Evaluating motion capture as a means of system identification of a quadcopter

Martin Ottenklev

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6058
ISSN 0280-5316

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

Printed in Sweden by Tryckeriet i E-huset
Lund 2018

Abstract

This thesis describes the method of identifying unknown parameters that affect the dynamics of a given quadcopter, also known as grey box system identification. This was primarily done utilising an inertial measurement unit and a motion capture camera system. A system of equations describes the dynamics of the quadcopter, and was later coupled with data gathered while flying, in order to use different methods of system identification. As quadcopters are unstable, the first task was to design a stabilising regulator, making stable flight possible, and thus gathering flight data.

A few parameters regarding motor dynamics were evaluated via simple experiments with tools including a tachometer, a scale and a microphone.
When it comes to flight dynamics, the first method of identification was to use a prediction error method which, given data regarding input signals, output signals and a mathematical model, tries to evaluate unknown parameters by minimising the error between state measurements and estimated states based on the earlier mentioned model, in each timestep. This method proved to be unsuccessful, for reasons partly unknown, and was later changed for a method utilising an extended Kalman filter, which gave more reliable results. Possible explanations to this phenomena may include that the Kalman filter implemented beforehand in the camera system may need to be retuned and that the aforementioned mathematical model needs to be reevaluated.
Estimated parameter values works well with the model, but that is not so say that there is not room for improvement.

Acknowledgements

First I would like to thank Erik Ekelund for his enthusiasm, never-ending patience, words of “encouragement” and his actual words of encouragement. I am of course also grateful for the support of my supervisor at LTH, Anders Robertsson and my supervisor at SAAB Dynamics, Kristoffer Bergman.

Everyone I’ve had the pleasure of meeting during my time at SAAB to make this a memorable experience, and Torbjörn Crona for giving me the opportunity to carry out this project.

Last but not least I would like to thank Jonas Carlsson, without whom I would not have gotten a foot in at SAAB in the first place.

Contents

1

Introduction

This thesis details the work regarding system identification and control implementation on a quadcopter platform provided by SAAB Dynamics. The following chapter aims to provide an overview of the project.

1.1 Background

Unmanned aerial vehicles (UAV's) have rapidly increased in popularity over the last years, both in civil use as well as in various military operations. Different configurations provide a wide range of specifications that may be used for operations in dangerous environments, search and rescue missions and inspection of places that might be difficult to reach. The type of UAV to be examined was a so called multicopter with four rotors (also known as a quadcopter). A similar platform has been subject to two previous master's theses, [Kugelberg, 2016] and [Reizenstein, 2017]. Slight configurations has been made since then in order to increase the performance in terms of motor power, range and battery time.

1.2 Problem formulation

In order to design high performing controllers, it is of utmost importance to be aware of how the system behaves, which is what system identification aims to achieve. As the versatility of UAV's has gotten more attention, a lot of research has been conducted in the field, such as [Landry, 2015] and [Förster, 2015].
Quadcopters lack any form of natural stabilising element. This combined with the fact that they are underactuated, meaning in this case that they have more degrees of freedom (DOF) than control inputs (6 DOF's, 4 control inputs), results in good software control algorithms being a necessity in order to control them.

1.3 Objectives

The most prominent task of this thesis was to perform system identification on the quadcopter by different means, mostly by using a motion capture camera system. Pre-existing PID-controllers were to be retuned such that they perform well not only close to hovering, but over a wider range of flight behaviour.

1.4 Related work

Previous work has been focused on estimating the platform dynamics close to hover in order to implement angle controller, [Kugelberg, 2016] and position and trajectory control using PID and LQ regulation, [Reizenstein, 2017].

1.5 Limitations

Hardware and most of the software was provided for by SAAB Dynamics. As no time has been spent working on hardware, reasoning for different hardware choices will not be discussed in depth. Experiments were conducted by a human operator, imposing limitations on the arbitrariness of the input.
It was assumed that the quadcopter dynamics are highly non-linear. As quadcopters are unstable by default, it was not possible to control it using open-loop control, that is to say without some form of regulator.

1.6 Thesis structure

This thesis is divided into 7 chapters. Chapter 2 describes the platform of use, how it is modelled in a mathematical sense, software, hardware and different flight modes to be configured. Thereafter follows Chapter 3, which briefly discusses a little bit more about the software and the program structure used for operating the quadcopter and gathering flight data.
Chapter 4 contains information about system identification theory and what methods were used to perform it, along with its results. This is followed by Chapter 5, which provides an insight in control theory, its implementation and how to design it for operable flight of the platform.
Chapter 6 discusses some final conclusions regarding different experiments, their results and hardships, and how they might be overcome. Finally, Chapter 7 details possible future work that might be of interest on the quadcopter, both within the fields that this thesis has touched, and in entirely new ones.

2

Quadcopter platform

The quadcopter can be seen in Figure 2.1.

Figure 2.1 The quadcopter platform used in the thesis. The microcomputer can be found inside the green protective case. As the case was not equipped with any form of locking mechanism a bit of duct tape came in handy.

2.1 Hardware

Installed hardware is found in Table 2.1

Table 2.1 Hardware present in the quadcopter platform along with performance and purpose for each part.

BeagleBone Blue
The main computer performing most of the calculations was a BeagleBone Blue, which is a microcomputer with a 1 GHz ARM Cortex-A8 processor.
IMU
The IMU (Inertial Measurement Unit) was used for measuring acceleration, angles and angular velocities.
Satellite Receiver
OrangeRx R110x is a satellite receiver meant to extend the range between the quadcopter and the remote as well as increasing the link robustness.
Motors
Four Luminier FX2206-13 2000 KV brushless motors were used to provide thrust. The KV rating of a brushless motor is the so called motor velocity constant, measured in RPM per volt. 2000 KV means that the angular rate will equal 2000 RPM per supplied volt.
Propellers
Rotors used have a diameter of 5 inches ($\approx$ 12.7 centimetres), with a pitch of 3 inches ($\approx$ 7.6 centimetres), and 3 blades each. Generally, more blades, longer blades and larger pitch all results in more thrust, torque and turbulence [*Droneomega propeller guide*]. As flight dynamics depend on these factors the same type of propellers have been used throughout the entire thesis (although in some cases of varying colours).
ESC
Each motor is controlled and powered by an Electronic Speed Controller (ESC).

