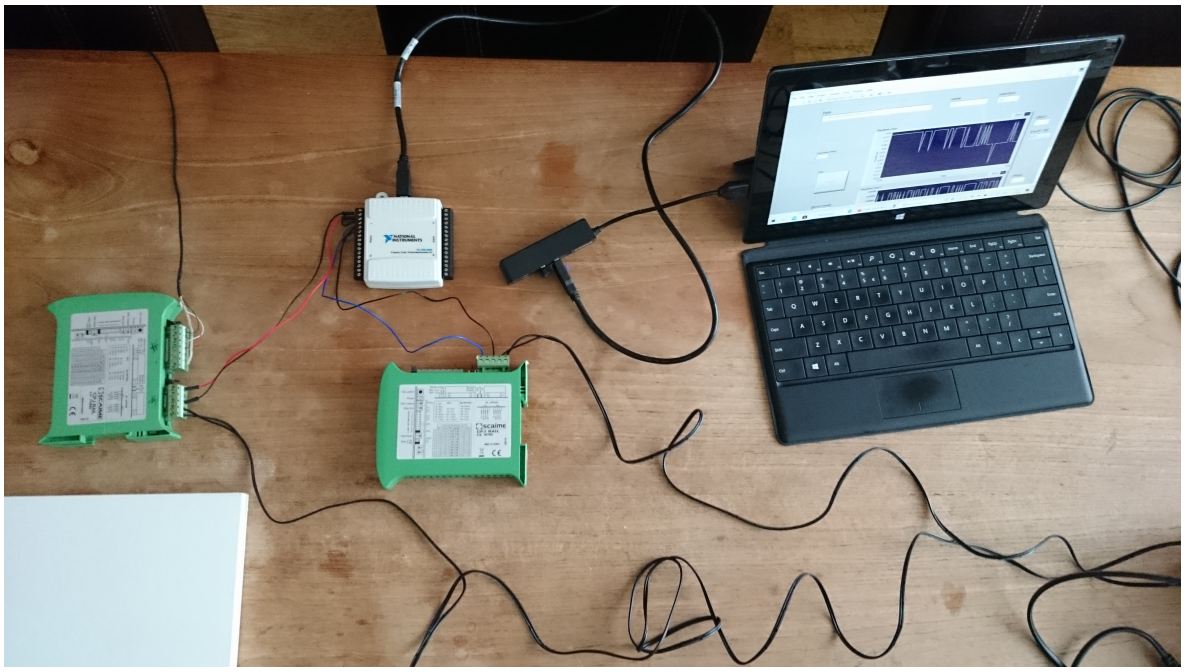
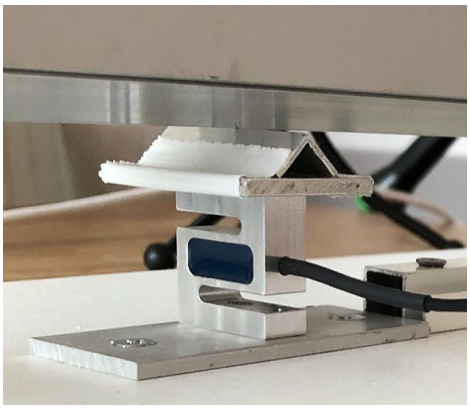


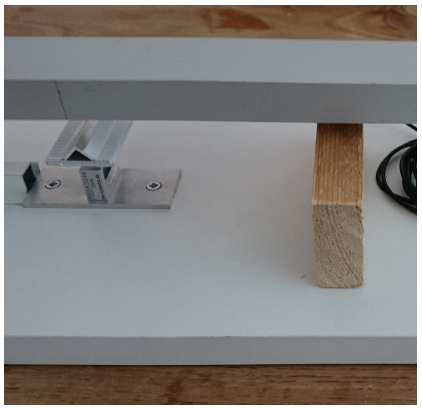
Figure B-7: Sensor data recording using measuring amplifier, data acquisition system and VI in LabView 2018. Special attention is paid to preventing electrical interference for the wires carrying analog force signals by separating the power and force data wires as much as possible.

touch the sensor, except during the experiments (see Figure B-8b).

Important to note is the fact that this thrust setup is based on the principle of relative force measurements. Therefore, the force value measured without any rotating rotor averaged over 10 s is subtracted from all force values measured in the experiment, thereby giving the force the rotating rotors generate. Linearity of the sensor is thus very important. Therefore, a quick check is performed using a spring and a digital kitchen weighing scale that showed a linear relationship between measured weight and force values.



(a) Axially loaded force sensor.



(b) Wooden block to prevent sensor overloading.

Figure B-8: Aspects of the sensor that require attention: axial loading and overloading prevention.

B-3-2 Measurement plan

The following steps need to be taken in order to be able to perform proper thrust coefficient measurements using the thrust setup:

1. Put wooden block next to sensor to prevent overloading.
2. Attach setup to table with glue clamps.
3. Connect force sensor via measuring amplifier and DAQ to VI and ensure that VI can properly read out sensor values.
4. Place fully charged battery in AR.Drone 2.0.
5. Connect with laptop to AR.Drone 2.0 Wi-Fi network.
6. Check connection with AR.Drone 2.0 in MATLAB.
7. Put indoor hull around AR.Drone 2.0 frame.
8. Mount AR.Drone 2.0 on lever arm using the reusable cable ties and put aluminium bar in between. The front camera should point away from the pivot point, according to the lever arm marker on the backside of the quadrotor.
9. Put digital hand-tachometer on tripod and ensure that the red light points to the reflective marker on rotor 4 (see Figure B-9).
10. Put camera on tripod, turn it on and ensure that the hand-tachometer display is clearly visible (see Figure B-9).
11. Create a column space on paper with PWM values 0-100-0 on it to write down the corresponding rotor rotational velocities next to it during the experiment.
12. Build desired Simulink model (let all rotors turn at the same velocity from 0 to 100 PWM and back in steps of 5) and connect to target.
13. Remove wooden block.
14. Start camera recording.
15. Give unique information about the experiment, such that it can be recognized later on.
16. Start running the Simulink model in External mode and simultaneously start logging the force sensor data in the VI.

The experiments are conducted by two persons. One person reads the PWM value for the current 10 s and writes down the corresponding rotational velocity (in RPM) that the other person reads from the hand-tachometer display. These values could be checked later on using the video recording.

Once the 10 s of the last PWM signal have expired, do the following:

Figure B-9: Digital hand-tachometer mounted on tripod with a camera pointing to the tachometer.

1. Simultaneously stop running the Simulink model in External mode and logging the force sensor data.
2. Stop the camera recording.
3. Place the wooden block next to the sensor.
4. Send the sensor data to the laptop running MATLAB.
5. Process and save the logged force sensor (using VI), PWM, battery (using Simulink) and rotational velocity (manually) data in a .mat file.
6. Possibly detach the AR.Drone 2.0 from the lever arm, replace the battery and attach it again to the arm as indicated above.

B-3-3 Datasets

As indicated in Section 5-4-1, only PWM values 5-80 are taken into account in the thrust experiments. Table B-4 provides the resulting dataset.

B-3-4 Validation

Figure B-10 shows each PWM-thrust relation with its linearizations around the two hovering operating points corresponding to a different battery type. As can be seen, the linearization has a larger error for PWM values below the operating point than for PWM values above the operating point. This is one of the causes for the shape of the hidden state process noise, as discussed in Section 6-2-2.

The data used to construct Figure 5-10 and Figure B-10 is recorded during 10 hovering experiments per battery type. For each experiment, a time interval is selected in which the

Table B-4: Data from three experiments to relate generated thrust to rotational velocity. The force values reported here are the result of the total measured force when four rotors are turning, divided by four.

		Experiment number					
		1 (batA)		**2** (batP)		**3** (batP)	
		ω_4 (RPM)	**F** (N)	ω_4 (RPM)	**F** (N)	ω_4 (RPM)	**F** (N)
PWM	**5**	1425	0.147	1418	0.149	1424	0.149
		1426	0.150	1423	0.149	1426	0.148
	10	1608	0.191	1602	0.192	1608	0.193
		1608	0.194	1604	0.194	1608	0.187
	15	1785	0.242	1779	0.237	1785	0.239
		1783	0.238	1779	0.250	1784	0.242
	20	1967	0.298	1962	0.295	1968	0.295
		1965	0.294	1961	0.303	1966	0.293
	25	2143	0.355	2137	0.358	2143	0.352
		2140	0.350	2136	0.356	2141	0.363
	30	2325	0.428	2320	0.424	2325	0.425
		2322	0.421	2317	0.427	2323	0.420
	35	2500	0.504	2494	0.495	2500	0.496
		2497	0.513	2491	0.500	2497	0.494
	40	2682	0.584	2676	0.580	2683	0.573
		2678	0.608	2674	0.577	2679	0.586
	45	2856	0.664	2850	0.656	2856	0.663
		2853	0.666	2847	0.672	2853	0.662
	50	3038	0.759	3032	0.764	3038	0.745
		3033	0.778	3028	0.764	3035	0.767
	55	3218	0.868	3212	0.857	3219	0.822
		3216	0.835	3208	0.865	3214	0.852
	60	3392	0.989	3386	0.980	3391	0.977
		3388	0.961	3382	0.960	3388	0.960
	65	3573	1.069	3565	1.083	3572	1.069
		3570	1.079	3562	1.086	3569	1.086
	70	3746	1.180	3738	1.191	3745	1.195
		3741	1.153	3735	1.199	3742	1.171
	75	3926	1.304	3918	1.309	3925	1.313
		3924	1.290	3915	1.319	3924	1.310
	80	4099	1.444	4090	1.457	4098	1.424
		4097	1.467	4088	1.449	4096	1.443

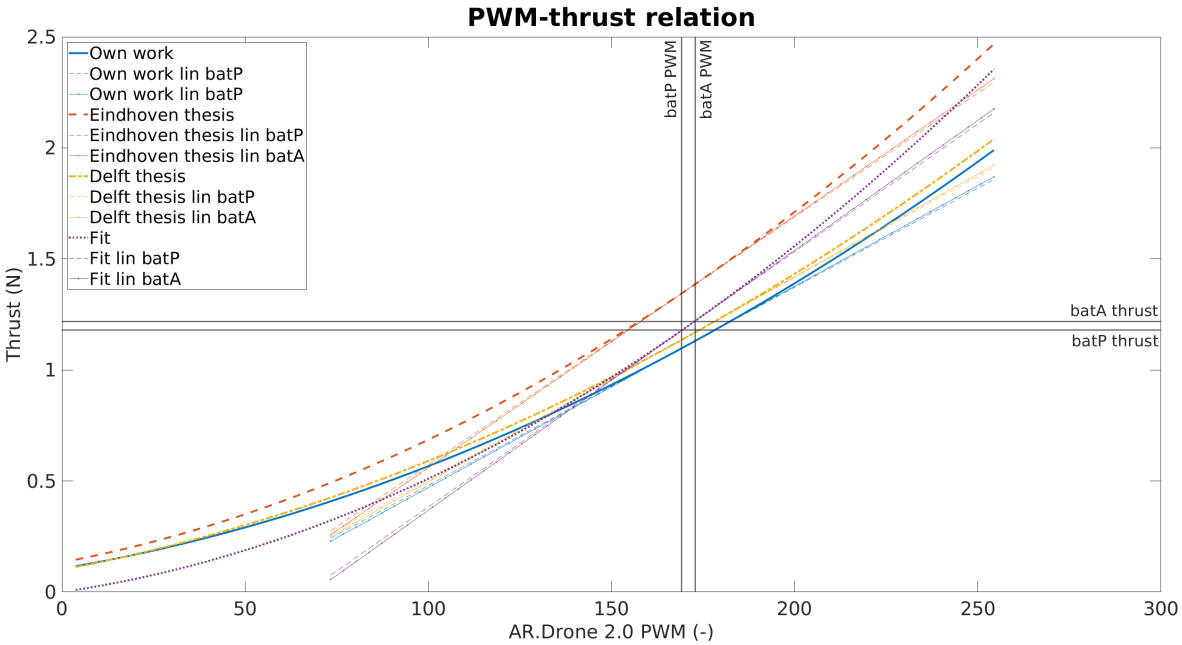


Figure B-10: PWM-thrust hovering operating points and PWM-thrust relation determined in Section 5-4-2 (Own work), [34] (Delft thesis), [64] (Eindhoven thesis) and fitted (Fit) using both operating points, as explained in Section 5-4-2. This figure also adds the linearizations around the batP and batA operating points per PWM-thrust relation.

s

PWM values were approximately constant. Given the fact that air flow circulations and thus low-frequency quadrotor vibrations took place after a constant period of time (first time after approximately 15 s of hovering, second time after 23 s of hovering, caused by the rotating rotor blades and the limited lab space), the averages are taken over maximum time intervals of 15 s. From the 10 experiments, the seven most reliable values (with least PWM variation) were chosen to be used. The thrust values per rotor in the operating points are calculated using $T_i = \frac{mg}{4}$, with m derived from Table 5-1 and g the gravitational acceleration. Table B-5 shows the average PWM values for each operating point.