Propeller specifications

The diameter of a propeller is simply twice the radius, the length from a tip to the centre of the propeller construction, see Figure 2.2. Definition of pitch is a bit less intuitive. In theory it is a measure of how far a propeller will move through the air during a single rotation. In practice this depends on factors such as propeller material, air density and different measures of efficiency. A simple explanation is provided here [*Droneomega propeller guide*]. Looking at the upper part of Figure 2.3, the left screw has a coarser thread, i.e. higher pitch, than the right one. Upon screwing the screws into a piece of wood the one with the coarser threading would

sink more into the wood per revolution. Looking again at Figure 2.3 it is illustrated that the right hand air plane needs to maintain a higher RPM to travel as far as the left hand air plane during a fixed period of time.

Figure 2.2 Propeller with a 2.5 inch radius → 5 inch diameter

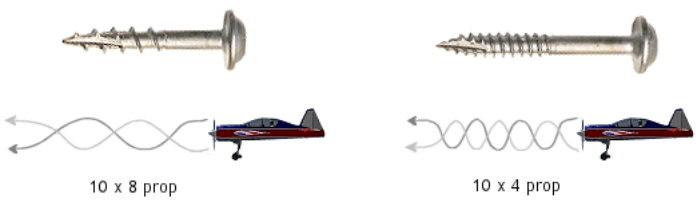


Figure 2.3 How propeller pitch affect flight dynamics. Figure source: [*How to choose propeller for a mini quad*].

2.2 Software

Installed software can be found in table 2.2

Table 2.2 Software present in the quadcopter.

Debian
The BeagleBone Blue runs Debian, which is a Unix-like open source operating system.
Robot Operating System (ROS)
ROS is a robotics middleware (a software toolbox for robot software development). It comes with a wide variety of useful tools for programming all types of robots. For a more thorough explanation of how ROS operates, see Chapter 3.
Robotics Cape
Robotics Cape is a toolbox originally developed for the BeagleBone Black, and it is also compatible with BeagleBone Blue (which is a more recent instalment of microcomputers in the BeagleBoard series). This toolbox adds upon ROS and provides tools for functionality of IMU, barometer, DC motor control, Servo, ESC control, DSM radio, GPIO and more.

2.3 Euler angles and body frame coordinates

In aviation it is common practice to use a so called NED (North-East-Down) coordinate system, where the X-axis points forward, the Y-axis points to the right and the Z-axes points downwards relative to the body frame. Rotations around the axes are referred to as roll, pitch and yaw respectively. Euler angles ϕ, θ and ψ may be used to relate the body frame coordinates to global ones using the rotation matrix

$$R = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - s_\psi c_\phi & s_\theta s_\psi + s_\theta c_\phi c_\psi \\ s_\psi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\theta s_\psi c_\phi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}. \tag{2.1}$$

Derivatives of the Euler angles can be expressed using angular velocities in the body fixed frame and a rotational matrix T

$$T = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix}. \tag{2.2}$$

In other words, R describes the relation between linear velocities in the body-fixed frame to the linear velocities in the inertial frame, whereas T describes the relation between angular velocities in the body-fixed frame and angular velocities in the inertial frame [Nilsson, 1998].

2.4 Quadcopter modelling

2.4.1 Non-linear model

The dynamics of a quadcopter may be modelled as

$$\dot{\eta} = J(\eta)v \tag{2.3}$$

$$\dot{\omega} = I^{-1}(-\omega \times I\omega + \tau) \tag{2.4}$$

, where

$$\eta = [\text{x y z } \phi \ \theta \ \psi]^T \tag{2.5}$$

denotes position and orientation in the inertial frame, whereas

$$v = [\text{u v w p q r}]^T \tag{2.6}$$

denotes linear and angular velocities in the body-fixed frame of the quadcopter. I is the moment of inertia matrix

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.7}$$

, ω are the angular velocities

$$\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.8}$$

, τ is the body moment

$$\tau = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{2.9}$$

and $J(\eta)$ is a matrix

$$J = \begin{bmatrix} R(\eta) & 0_{3\times3} \\ 0_{3\times3} & T(\eta) \end{bmatrix} \tag{2.10}$$

, where $0_{3\times3}$ is a zero matrix of size 3×3. η and v relate to each other using the rotational matrices such that

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R(\eta) \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.11}$$

and

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T(\eta) \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.12}$$

[Thornton and Marion, 2004].

2.4.2 Linear model

It is possible to linearise the quadcopter model around a hovering state, meaning that the roll and pitch angles, ϕ and θ, are close to zero. Linearising Equations 2.1 and 2.2 around this point gives

$$R_{linear} = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

and

$$T_{linear} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.14}$$

Resulting kinematic relations between linear and angular velocities of the body fixed frame with respect to the global frame is then described by

$$\dot{\eta}_{linear} = \begin{bmatrix} u \cdot c_\psi - v \cdot s_\psi \\ u \cdot s_\psi + v \cdot c_\psi \\ w \\ p \\ q \\ r \end{bmatrix} \tag{2.15}$$

, which may be compared to Equation 2.3.

2.4.3 Design implementations

A quadcopter has four motors with a rotor each. In order to avoid rotation in the yaw direction, due to propellers rotating in one direction causing a torque in the other direction, the setup is usually such that two motors along the diagonal rotate in the same direction, while neighbouring motors rotate in opposite directions. See Figure 2.4 for coordinate system and rotation of motors.

Control signals from the remote controller are sent as a dsm package to the BeagleBone. The signals $u_{throttle}$, u_{roll}, u_{pitch} and u_{yaw} are normalised such that

$$\begin{aligned} 0 \leq u_{throttle} \leq 1 \\ -1 \leq u_{roll} \leq 1 \\ -1 \leq u_{pitch} \leq 1 \\ -1 \leq u_{yaw} \leq 1. \end{aligned} \tag{2.16}$$

[Kugelberg, 2016] implemented the relation between motor and control signals as

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} u_{throttle} \\ u_{roll} \\ u_{pitch} \\ u_{yaw} \end{bmatrix}$$

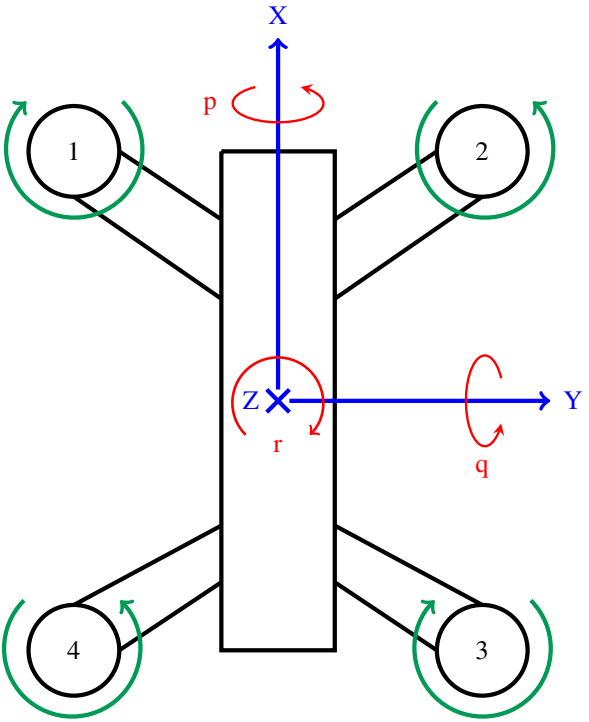


Figure 2.4 Rough sketch of the quadcopter platform that shows axes and rotational directions of the motors.

, and [Reizenstein, 2017] did the same. The configuration of the quadcopter was slightly different in this thesis. Motors were shifted one step counterclockwise and rotations were inverted. As a result, the relation between motor signals and control signals were

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_{throttle} \\ u_{roll} \\ u_{pitch} \\ u_{yaw} \end{bmatrix}. \tag{2.17}$$

2.5 Ground effect

When rotors are close to the ground they will push the air down into the ground, causing it to “bounce back”, generating an upwards force acting on the platform. This effect may be present up to a height of $H = 6\text{R}$, where R is the radius of a rotor

[Bernard et al., 2017]. As this effect is quite non-linear it is difficult to compensate for. Experiments executed in this thesis were mostly done on such a height that this effect was negligible.

2.6 Manual flight modes

There were two main modes of flying, angle mode and rate mode (popularly named self-level mode and acro mode respectively in the quadcopter community).

2.6.1 Angle mode

Angle mode was a flight mode in which the quadcopter aimed to self-level, using the accelerometer and the gyroscope in the IMU. As the name suggests, the operator was controlling the angular references using the control sticks on the remote controller. In yaw however it was the angular velocity (instead of the angle) that was to be controlled. This as it would feel unnatural to keep a control stick at a constant position if the pilot were to, for instance, try to align the platform with himself. If no input beside throttle was provided by the pilot, the quadcopter should try to keep a zero angle in roll and pitch and an angular velocity of zero around the yaw axis.

2.6.2 Rate mode

In rate mode the pilot control the angular velocities (the rates). The accelerometer was not used, but the gyroscope on the other hand was. Without input commands (other than throttle) the quadcopter should try to keep its current attitude. This way of flying is often used by more experienced pilots when performing more acrobatic manoeuvres. Just like the reasoning behind controlling yaw rate in angle mode (instead of yaw angle) it might be preferable if the quadcopter does not flip back after performing a loop, as soon as the pilot lets go of the control stick.

3

Programming

This chapter will discuss methods of programming, toolboxes and their implementations.

3.1 ROS

ROS is not an operating system per definition, but rather a robotics middleware built to make software development of robots easier. See table 3.1 for some basic ROS concepts.

3.2 Robotics cape

Developed by Strawson design, robotics cape builds upon ROS and adds a lot of useful features in order to simplify programming of various robots. It was designed with the beaglebone black in mind, but it is compatible also with the beaglebone blue, which was used in this thesis.

Table 3.1 ROS concepts and usage.

Nodes
Nodes are processes that perform computations such as calculating an optimal path with respect to time, aiming a laser pointer or performing localisation.
Master
ROS master provides name registration for nodes such that different nodes are able to find each other and exchange information.
Messages
Communication between nodes is done by passing messages. They support most standard primitive types such as ints, floats, booleans, strings etc.
Topics
Messages are routed via a transport system using publishers and subscribers. The topic is a name used to identify the content of the message.
Publishers
A publisher will publish a topic such that a subscriber may listen to it. There may be several publishers to a single topic and one publisher may publish multiple topics.
Subscribers
Subscribers listens to topics sent by publishers. Just as there may be several publisher per topic, there may be several subscribers listening to a single topic. A single subscriber may also listen to multiple topics.
Services
A service is composed of a request and a response. A providing node offers a service and a client node uses it by sending a request and waiting for a reply.
Bags
Bags are formats for storing and playback of ROS message data. Bagfiles can be opened and analysed in matlab.

[*ROS - concepts description*]

Programming languages that are fully supported by ROS are C++, Python and Lisp. Experimental libraries are available for a handful other languages such as Java, Lua, Ruby and R. In this thesis, C++ was used. ROS has a wide variety of tools for things like plotting data, making GUI's, data collection and playback of said data. See table 3.2 for some of the tools that were used in this thesis.

Table 3.2 ROS tools and usage.

rosrun
Rosrun is used to run an executable file in an arbitrary package without having to specify the full pathway to the file.
rosbag
In order to record data from a publisher, rosbag is used. Data is stored at a specified location in a bagfile. Recorded data may later be played back using the ros tool rqt_bag. Data may also be transferred to matlab where it is able to analyse it.
Launch Files
A launch file may specify multiple nodes that will be executed at the same time. Launch files are executed using the predefined command roslaunch.
rqt_gui
rqt_gui is a tool able of generating a GUI and plotting data. As a GUI it is able to change control parameters online.

3.3 Program structure

The main script includes predefined functions from both ROS and Robotics Cape as well as functions written by the author of the thesis. Using Robotics Cape it was possible to assign functionality such as switching regulator mode, switching flight mode and arming/disarming the quadcopter using switches on the remote controller. The main script was heavily interrupt based. Whenever a new signal was sent from the controller, an exception was caught, and new motor signals were calculated. See Figure 3.1 for a flowchart describing how the program operated in terms of computing new control signals and recording in-flight information.