Table B-5: Average PWM value for each hovering experiment per battery type.

		PWM	
		(batP)	(batA)
Experiment number	**1**	169.5	172.8
	2	169.5	172.6
	3	169.6	173.1
	4	168.2	173.3
	5	168.6	173.1
	6	169.2	173.1
	7	168.8	172.4
	Average	**169.1**	**172.9**

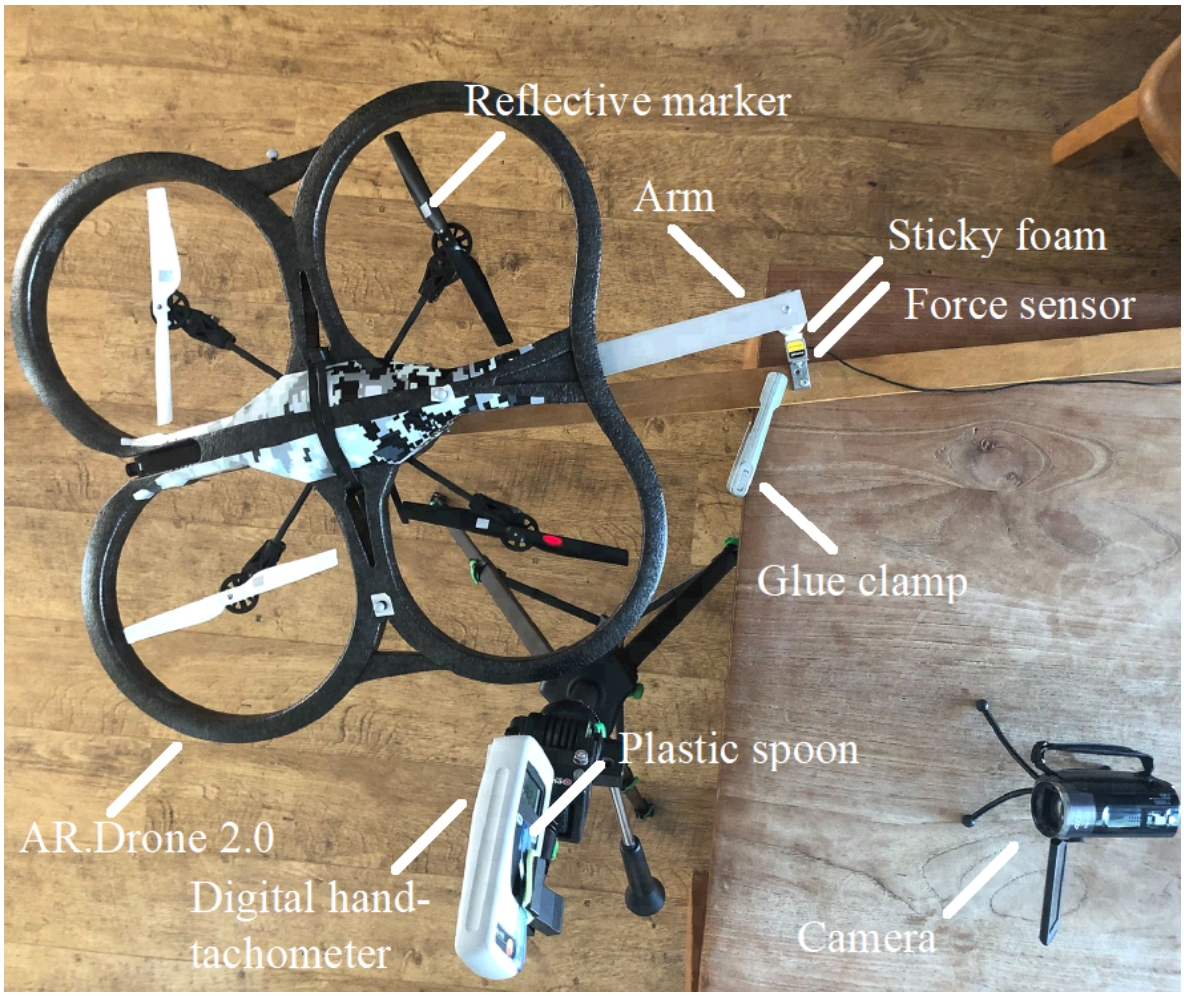


Figure B-11: Torque setup with important elements highlighted.

B-4 Torque coefficient c_Q

This section provides the design of the torque setup in Section B-4-1, together with the measurement plan in Section B-4-2 used to obtain the datasets in Section B-4-3 that are used to visualize the torque coefficient results in Section 5-4-3.

B-4-1 Torque coefficient setup design

Figure B-11 is a copy of Figure 5-11 and shows the torque setup with the important elements highlighted, which will be elaborated on below. The contact point of arm and sensor is displayed in more detail in Figure B-12.

In this case only a wooden bar is needed to serve as basis for the setup, in contrast to the thrust setup. This bar is mounted on the same solid table using a glue clamp.

The AR.Drone 2.0 is mounted on a wooden bar that covers the whole backside of the quadrotor (to avoid compression of the backside and to ensure horizontal alignment of the quadrotor)

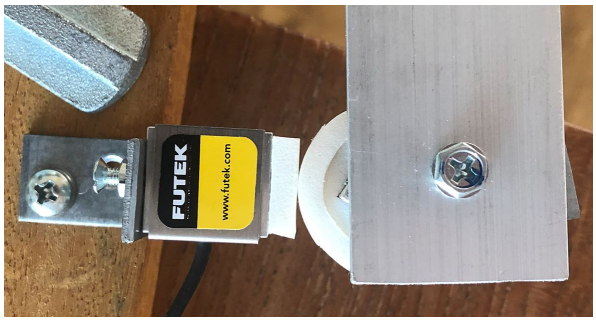


Figure B-12: Sensor contact point in torque setup.

using two cable ties. The wooden block is then mounted on an aluminium bar (serving as arm to generate a force on the sensor) that is stiff in the rotation direction, except for the direction of desired rotation. The block-bar combination is then mounted on the wooden basis of the setup via the ball bearing of an inline skate wheel. This way, the torque generated by the turning rotors translate into a rotating movement of the arm against the force sensor. Using the force value and arm length, the exerted torque can be calculated.

The force values as a result of torque generation by the quadrotor are smaller than the force values in the thrust experiments (compare the values in Table B-4 and Table B-7), so a force sensor with a smaller range (FUTEK FSH02664) is used. This sensor has a mechanical 10x overloading protection [76]. Furthermore, it has a nominal force range of 100 g (corresponding to 1 N), but is calibrated at 30 g (corresponding to 0.3 N). Using the same rule of thumb as before, this would give us a resolution of $3 \cdot 10^{-4}$ N. Both force range and resolution are sufficient for the measured force values provided in Table B-7.

As in the case of the thrust setup, the small force sensor should also be axially loaded. This is ensured by mounting it to an L-profile that is mounted on the wooden setup basis and can be rotated in the right way to ensure a right angle with the L-profile with triangular aluminium bar that is mounted on the arm (see Figure B-12). Similar to the thrust experiments, the triangular bar ensures that the sensor is axially loaded.

During torque measurements, the quadrotor introduces vibrations in the direction of rotation. To reduce the impact of vibrations on the experiment results, sticky foam is placed on both sides of the contact point of arm and sensor (see Figure B-12). This will damp the vibrations and lower the vibration frequency. When averaging the force data over 10 s, the vibrations are assumed to be filtered out.

To read out the force values of this sensor, the same type of measuring amplifier, the same DAQ and the same VI are used as for the thrust experiments.

The torque experiments are conducted by providing the same PWM signal as used for the thrust experiments to one pair of diagonally located rotors (either rotor 1 and 3, or rotor 2 and 4). This way, the total torque is generated around the theoretical COM of the quadrotor, which is exactly where it is mounted on the ball bearing. The total measured torque can be divided by two to get the torque for each individual rotor. The good thing about this setup is the fact that the L-profile can be turned 180°. This way, both sets of rotors (each set generating a torque in the opposite direction) can be used to validate the torque coefficient results. As shown in Table B-7, this experiment is conducted for both sets of rotors.

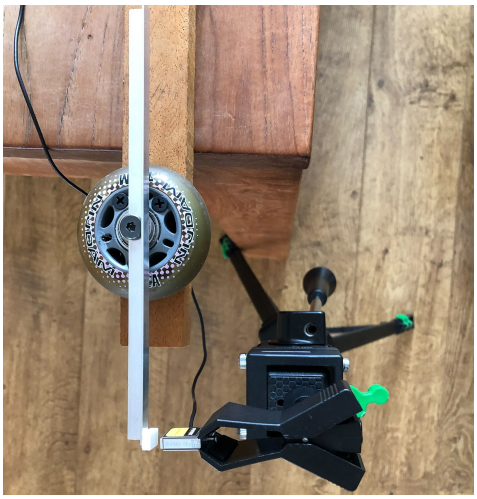


Figure B-13: Setup to determine friction of ball bearing.

Finally, the friction of the ball bearing is taken into account, because it is an unknown quantity that may possibly not be neglected. The setup is created by mounting an aluminium L-profile bar on the ball bearing, such that it has equal lengths (10 cm) on both sides of the bearing, as shown in Figure B-13. Equal lengths ensure that the effect of gravity is reduced in case the setup is not exactly mounted horizontally. A relatively short bar is chosen to reduce the inertia effects. To prevent from bouncing effects that will otherwise occur, a foam layer is attached to the bar on the point of contact with the sensor. In order to determine friction, the sensor is slowly pushed against the bar until the point that it has just started to move. This is performed eight times. The absence of a clear initial peak in the force data indicates that static friction is negligible and only the dynamic friction should be taken into account. Since the sensor data in the torque setup is vibrating, it is reasonable to assume that only dynamic friction needs to be taken into account. The average force measured during movement of the bar is shown in Table B-6 for all eight measurements. The average force value of all experiments is taken to calculate the torque that is subtracted from the torque measurement data from the torque experiments with the AR.Drone 2.0.

Table B-6: Results of ball bearing friction experiments.