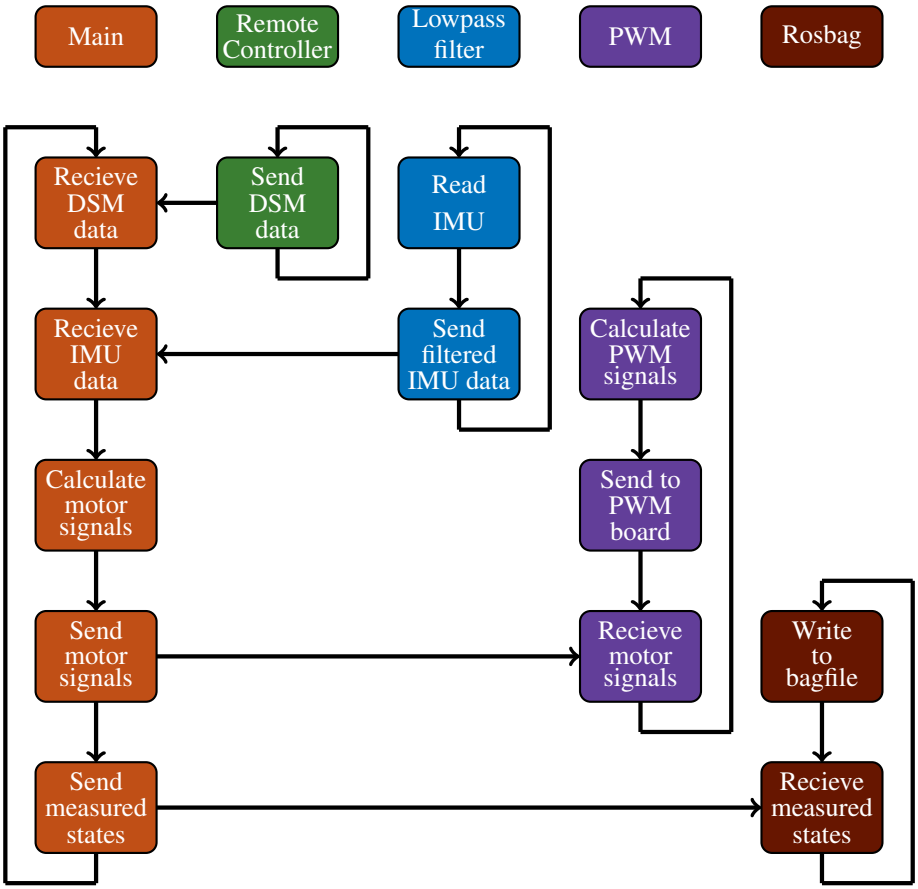


Figure 3.1 Flowchart of program architecture when using a remote controller.

4

Model estimation and parameter identification

System identification is the process of estimating a model of a given system based on observed input- and output-data over time. Two of the more common models in the field of system identification are black-box models and grey-box models. A grey-box model is based on some form of prior knowledge regarding physical modelling of the system, whereas a black-box model contains no such information. In this thesis, grey-box modelling will be covered, with the model described by Equations 2.3 and 2.4 as a starting point. This chapter will discuss some theory of system identification, chosen methods and some parameter identification.

4.1 Motor dynamics

The quadcopter was equipped with four brushless DC motors. Dynamics of a brushless DC motor may be modelled using a first order transfer function on the form

$$\Omega(s) = \frac{K}{sT_m + 1} e^{-Ls}\, U(s)\ [rad/s] \tag{4.1}$$

where K is the static gain, T_m is the time constant and L is the time delay[Fogelberg, 2013],[Chéron et al., 2010]. The time constant depends on factors such as motor and ESC, whereas time delay mainly occurs in the communication between the CPU and the ESC:s (usually a few milliseconds). It seemed reasonable to suspect that the difference in order of magnitude for the two was significant enough for the model to be approximated without the delay,

$$\Omega(s) = \frac{K}{sT_m + 1}\, U(s)\ [rad/s]. \tag{4.2}$$

4.1.1 Motor thrust

In order to make the motors spin, a PWM signal value was sent to the ESC:s. This number could take a value in the interval [0.0, 1.0] (for each motor) which was nor-

malised between PWM-lengths $1000\mu s - 2000\mu s$ using robotics capes functionality for PWM signals. The motor itself did not provide any way of measuring the rotational speed for a given PWM command. This could be solved by attaching a bit of reflective tape to a propeller, sending a PWM signal, and measuring the rotational velocity using a tachometer. The tachometer had a resolution of 0.1 RPM and was described as having a precision of $\pm 0.05\%$. In order to avoid the quadcopter flying away when doing this experiment, it was placed upside down. One could send the signal to only one of the motors and prevent it from taking off that way. It was however theorised that the angular velocity of a given propeller may change depending on how many motors were activated, and thus all four were turned on for this experiment. It it also worth pointing out that angular velocity may depend on how well charged the battery is, but that factor was left out for simplicity. As slight variations may occur between different motors, the experiment was repeated several times for each motor and an average was calculated. At 50 % of maximum motor power the thrust was such that the quadcopter was capable of rising very swiftly (had it not been placed upside down during this particular experiment) and the motors began emitting a high-pitched noise. Due to these factors, experiments where not done on higher rotational speeds both due to safety precautions and the fact that a higher RPM probably would not be necessary for the project. As can be seen in Figure 4.1, the angular velocity was quite linear as a function of normalised throttle within the working range. Polynomial fits of first and second order were calculated using a least-square fit method in matlab. The fits may be seen in figure 4.2. The two polynomial fits were defined as

$$\begin{aligned} \Omega(u_{th}) &= 2738.5 \cdot u_{th} \text{ [rad/s]} \\ \Omega(u_{th}) &= -443.07 \cdot u_{th}^2 + 2963.2 \cdot u_{th} \text{ [rad/s]} \end{aligned} \tag{4.3}$$

As may be seen in both in Equation 4.3 and in Figure 4.2, the difference between the polynomials is very small within the working range. It was thus preferred to use the first order polynomial in order to achieve a simpler model of the motor dynamics.

In order to evaluate how much thrust was generated at a given angular velocity the quad was placed upside down on a scale. The choice of putting it upside down was done to prevent the effect of ground effect. It could also have been possible to attach the quadcopter to a rack and having it placed at such a height that ground effect was of no concern. Multiple angular velocity commands were given and and the resulting mass differentiation was measured. The thrust was then calculated using $F = \Delta mg$. Results of the experiment is viewed in Figure 4.3. As all four propeller were mounted during the experiment, the result is an average of each single motor.