		F (N)
Experiment number	**1**	0.0021
	2	0.0018
	3	0.0018
	4	0.0018
	5	0.0019
	6	0.0018
	7	0.0019
	8	0.0020
	Average	**0.0019**

B-4-2 Measurement plan

The following steps need to be taken in order to be able to perform proper torque coefficient measurements using the torque setup (quite similar to the steps for the thrust setup, but listed here for the sake of completeness):

1. Attach setup to table with glue clamps.
2. Mount force sensor on L-profile and rotate 180°, such that it cannot be loaded accidentally when mounting the quadrotor.
3. Connect force sensor via measuring amplifier and DAQ to VI and ensure that VI can properly read out sensor values.
4. Place fully charged battery in AR.Drone 2.0.
5. Connect with laptop to AR.Drone 2.0 Wi-Fi network.
6. Check connection with AR.Drone 2.0 in MATLAB.
7. Put indoor hull around AR.Drone 2.0 frame.
8. Mount AR.Drone 2.0 on wooden block and aluminium arm using the reusable cable ties and several screws. Mount this part on the ball bearing of the inline skate wheel. Check if the front camera points away from the sensor contact point.
9. Put digital hand-tachometer on tripod and ensure that the red light points to the reflective marker on rotor 4. For easiness, ensure that the red light remains on by fixing a plastic spoon that pushes the 'on'-button of the hand-tachometer (see Figure B-11).
10. Put camera on tripod, turn it on and ensure that the hand-tachometer display is clearly visible (see Figure B-11).
11. Create a column space on paper with PWM values 0-100-0 on it to write down the corresponding rotor rotational velocities next to it during the experiment.
12. Build desired Simulink model (let rotor 1 and 3 or rotor 2 and 4 turn at the same velocity from 0 to 100 PWM and back in steps of 5) and connect to target.
13. Rotate force sensor back, such that it makes a 90° angle with the arm.
14. Start camera recording.
15. Give unique information about the experiment, such that it can be recognized later on.
16. Start running the Simulink model in External mode and simultaneously start logging the force sensor data in the VI.

The experiments are conducted by two persons. One person reads the PWM value for the current 10 s and writes down the corresponding rotational velocity (in RPM) that the other person reads from the hand-tachometer display. These values could be checked later on using the video recording.

Once the 10 s of the last PWM signal have expired, do the following:

1. Simultaneously stop running the Simulink model in External mode and logging the force sensor data.
2. Stop the camera recording.
3. Rotate the sensor 180°.
4. Remove plastic spoon from hand-tachometer.
5. Send the sensor data to the laptop running MATLAB.
6. Process and save the logged force sensor (using VI), PWM, battery (using Simulink) and rotational velocity (manually) data in a .mat file.
7. Possibly detach the AR.Drone 2.0 from the arm, replace the battery and attach it again to the arm as indicated above.

B-4-3 Datasets

As indicated in Section 5-4-1, only PWM values 5-80 are taken into account in the torque experiments. Table B-7 provides the resulting dataset.

Table B-7: Data from four experiments to relate generated torque to rotational velocity. The torque values reported here are the result of the total measured torque when two rotors are turning, minus torque caused by dynamic friction, divided by two.

		Experiment number							
		1 (batP)		**2** (batP)		**3** (batP)		**4** (batP)	
		ω_4 (RPM)	τ (N m)	ω_4 (RPM)	τ (N m)	ω_3 (RPM)	τ (N m)	ω_3 (RPM)	τ (N m)
PWM	**5**	1411	0.0040	1429	0.0039	1422	0.0039	1420	0.0042
		1421	0.0040	1426	0.0049	1420	0.0040	1424	0.0040
	10	1596	0.0052	1612	0.0050	1605	0.0054	1604	0.0055
		1603	0.0050	1608	0.0063	1602	0.0051	1605	0.0055
	15	1774	0.0066	1788	0.0066	1781	0.0068	1781	0.0067
		1778	0.0064	1783	0.0077	1777	0.0064	1781	0.0065
	20	1957	0.0081	1970	0.0079	1963	0.0082	1964	0.0079
		1960	0.0082	1965	0.0095	2958	0.0083	1962	0.0080
	25	2133	0.0095	2145	0.0096	2138	0.0102	2140	0.0102
		2134	0.0097	2140	0.0108	3132	0.0100	2137	0.0095
	30	2315	0.0110	2327	0.0110	2319	0.0121	2322	0.0123
		2315	0.0117	2322	0.0134	2313	0.0114	2318	0.0120
	35	2490	0.0132	2502	0.0127	2495	0.0138	2496	0.0149
		2490	0.0135	2496	0.0150	2487	0.0136	2492	0.0135
	40	2672	0.0148	2685	0.0142	2676	0.0163	2679	0.0168
		2671	0.0159	2678	0.0171	2669	0.0166	2674	0.0161
	45	2845	0.0171	2858	0.0164	2850	0.0190	2853	0.0193
		2845	0.0178	2853	0.0195	2842	0.0196	2849	0.0187
	50	3026	0.0195	3038	0.0190	3029	0.0221	3033	0.0222
		3026	0.0205	3033	0.0225	3024	0.0211	3030	0.0218
	55	3207	0.0226	3219	0.0230	3209	0.0244	3213	0.0250
		3205	0.0263	3214	0.0257	3204	0.0252	3210	0.0242
	60	3382	0.0266	3393	0.0270	3382	0.0270	3387	0.0271
		3379	0.0258	3388	0.0286	3377	0.0290	3384	0.0274
	65	3561	0.0293	3574	0.0287	3560	0.0315	3567	0.0305
		3560	0.0303	3569	0.0324	3558	0.0299	3564	0.0306
	70	3734	0.0341	3749	0.0337	3733	0.0341	3740	0.0335
		3732	0.0345	3741	0.0339	3730	0.0332	3737	0.0344
	75	3914	0.0361	3928	0.0361	3913	0.0367	3920	0.0376
		3913	0.0363	3922	0.0368	3910	0.0366	3917	0.0370
	80	4088	0.0398	4100	0.0388	4086	0.0409	4093	0.0403
		4085	0.0389	4096	0.0413	4083	0.0399	4091	0.0404

Appendix C

Experimental setup

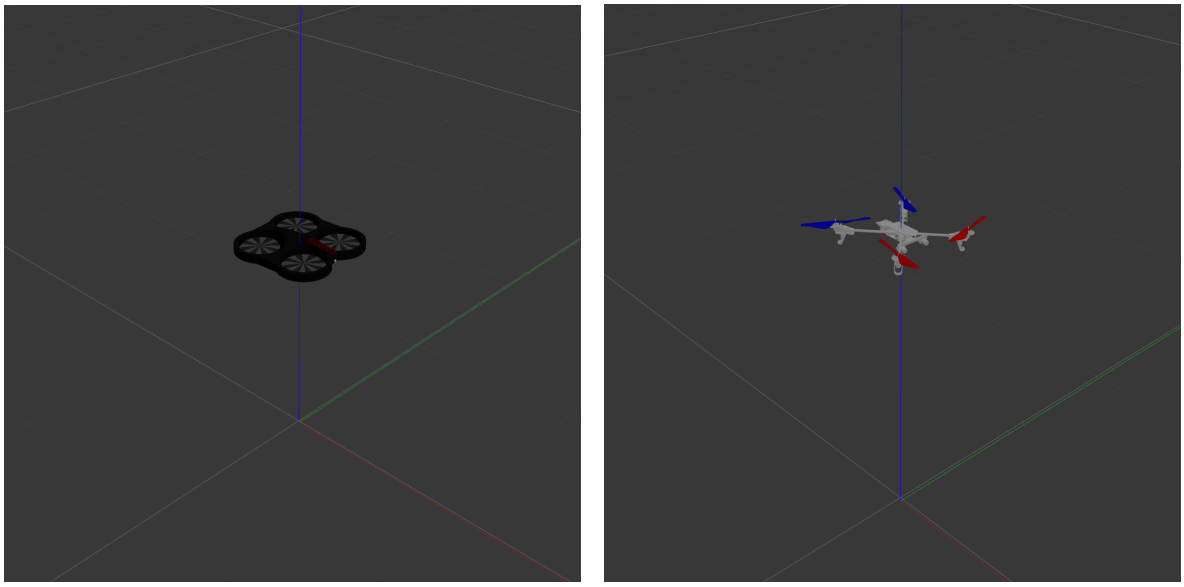
In this appendix, Section C-1 shows two different simulators that could be used to simulate the Parrot AR.Drone 2.0 in future research and Section C-2 gives the experiment plan used in this thesis to ensure consistent and well-documented lab data.

C-1 Parrot AR.Drone 2.0 Gazebo simulators

For the Parrot AR.Drone 2.0, two different Gazebo simulators could be used.

The first simulator is called the *tum_simulator* [77]. Figure C-1a shows the hovering AR.Drone 2.0 in simulation. The main advantage of this simulator is the fact that it can be used by the *tum_ardrone* package and directly substitutes the *ardrone_autonomy* package (used to communicate with the real AR.Drone 2.0) without interfacing problems. Another advantage is the fact that the wind plugin of the *rotors_simulator* simulator (described below) is made compatible with this simulator in this thesis, thereby being able to simulate the quadrotor with influence of wind. The disadvantage of this simulator is the fact that it is meant for research in the area of computer vision. Therefore, the simulator movements are based on forces and torques acting on the simulated quadrotor rigid-body. These forces and torques could be used as alternative model inputs for the model described in Section 3-4. However, these signals appear to contain very spiky behaviour. An example of spiky torque data is shown in Figure C-2. Therefore, it is recommended not to use this simulator for future research, unless a completely different quadrotor model is used that does not involve the rigid-body thrust and torques and/or motor PWM values.

The other simulator is called the *rotors_simulator* [78] and is shown in Figure C-1b. The *rotors_simulator* is one of a diversity of ROS packages built by the Autonomous Systems Lab (ASL) in Eidgenössische Technische Hochschule (ETH) Zürich. The main advantage of this simulator is the fact that it simulates quadrotor movement based on the rotor velocities (instead of directly using thrust and torques on a rigid-body as described above). This gives more realistic quadrotor behaviour. Another advantage of this package is the fact that it is not limited to simulation of the Parrot AR.Drone 2.0, but can also simulate other



(a) tum_simulator. **(b)** rotors_simulator.

Figure C-1: View on quadrotor hovering at position $(x = 0, y = 0, z = 1)$ in tum_simulator and rotors_simulator.

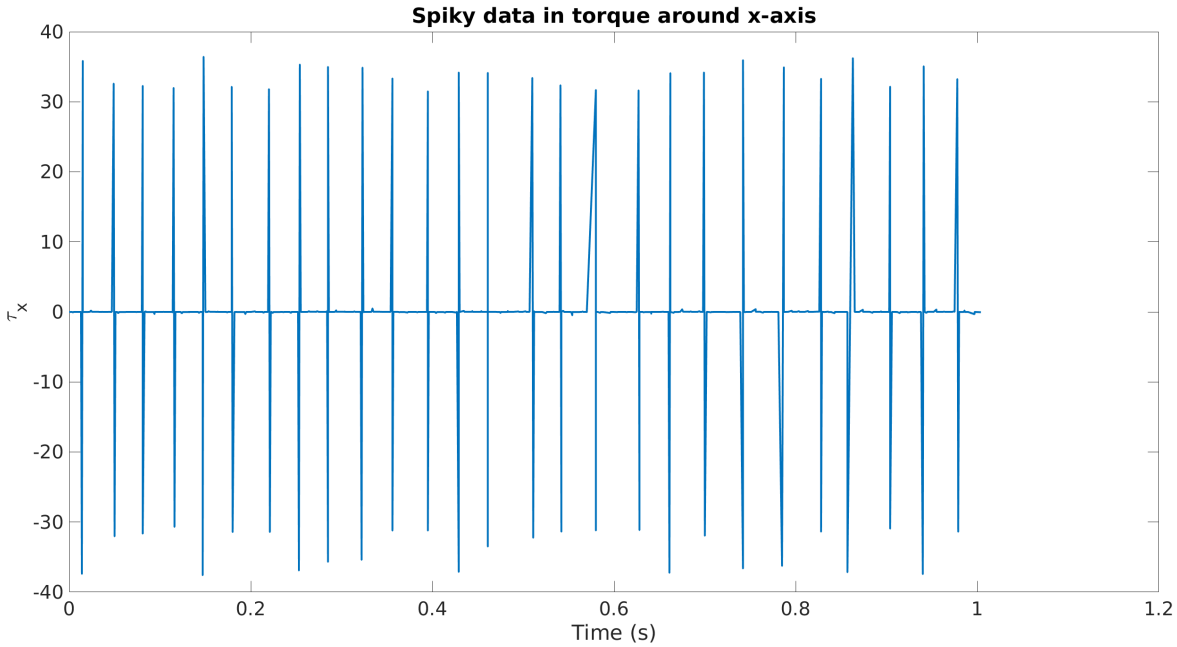


Figure C-2: Spiky torque data using tum_simulator.

quadrotors. This makes the simulator suitable for future research with potentially different quadrotors. This package also has disadvantages. First of all, it contains a lot of quadrotor parameters that should all be known in order to simulate it properly. In itself, this is not a real disadvantage, but the fact that the package simulates a quadrotor called 'ardrone', not 'ardrone2', makes it more difficult. One should find out whether the quadrotor behaviour between AR.Drone and AR.Drone 2.0 differs much and in what respect in order to properly adjust the simulation model parameters of the quadrotor. Another disadvantage is the fact that this simulator contains an AR.Drone model without indoor hull, which could possibly introduce a significant discrepancy between simulation physical flight. The indoor hull could possibly be taken from a file defining the indoor hull mesh in the *tum_simulator* package.