Thrust force as a function of angular velocity is often modelled as a second order polynomial [Bergman and Ekström, 2014],[Fogelberg, 2013]. As seen in Figure 4.4, that seemed to be a good estimation also with the platform used in this thesis. One may also consider the expression for centripetal force, $F = mr\omega^2$, meaning that the

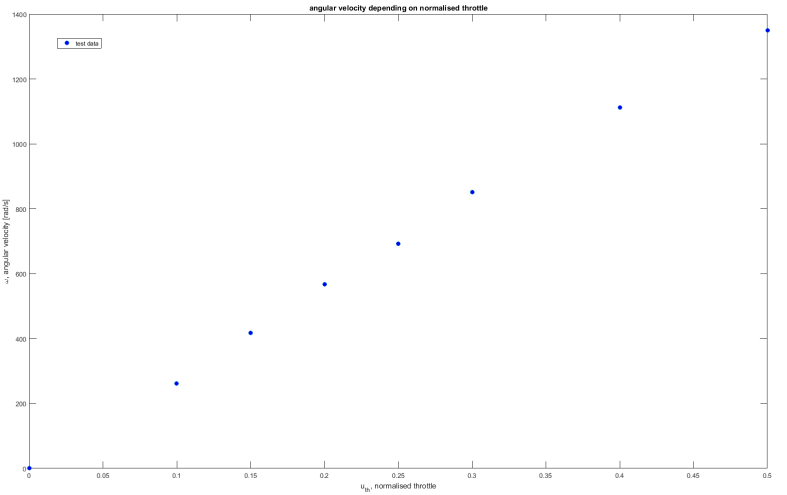


Figure 4.1 Angular velocity depending of normalised throttle. It is quite linear within the working range.

thrust force is proportional to the square of the angular velocity.

The second and third order polynomials are nearly indistinguishable in the working range. Thus it was preferred to choose to model the thrust after the second order polynomial in order to achieve a simpler model.

Finally, thrust force as a function of angular velocity could be written as stated in equation 4.4, where the second order polynomial was the one chosen throughout the thesis to model the motor dynamics.

$$\begin{aligned} &\text{Second order polynomial} \\ &Thrust(\Omega) = 8.78 \cdot 10^{-7}\Omega^2 - 8.89 \cdot 10^{-6}\Omega \ [N] \\ &\text{Third order polynomial} \\ &Thrust(\Omega) = 9.84 \cdot 10^{-11}\Omega^3 + 6.64 \cdot 10^{-7}\Omega^2 + 9.46 \cdot 10^{-5}\Omega \ [N] \end{aligned} \tag{4.4}$$

4.1.2 Time constant

The time constant describes how long it takes a system to change between two specified values. This is also referred to as rise time or fall time (depending on whether the signal is increasing or decreasing). Rise time is defined as the time required to for the response to rise from $x\%$ to $y\%$ of its final value [Levine, 2011], where x and y depends on the damping of the system. In most cases they correspond to 10% and 90% respectively [Orwiler, 1969]. It should have been possible to estimate a time constant using tachometer from the previous experiment regarding rotational speed

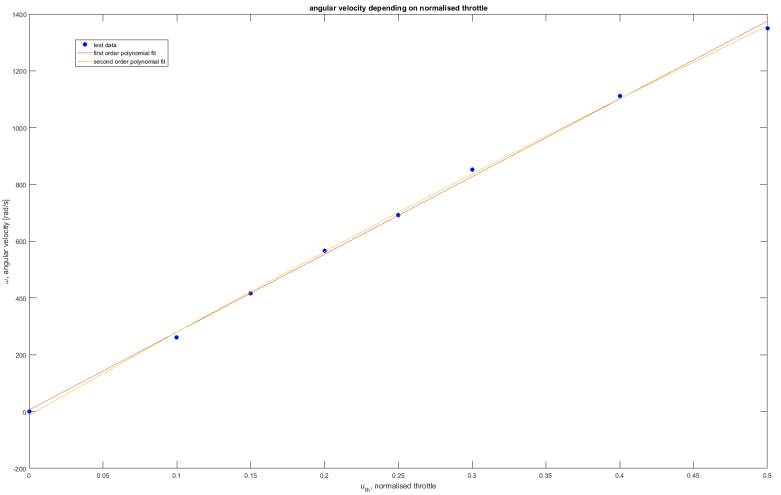


Figure 4.2 Angular velocity as a function of normalised throttle. A first and a second order fit is seen. The difference between the two is so small that it is deemed negligible.

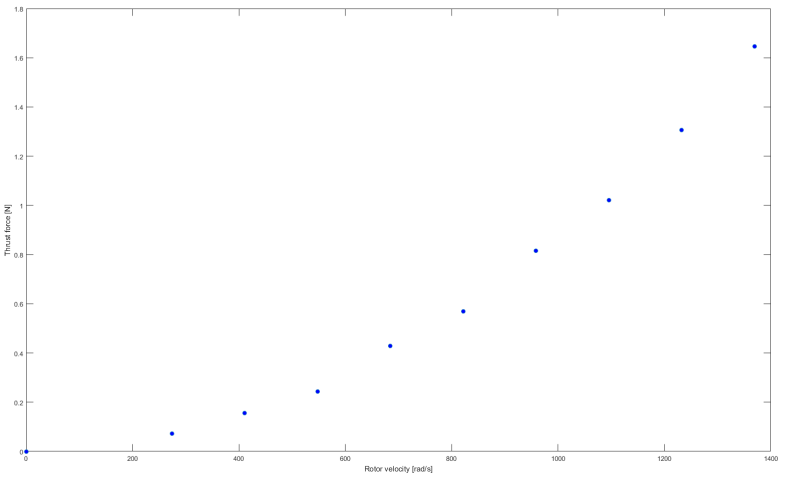


Figure 4.3 Thrust force depending on angular velocity of a motor.

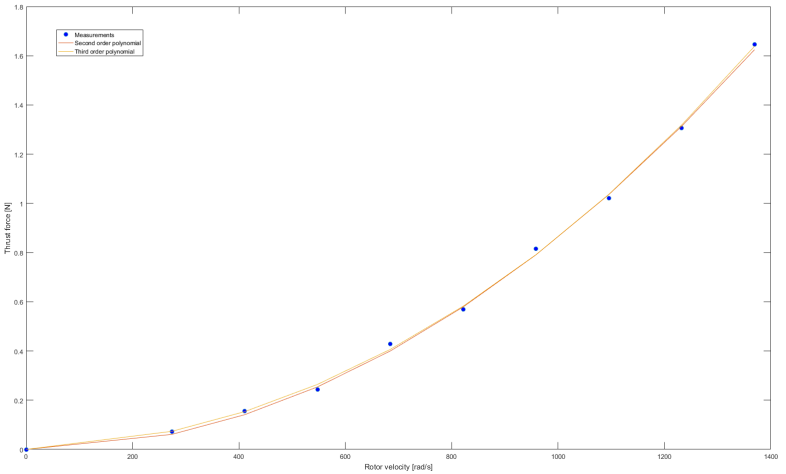


Figure 4.4 Thrust force depending on angular velocity for a single motor. Polynomial fits of second and third order are seen together with the test data.

depending on normalised throttle, by evaluating the time required to reach a new desired angular velocity. This was however not possible as the tachometer lacked logging functionality. Instead it was noticed that each angular velocity seemed to emit a distinct pitch. Thus it should be possible to use Fourier transformation and examine if there were certain dominating frequencies for an arbitrary control signal [Bergman and Ekström, 2014]. Differences in dominating frequency over time, during a step response may be seen in figure 4.5. It shows that the rise time is close to 0.1 seconds.