It can be concluded that both simulators have their own advantages and disadvantages. However, the spiky data coming from the *tum_simulator* introduces such an issue to state estimation that the *rotors_simulator* is the preferred simulator to use for future research.

C-2 Experiment plan

This section describes a universal experiment plan that can be used to conduct flight experiments with the AR.Drone 2.0 in the NERDlab using the experimental setup as described in Chapter 4.

First of all, the following materials should be taken to the lab:

1. Laptop with software installed
2. Parrot AR.Drone 2.0
3. Batteries
4. Battery charger(s)
5. Joystick
6. Universal Serial Bus (USB) extension cable (to always have the joystick controller in Line Of Sight (LOS) with its USB receiver)
7. USB Wi-Fi adapter
8. Video camera with sufficient storage space
9. Tripod for video camera
10. OptiTrack markers
11. Double-sided tape
12. Black tape
13. Scissors
14. Spare parts:

(a) Parrot AR.Drone 2.0

(b) Batteries

The following steps need to be taken before going to the lab:

1. Reserve time slot in lab using online booking system and inform lab supervisor if needed.
2. Charge all batteries.
3. Pack all items needed in the lab as given above.
4. Prepare software for direct usage.

The following steps need to be taken once before going to the lab:

1. Ensure that quadrotor can be powered on when aligned with OptiTrack coordinate axes (using replaceable part of the indoor hull shown in Figure C-3).

The following steps need to be taken once in the lab before being able to retrieve usable data from the flight experiments:

1. Set up network connection with quadrotor via routers to significantly reduce the amount of reconnection timeouts with AR.Drone 2.0 (see Figure C-4).
2. Set up Motive:Tracker program using a dedicated manual [79].
3. Determine origin and orientation of OptiTrack coordinate axes, configure OptiTrack with this setting and put tapes on the ground with indication of x- and y-axis and with indication of quadrotor leg positions.

The following steps need to be taken before conducting flight experiments in the lab:

1. Put the window screens down (to avoid ground reflections for better optical flow velocity estimation).
2. Put clear markers on the ground to improve optical flow velocity estimation.
3. Place the quadrotor with markers in the OptiTrack camera FOV.
4. Start Motive:Tracker on TUD desktop and check whether the recognized rigid-body is vibrating or not. If it is vibrating, consult the average marker position error as indicated in the program. If this value is small, continue. Otherwise, consider recreating the rigid-body with the markers.
5. Set up the lab network according to Figure C-4 by connecting router 1 and router 2 via Ethernet cables to the laptop.
6. Turn on the quadrotor and make sure it connects with router 2.
7. Check connection using joystick control.

8. Perform a test flight to ensure that a ROS *bag* file is saved with at least data from the the ROS topics listed below and to check if the data looks as expected (i.e. properly represents the quadrotor movement and has expected sampling frequency).
    - *ardrone/imu*
    - *ardrone/navdata*
    - *ardrone/odometry*
    - *ardrone2/pose*
    - *cmd_vel*
    - *tf*
9. Set up the camera on the tripod in a useful location and turn it on.

Steps to take for every flight experiment (outlined for the experiment considered in this thesis below):

1. Place quadrotor at the origin of the OptiTrack system (tape cross on the ground) with correct orientation.
2. If OptiTrack position and orientation are not zero: set these quantities to zero in Motive:Tracker.
3. Turn on the quadrotor and make sure it connects with router 2.
4. Place wind source in lab and power on.
5. Start camera recording and indicate which experiment will take place.
6. Perform experiment:
    (a) Take off and fly quadrotor to a position approximately 6 m away from the wind source and keep it hovering at that position.
    (b) Turn on the wind source to mode 2.
    (c) Wait until the air flow coming from the wind source is stable.
    (d) Keep the experiment setting constant for a desired duration (e.g. at least 30 s).
    (e) Turn off the wind source to mode 0.
    (f) Wait until the quadrotor movement is stable again.
    (g) Land quadrotor.
7. Stop video recording.
8. Note the experiment number, goal and whether or not it was successful.
9. Rename the ROS *bag* file according to the following naming convention:
   ardrone2_exp_yyyy-mm-dd_expnumber_bat#.bag
   ('ardrone2' to indicate which quadrotor is flying, 'exp' to indicate that it is an experiment and not a simulation, replace 'yyyy-mm-dd' with the year, month and day, replace 'expnumber' with the number of the experiment during that specific lab session, replace 'bat#' with the battery type and number (e.g. batP1 or batA2).

Figure C-3: Replaceable part of indoor hull to be able to power on the quadrotor when aligned with OptiTrack coordinate axes.

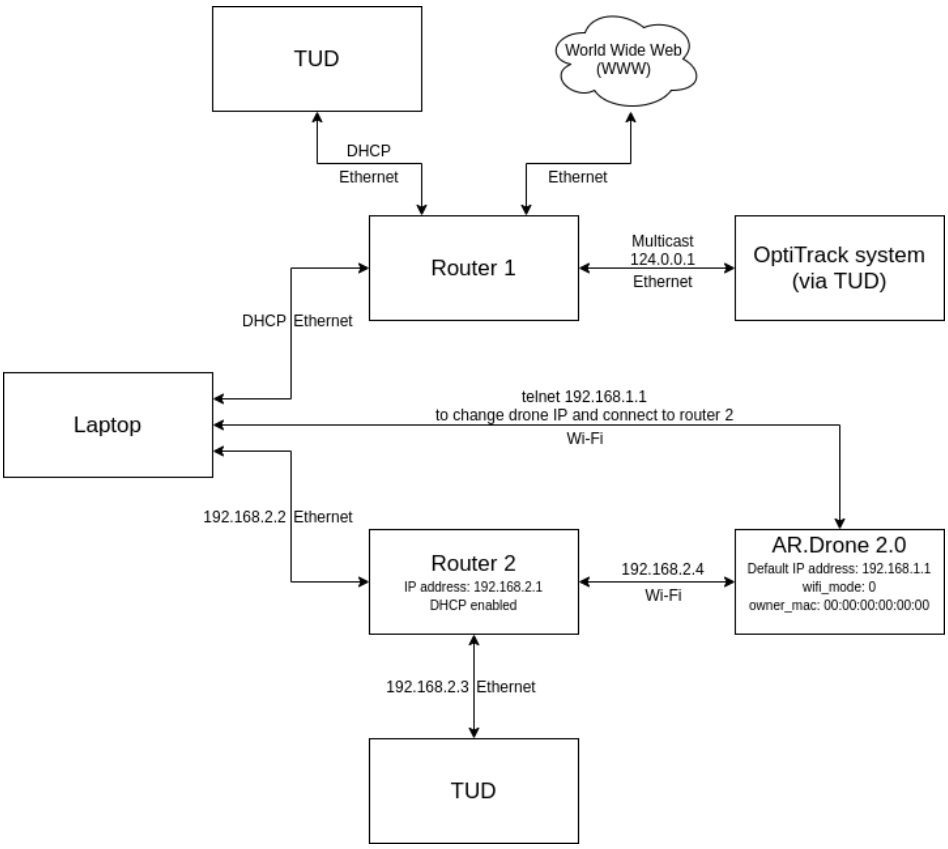


Figure C-4: Lab network configuration.

Appendix D

Noise analysis

This appendix provides figures supporting the statements in Chapter 6.

D-1 Distribution of higher-order derivatives of measurement noise

Figure D-1 and Figure D-2 show the distribution of the 3rd- and 4th-order and 5th- and 6th-order derivatives of measurement noise z, respectively.

D-2 Distribution of higher-order derivatives of process noises

Figure D-3 and Figure D-3 show the distribution of the 3rd- and 4th-order and 5th- and 6th-order derivatives of process noise w_1, respectively.

Figure D-5 and Figure D-5 show the distribution of the 3rd- and 4th-order and 5th- and 6th-order derivatives of process noise w_2, respectively.

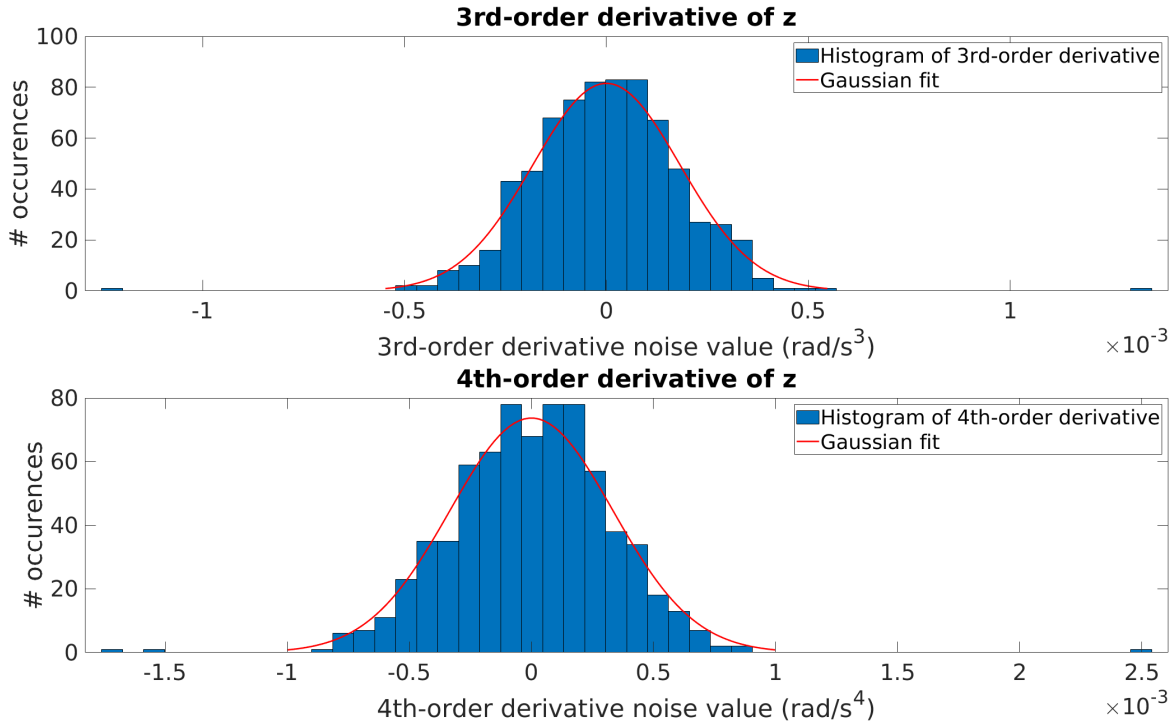


Figure D-1: Distributions of 3rd- and 4th-order derivatives of z, together with their corresponding Gaussian fit.