4.1.3 Motor torque

As motors spin in one direction, they will give rise to an opposing torque in the other direction. This is the basis on which yaw rotations depends. As per Figure 2.4, motors that are diagonally opposed spin in the same direction. Thus they give rise to a torque in the same direction (the direction opposite of their rotation). If, for instance, motors 1 and 3 in Figure 2.4 were to spin a little bit faster than motors 2 and 4, the quadcopter would rotate counterclockwise. This is also evident from looking at Equation 2.17.

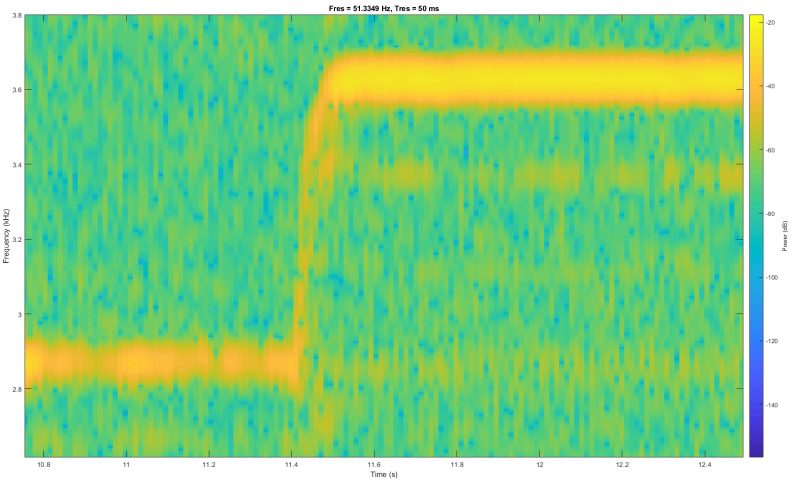


Figure 4.5 Frequency distribution over time when performing during a step response.

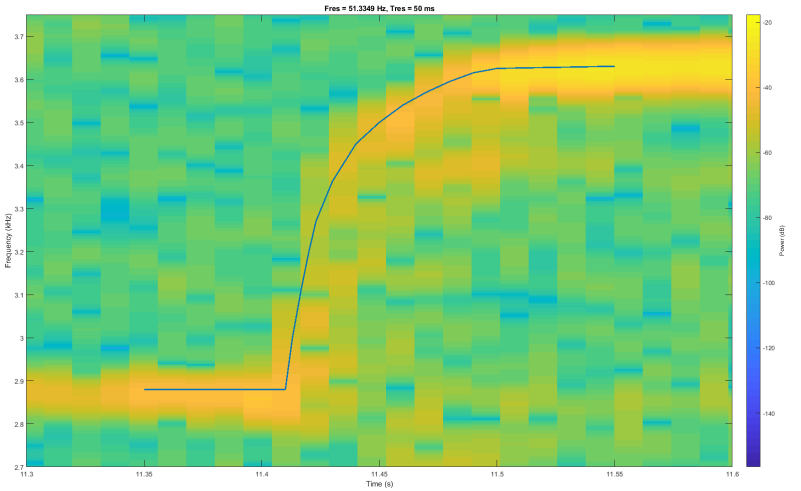


Figure 4.6 Frequency content plotted with a polynomial fit of the yellow curve.

4.2 Data collection

As mentioned in Chapter 3 there is a ROS tool named rosbag. A launchfile which starts one node for the main script and one node for rosbag was executed. Data was collected in the flying grounds indoors, using the IMU in the quadcopter and the available camera system. Information from the camera system was gathered by placing a few reflective balls on the platform, see figure 4.7, and starting a rosnode that connects to the cameras [*Camera system rosnode*]. The rosnode used an extended Kalman filter (see Section 4.4) to estimate the position, velocity, angles and angular rates of the quadcopter. Said filter was already implemented before the start of this thesis.

Figure 4.7 The quadcopter equipped with highly reflective silvery spheres, used by the camera system to detect objects.

In order to estimate a model of a given system, data sets containing input- and output-data is required. A data set consisting of N points can be denoted Z_N and defined as

$$Z_N = \{y_k, u_k, o_k\}_{k=1}^{N} \tag{4.5}$$

where y_k is the output, u_k is the input, and o_k contains all other signals that are measured. As it is not possible to make ROS sample with a constant sampling frequency, the data had to be resampled. This was done either with a linear or a spline interpolation, depending on which yielded the better result.

During test flights it was noted that the motion capture sometimes seemed to lose track of the quadcopter, which lead to strange measurements such as angular velocities up to a hundred revolutions per second, angles being constant for short periods of time, and the quadcopter being perfectly still at times. Figure 4.8 depicts an example of an unreasonably high angular rate. Due to this, it was determined to use the IMU as much possible, meaning that angles and angular rates were gathered from the IMU, whereas the global positions and body frame linear velocities were gathered using the motion capture system.

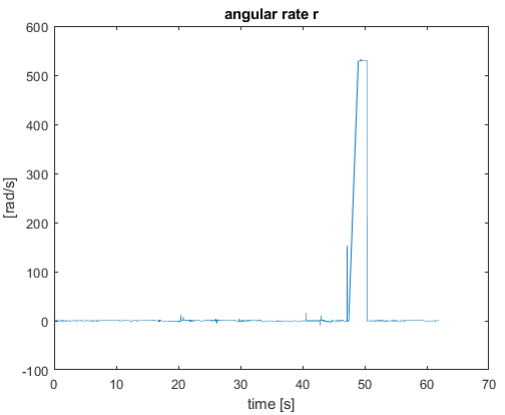


Figure 4.8 Example of when the camera measurement was clearly wrong, as it corresponds to the quadcopter making almost 100 revolutions per second for a while. Most likely it was a side effect of loosing track of the platform over a short period of time.

4.3 Prediction error method (PEM)

Prediction error method uses a predictor $\hat{y}_k(\theta)$ which in the non-linear case may be written as

$$\hat{y}_k = h(\hat{x}_k, u_k, \theta) \tag{4.6}$$

where $\hat{x}_k$ corresponds to the estimated state vector, u_k are the system inputs and $\hat{y}_k(\theta)$ is the predicted model output [Ljung, 1999]. Fitting is done by minimising a cost function $V_N(\theta, Z^N)$, where Z^N is a data set $Z^N = [y(1), u(1), y(2), u(2), ..., y(N), u(N)]$, such that the estimate $\hat{\theta}_N$ is defined by the minimisation of

$$\hat{\theta}_N(Z^N) = \arg\min V_N(\theta, Z^N). \tag{4.7}$$

4.3.1 Parameter estimation using PEM

In order to use PEM, initial states and initial parameter estimations had to be defined.

Acceleration components in the body fixed coordinate system could be modelled using the second order polynomial in equation 4.4 and some damping factor from air resistance as

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \frac{Thrust(\Omega)}{m} \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} - \frac{D_{linvel}}{m} \begin{bmatrix} |u|u \\ |v|v \\ |w|w \end{bmatrix} + R^{-1} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{4.8}$$

where D_{linvel} is a diagonal matrix

$$\begin{bmatrix} D_u & 0 & 0 \\ 0 & D_v & 0 \\ 0 & 0 & D_w \end{bmatrix} \tag{4.9}$$

, which describes damping factors from air resistance in the three dimensions, which in turn depends on the current velocity. g is the gravitational constant, m is the quadcopter mass and R is the rotational matrix seen in Equation 2.1. In the body fixed system, the thrust force was always pointed in negative z-axis direction, thus the vector $[0 \quad 0 \quad -1]^T$.

Studying Equation 4.8 and looking back at equation 2.4, it seemed reasonable to introduce some dampening terms on angular velocities, such that

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = I^{-1} \left(- \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \right) - D_{angvel} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{4.10}$$

, where D_{angvel} is a diagonal matrix

$$\begin{bmatrix} D_p & 0 & 0 \\ 0 & D_q & 0 \\ 0 & 0 & D_r \end{bmatrix}. \tag{4.11}$$

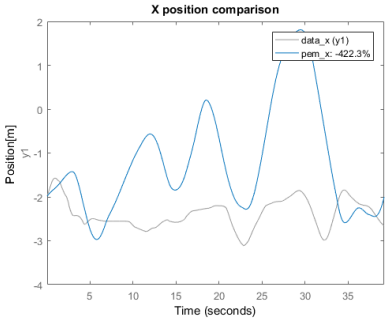
Things of interest to estimate were the elements in the moment of inertia matrix, how the torque depends on the angular velocities of the propellers and the dampening factors in the matrices D_{linvel} and D_{angvel}. Much like thrust, torque is often modelled as a second order polynomial depending on the rotor velocity [Månsson and Stenberg, 2014],[Fogelberg, 2013]. Studying Figure 2.4 and Equation 2.17 it is reasonable so assume that the components of body moment may be modelled as

$$\begin{aligned} \tau_\phi &= M_\phi(\Omega_1^2 + \Omega_4^2 - \Omega_2^2 - \Omega_3^2) \\ \tau_\theta &= M_\theta(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \\ \tau_\psi &= M_\psi(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2) \end{aligned} \tag{4.12}$$

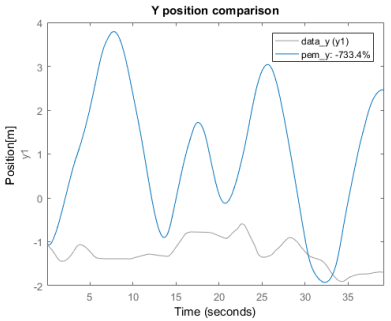
, where Ω_i is the rotor velocity of a given motor, for some constants M_ϕ, M_θ, M_ψ, as long as the quadcopter is decently symmetrical in its construction. Under other circumstances it might be more appropriate to introduce one constant per motor per axis.
As a measure of how well the model fits the data it is possible to study the so called goodness of fit. This may be evaluated using a normalised root mean square error (NRMSE), seen in Equation 4.13.

$$fit = 1 - \frac{\sqrt{\sum_{k=1}^{N}(y(k) - \hat{y}(k))^2}}{\sqrt{\sum_{k=1}^{N}(y(k) - \frac{1}{N}\sum_{k=1}^{N} y(k))^2}} \tag{4.13}$$

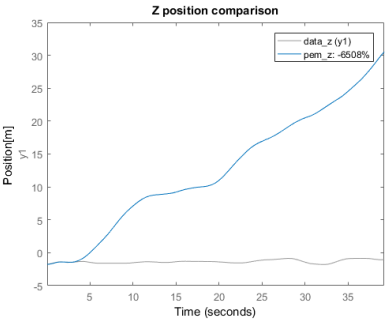
Using this method, the goodness of fit can range from negative infinity to 100 %. PEM fits for the non-linear model while flying in angle mode may be seen in Figures 4.9-4.12.



4.9a) X position fit

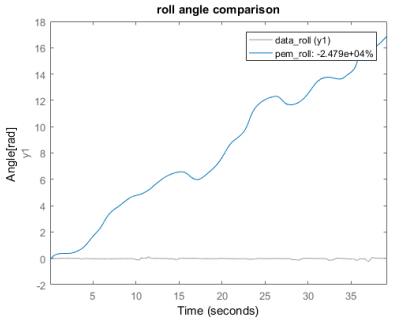


4.9b) Y position fit

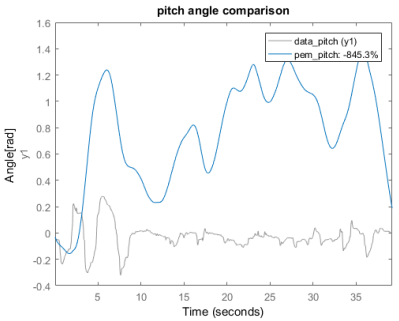


4.9c) Z position fit

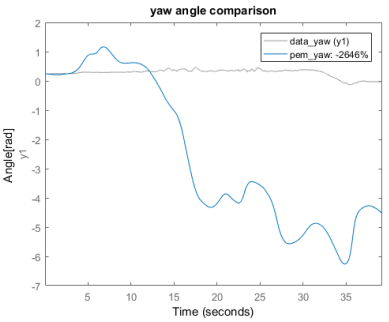
Figure 4.9 Fits for global coordinates using PEM method on data gathered while flying in angle mode. The grey lines are the measurements while the blue lines corresponds to the predictions. As is clearly evident the fits were quite poor.