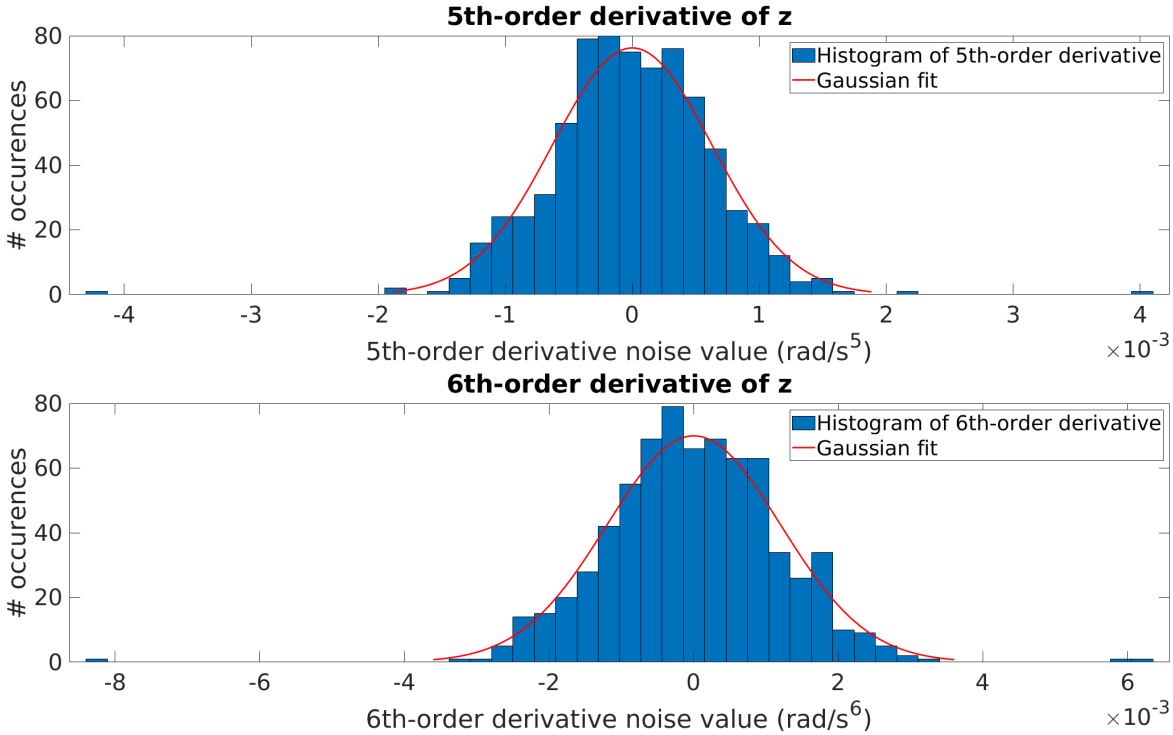


Figure D-2: Distributions of 5th- and 6th-order derivatives of z, together with their corresponding Gaussian fit.

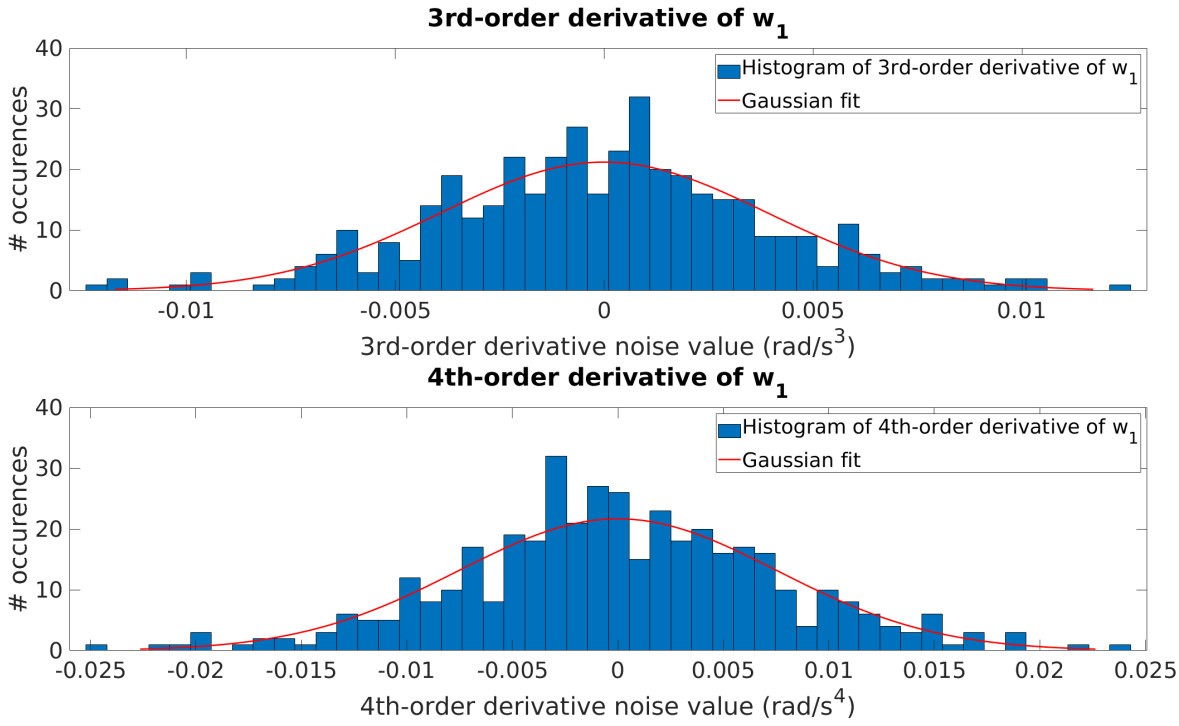


Figure D-3: Distributions of 3rd- and 4th-order derivatives of w_1, together with their corresponding Gaussian fit.

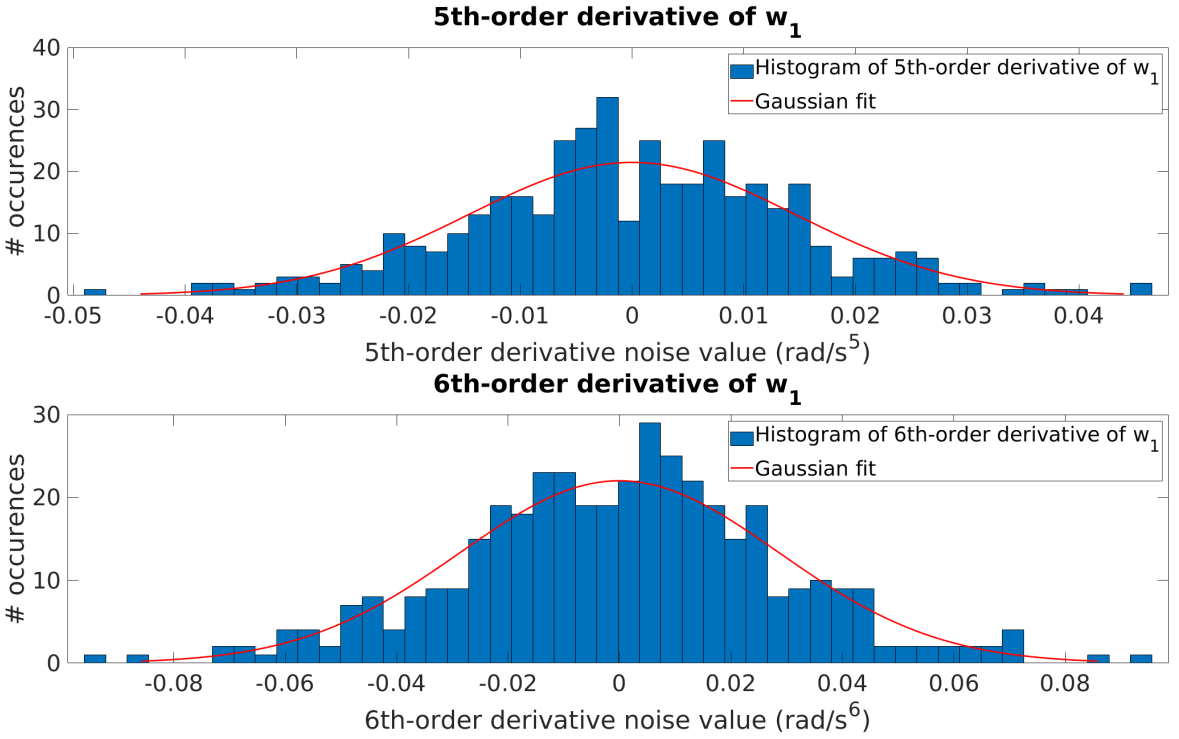


Figure D-4: Distributions of 5th- and 6th-order derivatives of w_1, together with their corresponding Gaussian fit.

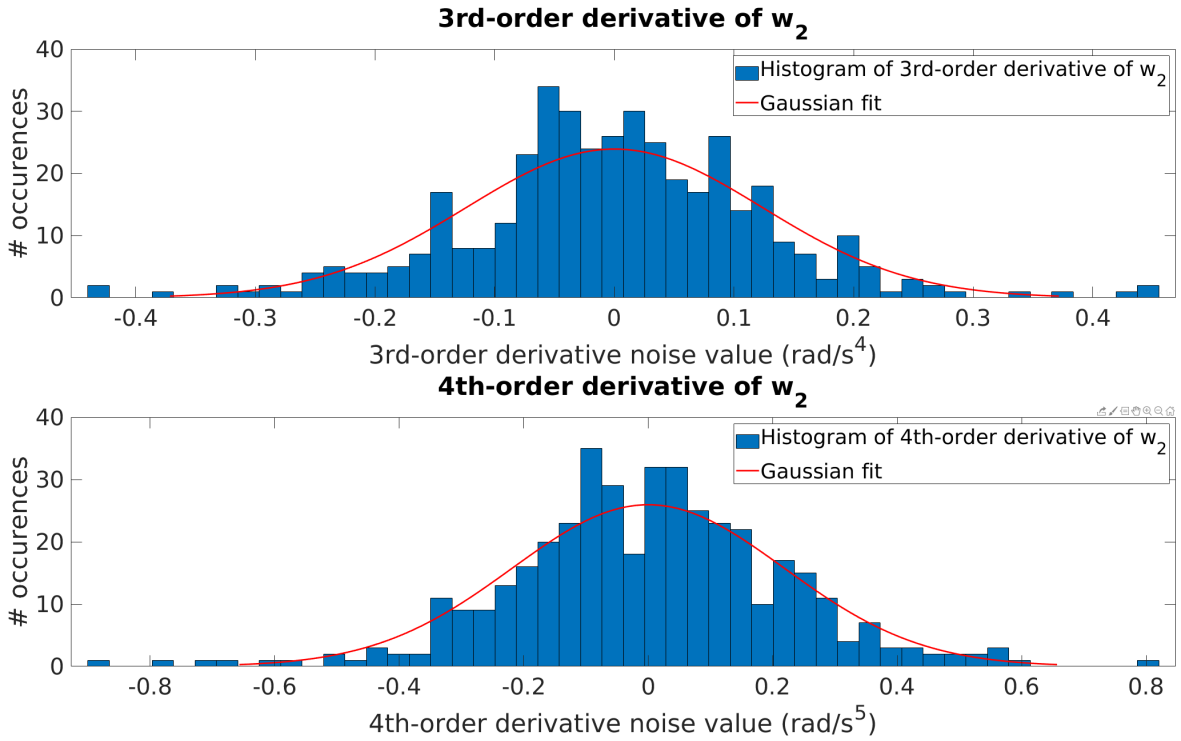


Figure D-5: Distributions of 3rd- and 4th-order derivatives of w_2, together with their corresponding Gaussian fit.

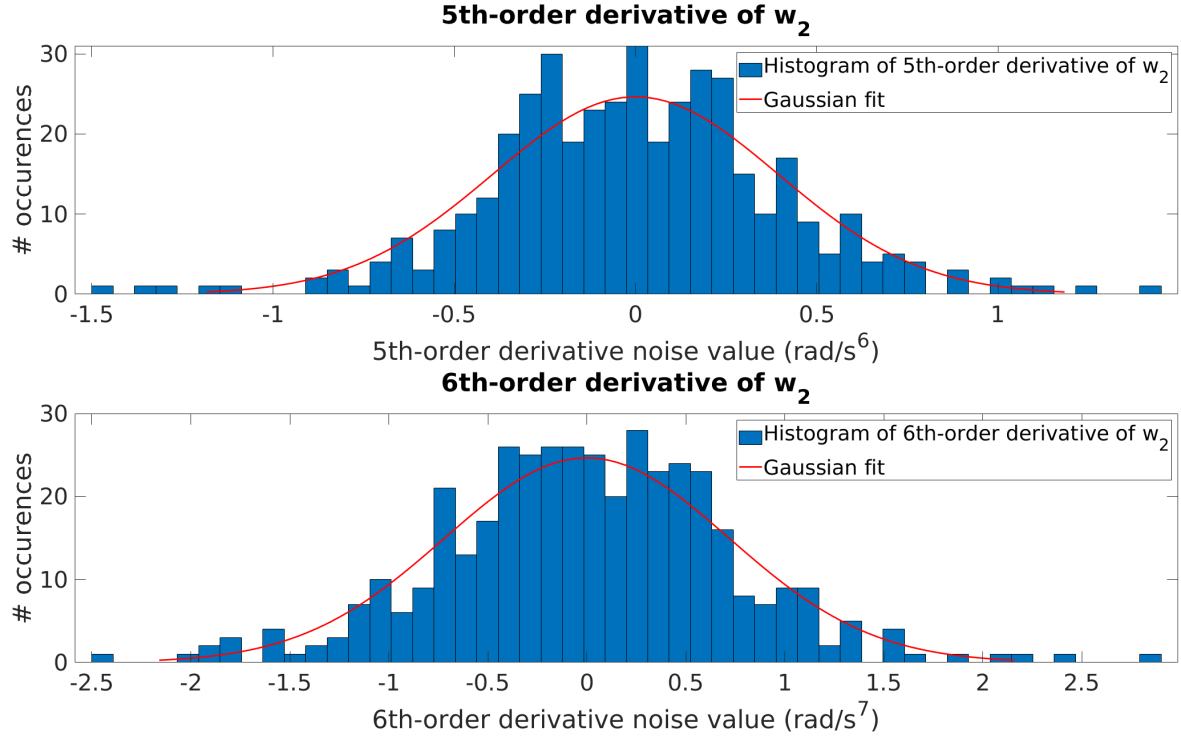


Figure D-6: Distributions of 5th- and 6th-order derivatives of w_2, together with their corresponding Gaussian fit.

Appendix E

Filter results

This appendix provides the data supporting the main conclusions drawn in Chapter 7 regarding the performance of the DEM filter.

E-1 DEM filter results for different p, d and s values

Table E-1 until E-24 show the SSE values of the DEM filter for each combination of embedding orders p and d, ranging from 0 up to and including 7 per smoothness value.

Table E-1: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 1 \cdot 10^{-4}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	$6.31 \cdot 10^3$	$6.31 \cdot 10^3$	$6.32 \cdot 10^3$	$1.55 \cdot 10^3$	$1.55 \cdot 10^3$	637	638	329
	2	$6.81 \cdot 10^3$	$6.81 \cdot 10^3$	$6.81 \cdot 10^3$	$1.83 \cdot 10^3$	$1.83 \cdot 10^3$	808	809	445
	3	$1.83 \cdot 10^3$	$1.83 \cdot 10^3$	$1.83 \cdot 10^3$	$1.83 \cdot 10^3$	$1.83 \cdot 10^3$	809	810	446
	4	$1.83 \cdot 10^3$	$1.83 \cdot 10^3$	$1.83 \cdot 10^3$	$1.83 \cdot 10^3$	$1.83 \cdot 10^3$	810	811	447
	5	812	812	812	812	812	812	812	448
	6	815	814	814	814	814	814	814	450
	7	452	452	452	452	452	452	452	452

Table E-2: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 2 \cdot 10^{-4}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	733	733	735	160	161	59.8	60.0	29.1
	2	921	920	920	236	236	101	101	54.1
	3	237	237	237	237	237	101	101	54.5
	4	238	238	238	238	238	101	102	54.9
	5	102	102	102	102	102	102	102	55.5
	6	103	103	103	103	103	103	103	56.2
	7	57.2	57.1	57.1	57.1	57.1	57.1	57.1	57.1

Table E-3: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 3 \cdot 10^{-4}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	191	191	192	38.0	38.2	14.7	14.7	8.25
	2	277	277	277	68.2	68.4	29.6	29.6	17.2
	3	68.7	68.6	68.7	68.6	68.8	30.0	30.0	17.6
	4	69.3	69.2	69.2	69.2	69.2	30.3	30.3	17.8
	5	30.8	30.8	30.8	30.8	30.8	30.8	30.8	18.3
	6	31.5	31.5	31.5	31.5	31.5	31.5	31.5	18.8
	7	19.6	19.5	19.5	19.5	19.5	19.5	19.5	19.5

Table E-4: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 4 \cdot 10^{-4}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	69.4	69.4	70.0	14.2	14.3	6.83	6.84	5.11
	2	115	115	115	28.9	28.9	14.1	14.1	9.64
	3	29.3	29.3	29.3	29.3	29.3	14.5	14.5	10.2
	4	29.7	29.6	29.6	29.6	29.6	14.7	14.7	10.2
	5	15.2	15.1	15.1	15.1	15.1	15.1	15.2	10.7
	6	15.7	15.6	15.6	15.6	15.6	15.6	15.6	11.0
	7	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7

Table E-5: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 5 \cdot 10^{-4}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	31.3	31.3	31.6	7.69	7.71	5.02	5.02	4.64
	2	58.2	58.1	58.1	16.0	16.0	9.34	9.35	7.26
	3	16.5	16.4	16.4	16.4	16.5	9.88	9.89	7.97
	4	16.7	16.7	16.7	16.7	16.7	9.93	9.94	7.88
	5	10.4	10.4	10.4	10.4	10.4	10.4	10.4	8.37
	6	10.8	10.7	10.7	10.7	10.7	10.7	10.7	8.62
	7	9.28	9.27	9.27	9.27	9.27	9.27	9.27	9.27

Table E-6: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 6 \cdot 10^{-4}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	16.7	16.7	16.9	5.55	5.55	4.65	4.65	4.74
	2	33.7	33.6	33.6	10.9	11.0	7.41	7.41	6.15
	3	11.5	11.4	11.4	11.4	11.4	8.10	8.11	7.04
	4	11.6	11.6	11.6	11.6	11.6	8.03	8.04	6.82
	5	8.52	8.51	8.51	8.51	8.51	8.51	8.52	7.39
	6	8.79	8.78	8.78	8.78	8.78	8.78	8.78	7.54
	7	8.22	8.22	8.22	8.22	8.22	8.22	8.22	8.22

Table E-7: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 7 \cdot 10^{-4}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	10.4	10.4	10.5	4.83	4.83	4.70	4.70	4.92
	2	21.8	21.8	21.8	8.60	8.61	6.39	6.40	5.47
	3	9.20	9.18	9.18	9.18	9.20	7.24	7.25	6.50
	4	9.23	9.21	9.22	9.22	9.22	7.05	7.06	6.18
	5	7.61	7.60	7.60	7.60	7.60	7.60	7.61	6.82
	6	7.79	7.78	7.78	7.78	7.78	7.78	7.78	6.90
	7	7.60	7.60	7.60	7.60	7.60	7.60	7.60	7.60

Table E-8: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 8 \cdot 10^{-4}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	7.45	7.45	7.49	4.64	4.64	4.85	4.84	5.04
	2	15.6	15.6	15.6	7.33	7.34	5.75	5.75	5.01
	3	8.04	8.03	8.03	8.03	8.05	6.72	6.73	6.11
	4	7.98	7.97	7.97	7.97	7.97	6.44	6.44	5.73
	5	7.06	7.05	7.05	7.05	7.05	7.05	7.06	6.40
	6	7.17	7.17	7.17	7.17	7.17	7.17	7.17	6.46
	7	7.16	7.16	7.16	7.16	7.16	7.16	7.16	7.16

Table E-9: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 9 \cdot 10^{-4}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.96	5.96	5.98	4.67	4.67	4.98	4.97	5.08
	2	12.1	12.1	12.1	6.54	6.55	5.29	5.29	4.67
	3	7.37	7.36	7.36	7.36	7.38	6.35	6.35	5.79
	4	7.21	7.21	7.21	7.21	7.21	6.00	6.00	5.39
	5	6.66	6.66	6.66	6.66	6.66	6.66	6.66	6.07
	6	6.73	6.73	6.73	6.73	6.73	6.73	6.73	6.12
	7	6.81	6.81	6.81	6.81	6.81	6.81	6.81	6.81

Table E-10: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 1 \cdot 10^{-3}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.21	5.21	5.22	4.78	4.78	5.05	5.05	5.08
	2	10.0	10.0	10.0	5.99	6.00	4.94	4.94	4.43
	3	6.93	6.92	6.92	6.92	6.93	6.05	6.05	5.53
	4	6.69	6.68	6.68	6.68	6.68	5.66	5.66	5.14
	5	6.34	6.34	6.34	6.34	6.34	6.34	6.34	5.81
	6	6.39	6.39	6.39	6.39	6.39	6.39	6.39	5.86
	7	6.54	6.54	6.54	6.54	6.54	6.54	6.54	6.54

Table E-11: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 2 \cdot 10^{-3}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.04	5.04	5.03	5.07	5.05	5.04	5.03	5.05
	2	5.07	5.07	5.07	4.06	4.06	3.80	3.80	3.69
	3	5.11	5.11	5.11	5.11	5.12	4.64	4.64	4.36
	4	4.76	4.76	4.76	4.76	4.76	4.42	4.43	4.26
	5	4.92	4.92	4.92	4.92	4.92	4.92	4.93	4.66
	6	5.08	5.08	5.08	5.08	5.08	5.08	5.08	4.87
	7	5.35	5.35	5.35	5.35	5.35	5.35	5.35	5.35

Table E-12: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 3 \cdot 10^{-3}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.11	5.11	5.07	5.10	5.06	5.14	5.11	5.19
	2	4.13	4.13	4.13	3.71	3.72	3.62	3.63	3.59
	3	4.51	4.51	4.51	4.51	4.52	4.06	4.07	3.81
	4	4.33	4.33	4.33	4.33	4.33	4.10	4.10	3.94
	5	4.38	4.38	4.38	4.38	4.38	4.38	4.38	4.14
	6	4.67	4.67	4.66	4.66	4.66	4.66	4.66	4.45
	7	4.81	4.81	4.81	4.81	4.81	4.81	4.81	4.81

Table E-13: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 4 \cdot 10^{-3}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.13	5.13	5.07	5.20	5.15	5.28	5.24	5.35
	2	3.82	3.82	3.82	3.61	3.62	3.57	3.57	3.55
	3	4.12	4.12	4.12	4.12	4.13	3.72	3.72	3.76
	4	4.13	4.13	4.12	4.12	4.12	3.85	3.86	3.66
	5	4.05	4.05	4.04	4.04	4.04	4.04	4.05	3.90
	6	4.36	4.36	4.35	4.35	4.35	4.35	4.35	4.13
	7	4.53	4.53	4.52	4.52	4.52	4.52	4.52	4.52

Table E-14: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 5 \cdot 10^{-3}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.20	5.20	5.11	5.31	5.25	5.41	5.36	5.49
	2	3.69	3.69	3.68	3.56	3.57	3.54	3.55	3.54
	3	3.82	3.82	3.82	3.82	3.83	3.79	3.79	4.78
	4	3.96	3.96	3.95	3.95	3.95	3.63	3.64	3.42
	5	3.95	3.95	3.94	3.94	3.94	3.94	3.95	3.89
	6	4.11	4.11	4.10	4.10	4.10	4.10	4.10	3.93
	7	4.41	4.40	4.39	4.39	4.39	4.39	4.39	4.39

Table E-15: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 6 \cdot 10^{-3}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.28	5.28	5.18	5.42	5.35	5.53	5.48	5.60
	2	3.62	3.62	3.62	3.54	3.55	3.53	3.53	3.52
	3	3.67	3.67	3.66	3.66	3.67	4.60	4.60	7.65
	4	3.80	3.80	3.79	3.79	3.79	3.43	3.44	3.29
	5	4.11	4.10	4.09	4.09	4.09	4.09	4.10	3.82
	6	3.96	3.96	3.93	3.94	3.94	3.94	3.94	3.86
	7	4.29	4.29	4.26	4.27	4.27	4.27	4.27	4.27

Table E-16: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 7 \cdot 10^{-3}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.36	5.37	5.25	5.52	5.45	5.62	5.57	5.68
	2	3.59	3.59	3.58	3.52	3.53	3.52	3.52	3.51
	3	3.76	3.76	3.75	3.75	3.75	6.58	6.58	13.3
	4	3.65	3.64	3.63	3.63	3.63	3.30	3.32	3.34
	5	4.39	4.39	4.37	4.37	4.37	4.37	4.39	3.61
	6	3.93	3.93	3.89	3.90	3.91	3.91	3.91	3.90
	7	4.38	4.38	4.34	4.35	4.36	4.36	4.36	4.36

Table E-17: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 8 \cdot 10^{-3}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.44	5.45	5.32	5.60	5.53	5.69	5.65	5.75
	2	3.57	3.57	3.56	3.51	3.52	3.51	3.51	3.50
	3	4.21	4.21	4.19	4.20	4.19	10.2	10.2	22.9
	4	3.51	3.51	3.48	3.48	3.49	3.27	3.30	3.69
	5	4.59	4.58	4.55	4.55	4.56	4.56	4.60	4.38
	6	4.03	4.03	3.98	3.98	4.01	4.01	4.01	3.92
	7	5.88	5.88	5.83	5.83	5.86	5.86	5.87	5.87

Table E-18: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 9 \cdot 10^{-3}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.51	5.52	5.39	5.66	5.60	5.75	5.71	5.80
	2	3.56	3.55	3.54	3.50	3.51	3.50	3.50	3.50
	3	5.17	5.16	5.15	5.15	5.14	16.2	16.1	37.8
	4	3.40	3.39	3.36	3.36	3.37	3.41	3.46	4.46
	5	4.50	4.50	4.46	4.46	4.47	4.47	4.55	10.7
	6	4.23	4.23	4.16	4.17	4.22	4.22	4.22	3.83
	7	11.5	11.5	11.4	11.4	11.5	11.5	11.5	11.5

Table E-19: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 1 \cdot 10^{-2}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.57	5.58	5.46	5.72	5.66	5.79	5.76	5.84
	2	3.55	3.55	3.53	3.50	3.50	3.49	3.50	3.50
	3	6.79	6.79	6.77	6.77	6.75	25.0	24.9	59.4
	4	3.32	3.32	3.28	3.28	3.30	3.76	3.85	5.79
	5	4.24	4.23	4.18	4.18	4.20	4.20	4.33	34.0
	6	4.48	4.47	4.39	4.39	4.48	4.48	4.48	3.63
	7	25.2	25.2	25.1	25.1	25.2	25.2	25.2	25.2

Table E-20: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 1.1 \cdot 10^{-2}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.63	5.63	5.52	5.76	5.71	5.83	5.80	5.87
	2	3.54	3.54	3.52	3.49	3.50	3.49	3.50	3.50
	3	9.27	9.27	9.24	9.24	9.21	37.5	37.4	89.4
	4	3.31	3.30	3.25	3.25	3.28	4.39	4.57	7.84
	5	4.68	4.67	4.59	4.60	4.64	4.64	4.80	98.4
	6	4.67	4.66	4.56	4.57	4.71	4.71	4.70	3.67
	7	50.2	50.2	50.1	50.1	50.3	50.3	50.3	50.3

Table E-21: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 1.2 \cdot 10^{-2}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.67	5.68	5.57	5.80	5.75	5.86	5.83	5.89
	2	3.54	3.54	3.51	3.49	3.49	3.49	3.51	3.50
	3	12.8	12.8	12.8	12.8	12.7	54.4	54.2	130
	4	3.37	3.36	3.30	3.30	3.34	5.39	5.70	10.8
	5	8.22	8.21	8.11	8.12	8.19	8.19	8.28	249
	6	4.73	4.71	4.60	4.60	4.83	4.82	4.79	5.05
	7	86.1	86.1	86.0	86.0	86.3	86.2	86.2	86.2

Table E-22: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 1.3 \cdot 10^{-2}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.71	5.72	5.62	5.83	5.79	5.88	5.86	5.91
	2	3.54	3.54	3.50	3.48	3.49	3.49	3.52	3.51
	3	17.6	17.6	17.6	17.6	17.5	76.7	76.4	182
	4	3.52	3.51	3.43	3.43	3.50	6.83	7.36	14.9
	5	19.9	19.9	19.7	19.8	19.9	19.9	19.5	562
	6	4.59	4.57	4.44	4.45	4.78	4.77	4.68	10.1
	7	124	124	124	124	125	124	124	124

Table E-23: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 1.4 \cdot 10^{-2}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.70	5.70	5.70	5.80	5.80	5.90	5.90	5.90
	2	3.50	3.50	3.50	3.50	3.50	3.50	3.50	3.50
	3	24.0	24.0	23.9	23.9	23.8	105	105	248
	4	3.80	3.80	3.70	3.70	3.80	8.80	9.70	20.4
	5	48.7	48.7	48.6	48.6	48.8	48.8	47.2	$1.16 \cdot 10^3$
	6	4.30	4.30	4.10	4.10	4.60	4.60	4.40	23.1
	7	148	147	147	147	148	148	148	148

Table E-24: SSE of DEM filter for $p, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ and $s = 1.5 \cdot 10^{-2}$ s.

		d							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p	**0**	134	134	134	134	134	134	134	134
	1	5.80	5.80	5.70	5.90	5.80	5.90	5.90	5.90
	2	3.50	3.50	3.50	3.50	3.50	3.50	3.60	3.50
	3	32.1	32.1	32.1	32.0	31.8	141	140	332
	4	4.20	4.20	4.10	4.10	4.20	11.4	12.9	27.7
	5	110	110	110	110	110	110	106	$2.24 \cdot 10^3$
	6	4.00	4.00	3.80	3.90	4.60	4.50	4.10	51.4
	7	139	139	139	139	139	139	139	139

Appendix F

Software

This appendix provides the references to the different software packages used in this thesis. Section F-1 gives some information about the ROS package developed in thesis. This ROS package is used to fly the AR.Drone 2.0 in simulation and in the NERDlab. Section F-2 provides the source and installation instructions of the MATLAB/Simulink toolbox used to conduct the system identification experiments.

F-1 AR.Drone 2.0 flight and simulation analysis

The ROS package constructed in this thesis is given in [80]. This package contains the ROS *launch* file to fly the AR.Drone 2.0 in simulation and physically in the lab, as well as the MATLAB files used to analyze the experiment and evaluate the filter results. The installation instructions for this package are given in the *README* file of the repository.

F-2 AR.Drone 2.0 support from MATLAB Embedded Coder

The software used to conduct the experiments is given in [40] and runs on the Windows operating system. A few more steps than mentioned on the GitHub page are required to install this package properly. Therefore, the complete installation procedure is listed below:

1. Download and install MATLAB R2016b or newer for Windows. The software is tested using the following installed MATLAB packages:
    - Aerospace Blockset 3.18
    - Aerospace Toolbox 2.18
    - Computer Vision System Toolbox 7.2
    - Control System Toolbox 10.1

- Embedded Coder 6.11
- Image Processing Toolbox 9.5
- Instrument Control Toolbox 3.10
- MATLAB 9.1
- MATLAB Coder 3.2
- Robotics System Toolbox 1.3
- Simulink 3D animation 7.6
- Simulink 8.8
- Simulink Coder 8.11
- Stateflow 8.8
- System Identification Toolbox 9.5

2. Install a C compiler. Run the command `mex -setup c` in the MATLAB Command Window to check if the computer has a C compiler installed. If not, the free MinGW64 compiler can be installed from the MATLAB Add-Ons Explorer.

3. Download and install the Code Sourcery ARM compiler. This is a free compiler which can be downloaded using the following link: https://sourcery.mentor.com/sgpp/lite/arm/portal/package8738/public/arm-none-linux-gnueabi/arm-2011.03-41-arm-none-linux-gnueabi.exe.

4. In Windows 8 or newer the installer needs to be run in Windows 7 compatibility mode (right-click on .exe, go to Properties, go to Compatibility tab, check the Run this program in compatibility mode box and select Windows 7 in the drop-down menu).

In case the FTP connection with the AR.Drone 2.0 gives trouble, try the following steps:

1. Download the Passive Mode FTP in MATLAB package using the following link: https://nl.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/6626/versions/2/download/zip).

2. Follow the README instructions included in this zip file.

3. Ensure that the computer is only connected to the AR.Drone 2.0 network.

Bibliography

[1] E. Guizzo, "What is a robot?." robots.ieee.org. https://robots.ieee.org/learn/what-is-a-robot/ (accessed Nov. 14, 2020).

[2] K. J. Friston, J. Kilner, and L. Harrison, "A free energy principle for the brain," *Journal of Physiology-Paris*, vol. 100, no. 1-3, pp. 70–87, 2006.

[3] K. J. Friston, "The free-energy principle: a unified brain theory?," *Nature reviews neuroscience*, vol. 11, no. 2, p. 127, 2010.

[4] K. J. Friston, N. Trujillo-Barreto, and J. Daunizeau, "Dem: A variational treatment of dynamic systems," *Neuroimage*, vol. 41, no. 3, pp. 849–885, 2008.

[5] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[6] A. A. Meera and M. Wisse, "Free energy principle based state and input observer design for linear systems with colored noise," in *2020 American Control Conference (ACC)*, pp. 5052–5058, IEEE, 2020.

[7] H. Von Helmholtz, *Handbuch der physiologischen Optik*, vol. 9. Voss, 1867.

[8] S. A. Kauffman, *The origins of order: Self-organization and selection in evolution*. OUP USA, 1993.

[9] K. J. Friston, J. Daunizeau, J. Kilner, and S. J. Kiebel, "Action and behavior: a free-energy formulation," *Biological cybernetics*, vol. 102, no. 3, pp. 227–260, 2010.

[10] W. Prinz, "Perception and action planning," *European journal of cognitive psychology*, vol. 9, no. 2, pp. 129–154, 1997.

[11] D. C. Knill and A. Pouget, "The bayesian brain: the role of uncertainty in neural coding and computation," *TRENDS in Neurosciences*, vol. 27, no. 12, pp. 712–719, 2004.

[12] K. J. Friston, "A theory of cortical responses," *Philosophical transactions of the Royal Society B: Biological sciences*, vol. 360, no. 1456, pp. 815–836, 2005.

[13] M. J. Beal and Z. Ghahramani, "The variational bayesian em algorithm for incomplete data: with application to scoring graphical model structures," *Bayesian statistics*, vol. 7, pp. 453–464, 2003.

[14] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel, "The helmholtz machine," *Neural computation*, vol. 7, no. 5, pp. 889–904, 1995.

[15] C. W. Fox and S. J. Roberts, "A tutorial on variational bayesian inference," *Artificial intelligence review*, vol. 38, no. 2, pp. 85–95, 2012.

[16] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[17] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[18] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[19] K. J. Friston, "Learning and inference in the brain," *Neural Networks*, vol. 16, no. 9, pp. 1325–1352, 2003.

[20] C. L. Buckley, C. S. Kim, S. McGregor, and A. K. Seth, "The free energy principle for action and perception: A mathematical review," *Journal of Mathematical Psychology*, vol. 81, pp. 55–79, 2017.

[21] K. J. Friston, "Variational filtering," *NeuroImage*, vol. 41, no. 3, pp. 747–766, 2008.

[22] K. J. Friston, J. Mattout, N. Trujillo-Barreto, J. Ashburner, and W. Penny, "Variational free energy and the laplace approximation," *Neuroimage*, vol. 34, no. 1, pp. 220–234, 2007.

[23] I. L. Hijne, "Generalised Motions in Active Inference by finite differences: Active Inference in Robotics," Master's thesis, Delft University of Technology, 2020.

[24] M. Wisse, "Derivation of generalised covariance matrix." unpublished, 2019.

[25] T. Foote and M. Purvis, "Rep 103 – standard units of measure and coordinate conventions." ros.org. https://www.ros.org/reps/rep-0103.html (accessed Sep. 19, 2020).

[26] Fotoaparatas, "Dronas parrot ar.drone 2.0 elite edition snow." fotoaparatas.lt. https://www.fotoaparatas.lt/dronas-parrot-ar-drone-2-0-elite-edition-snow (accessed Sep. 19, 2020).

[27] J. M. B. Domingues, "Quadrotor prototype," *MSc Thesis*, 2009.

[28] A. Tayebi and S. McGilvray, "Attitude stabilization of a vtol quadrotor aircraft," *IEEE Transactions on control systems technology*, vol. 14, no. 3, pp. 562–571, 2006.

[29] N. A. Chaturvedi, A. K. Sanyal, and N. H. McClamroch, "Rigid-body attitude control," *IEEE control systems magazine*, vol. 31, no. 3, pp. 30–51, 2011.

[30] J. Stuelpnagel, "On the parametrization of the three-dimensional rotation group," *SIAM review*, vol. 6, no. 4, pp. 422–430, 1964.

[31] R. W. Beard, "Quadrotor dynamics and control," *Brigham Young University*, vol. 19, no. 3, pp. 46–56, 2008.

[32] X. Zhang, X. Li, K. Wang, and Y. Lu, "A survey of modelling and identification of quadrotor robot," in *Abstract and Applied Analysis*, vol. 2014, Hindawi, 2014.

[33] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2451–2456, IEEE, 2004.

[34] Q. Li, "Grey-box system identification of a quadrotor unmanned aerial vehicle," Master's thesis, Delft University of Technology, Delft, 2014.

[35] M. Elsamanty, A. Khalifa, M. Fanni, A. Ramadan, and A. Abo-Ismail, "Methodology for identifying quadrotor parameters, attitude estimation and control," in *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1343–1348, IEEE, 2013.

[36] R. Mahony, R. W. Beard, and V. Kumar, *Modeling and Control of Aerial Robots*, pp. 1307–1334. Cham: Springer International Publishing, 2016.

[37] A. R. Partovi, A. Zong Yao Kevin, H. Lin, B. Chen, and G. Cai, "Development of a cross style quadrotor," in *AIAA Guidance, Navigation, and Control Conference*, p. 4780, 2012.

[38] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," tech. rep., Epfl, 2007.

[39] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5, pp. 4393–4398, IEEE, 2004.

[40] AR Drone 2.0 MATLAB/Simulink Target. (2017), Daren Lee. Accessed: Jul. 4, 2020. [Online]. Available: https://github.com/darenlee/SimulinkARDroneTarget/releases/tag/R2016_ab.

[41] P. W. Ledzian, "Network decentralized collision avoidance with applications in a scalable unmanned aerial system testbed," Master's thesis, Delft University of Technology, Delft, 2019.

[42] S. Piskorski, N. Brulez, P. Eline, and F. D'Haeyer, "Ar.drone developer guide." jpchanson.github.io. https://jpchanson.github.io/ARdrone/ParrotDevGuide.pdf (accessed Feb. 26, 2020).

[43] OptiTrack, "Prime 17w." optitrack.com. https://www.optitrack.com/cameras/prime-17w/ (accessed Dec. 1, 2020).

[44] John3, "computer icon vector png - desktop computer png image with transparent background." toppng.com. https://toppng.com/computer-icon-vector-png-desktop-computer-PNG-free-PNG-Images_218949 (accessed Dec. 1, 2020).

[45] OptiTrack, "Motive 2.2.0." optitrack.com. https://www.optitrack.com/motivenews.html (accessed Dec. 1, 2020).

[46] N. Elliott, "hp laptop icon png png image with transparent background." toppng.com. https://toppng.com/hp-laptop-icon-png-PNG-free-PNG-Images_119633 (accessed Dec. 1, 2020).

[47] W. commons, "File:ros logo.svg." commons.wikimedia.org. https://commons.wikimedia.org/wiki/File:Ros_logo.svg (accessed Dec. 1, 2020).

[48] W. commons, "File:matlab logo.png." commons.wikimedia.org. https://commons.wikimedia.org/wiki/File:Matlab_Logo.png (accessed Dec. 1, 2020).

[49] John3, "wi-fi router icon - router ico png image with transparent background." toppng.com. https://toppng.com/wi-fi-router-icon-router-ico-PNG-free-PNG-Images_250121 (accessed Dec. 1, 2020).

[50] P.-J. Bristeau, F. Callou, D. Vissiere, and N. Petit, "The navigation and control technology inside the ar. drone micro uav," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 1477–1484, 2011.

[51] A. Hernandez, C. Copot, R. De Keyser, T. Vlas, and I. Nascu, "Identification and path following control of an ar. drone quadrotor," in *2013 17th international conference on system theory, control and computing (ICSTCC)*, pp. 583–588, IEEE, 2013.

[52] M. Monajjemi, "ardrone_autonomy." wiki.ros.org. http://wiki.ros.org/ardrone_autonomy (accessed Dec. 1, 2020).

[53] J. Engel, "tum_ardrone." wiki.ros.org. http://wiki.ros.org/tum_ardrone (accessed Dec. 1, 2020).

[54] M. Monajjemi, "Docs ardrone_autonomy." ardrone-autonomy.readthedocs.io. https://ardrone-autonomy.readthedocs.io/en/latest/index.html (accessed Dec. 1, 2020).

[55] OptiTrack, "Optitrack prime 17w tech specs." optitrack.com. https://optitrack.com/cameras/prime-17w/specs.html (accessed Dec. 1, 2020).

[56] OptiTrack, "Motive:tracker." optitrack.com. https://optitrack.com/software/motive/tracker/ (accessed Dec. 1, 2020).

[57] K. Gräve and A. Bencz, "mocap_optitrack." wiki.ros.org. http://wiki.ros.org/mocap_optitrack (accessed Dec. 1, 2020).

[58] R. community, "About ros." ros.org. https://www.ros.org/about-ros/ (accessed Dec. 1, 2020).

[59] H. Machines, "Hbm 760 mm professional fan mobile, air displacement 10,200 m3 / h." hbm-machines.com. https://www.hbm-machines.com/producten/werkplaatsinrichting/elektrische-ventilatoren-an-aircos/hbm-760-mm-professionele-ventilator-verrijdbaar-luchtverplaatsing-10200-m3h (accessed Dec. 1, 2020).

[60] M. Y. Amir and V. Abbass, "Modeling of quadrotor helicopter dynamics," in *2008 International Conference on Smart Manufacturing Application*, pp. 100–105, IEEE, 2008.

[61] M. Bangura, R. Mahony, *et al.*, "Nonlinear dynamic modeling for high performance control of a quadrotor," in *Australian Robotics and Automation Association*, 2012.

[62] M. J. Mohammed, M. T. Rashid, and A. A. Ali, "Design optimal pid controller for quad rotor system," *International Journal of Computer Applications*, vol. 106, no. 3, pp. 15–20, 2014.

[63] M. Jardin and E. Mueller, "Optimized measurements of uav mass moment of inertia with a bifilar pendulum," *Journal of Aircraft - J AIRCRAFT*, vol. 46, pp. 763–775, 05 2009.

[64] N. Jeurgens, "Identification and control implementation of an ar. drone 2.0," Master's thesis, Eindhoven University of Technology, Eindhoven, 2017.

[65] Y. Sun, "Modeling, identification and control of a quad-rotor drone using low-resolution sensing," Master's thesis, Delft University of Technology, Delft, 2012.

[66] G. Perozzi, "A toolbox for quadrotors: from aerodynamic science to control theory," 2018. hal01696344.

[67] P. Abbeel, A. Coates, M. Montemerlo, A. Y. Ng, and S. Thrun, "Discriminative training of kalman filters.," in *Robotics: Science and systems*, vol. 2, p. 1, 2005.

[68] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, 2010.

[69] B. J. Odelson, M. R. Rajamani, and J. B. Rawlings, "A new autocovariance least-squares method for estimating noise covariances," *Automatica*, vol. 42, no. 2, pp. 303–308, 2006.

[70] S. Bolognani, L. Tubiana, and M. Zigliotto, "Extended kalman filter tuning in sensorless pmsm drives," *IEEE Transactions on Industry Applications*, vol. 39, no. 6, pp. 1741–1747, 2003.

[71] J. Wayne, "Helicopter theory," *Princeton University*, 1980.

[72] S. Zhao, "Time derivative of rotation matrices: A tutorial," *arXiv preprint arXiv:1609.06088*, 2016.

[73] D. H. Salunkhe, S. Sharma, S. A. Topno, C. Darapaneni, A. Kankane, and S. V. Shah, "Design, trajectory generation and control of quadrotor research platform," in *2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*, pp. 1–7, IEEE, 2016.

[74] M. Krznar, D. Kotarski, P. Piljek, and D. Pavković, "On-line inertia measurement of unmanned aerial vehicles using on board sensors and bifilar pendulum," *Interdisciplinary Description of Complex Systems: INDECS*, vol. 16, no. 1, pp. 149–161, 2018.

[75] M.-M. GmbH, "Kd24s ±100n." me-systeme.de. https://www.me-systeme.de/product-pdf?product_id=1723&lang=en (accessed Aug. 31, 2020).

[76] I. FUTEK Advanced Sensor Technology, "Jr. miniature s-beam load cell." media.futek.com. https://media.futek.com/content/futek/files/pdf/productdrawings/fsh02664.pdf (accessed Sep. 2, 2020).

[77] H. Huang and J. Sturm, "tum_simulator." wiki.ros.org. http://wiki.ros.org/tum_simulator (accessed Dec. 1, 2020).

[78] F. Furrer, M. Burri, M. Kamel, J. Nikolic, and M. Achtelik, "rotors_simulator." wiki.ros.org. http://wiki.ros.org/rotors_simulator (accessed Dec. 1, 2020).

[79] H. Zhu, "AMR Lab Tutorial." unpublished, 2019.

[80] ardrone2_dem_filter. (2020), Dennis Benders. Accessed: Dec. 9, 2020. [Online]. Available: https://github.com/dbenders1/ardrone2_dem_filter.