4.10a) Roll angle fit

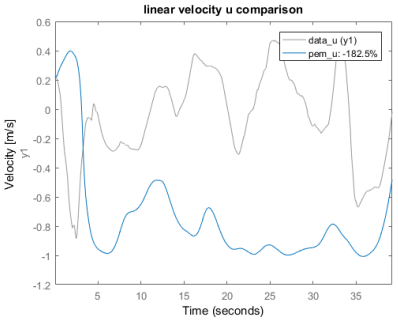


4.10b) Pitch angle fit

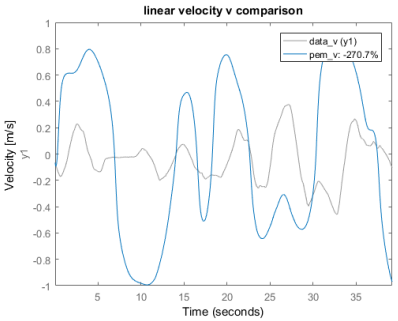


4.10c) Yaw angle fit

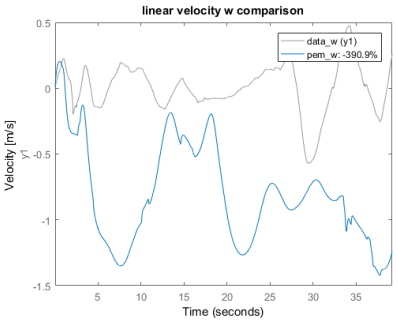
Figure 4.10 Fits for angles using PEM method on data gathered while flying in angle mode. Much like the case with the global positions, the fits were not satisfactory.



4.11a) Linear velocity u fit

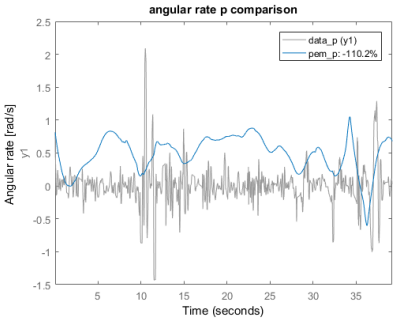


4.11b) Linear velocity v fit

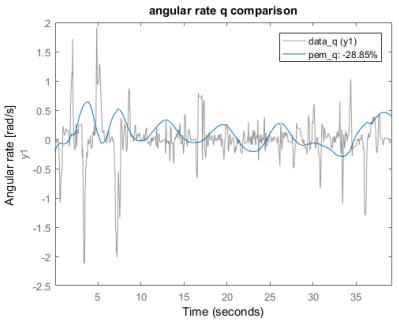


4.11c) Linear velocity w fit

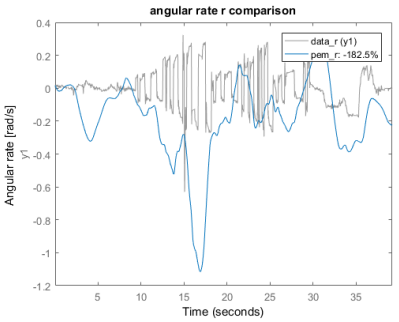
Figure 4.11 Fits for the linear velocities in the body frame, using PEM method on data gathered while flying in angle mode.



4.12a) Angular rate p fit



4.12b) Angular rate q fit

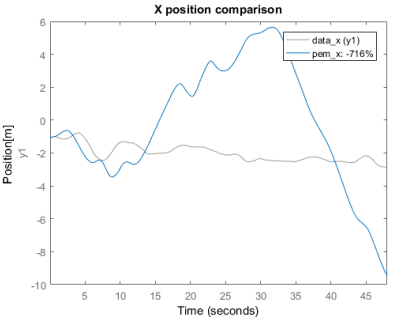


4.12c) Angular rate r fit

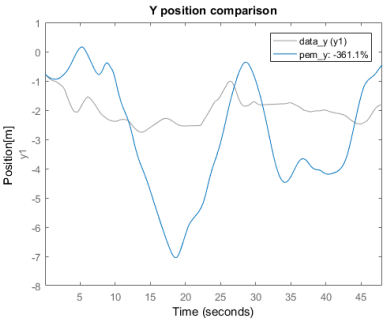
Figure 4.12 Fits for angular rates in the body system using PEM method on data gathered while flying in angle mode.

As all fits for experiments executed while flying in angle mode were quite poor,

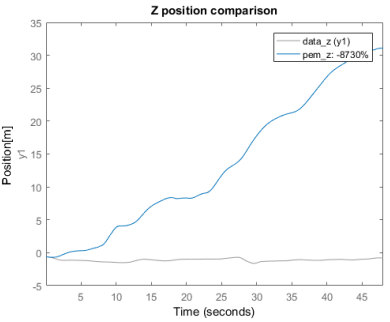
similar experiments where tried out while flying in rate mode, where the accelerometer is not active. Results may be viewed in Figures 4.13 - 4.16.



4.13a) X position fit

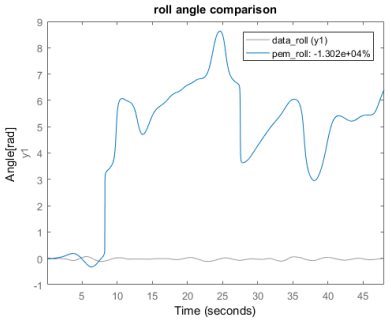


4.13b) Y position fit

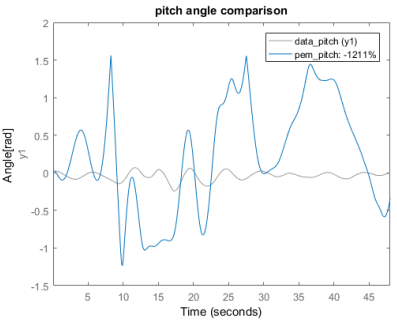


4.13c) Z position fit

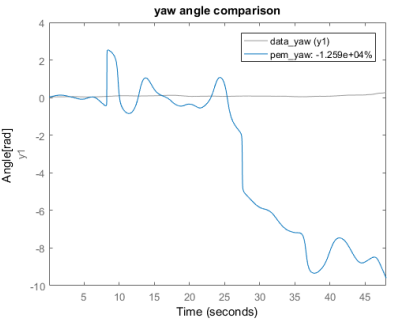
Figure 4.13 Fits for global coordinates using PEM method on data gathered while flying in rate mode. Just like when flying in angle mode, the fits were poor.



4.14a) Roll angle fit

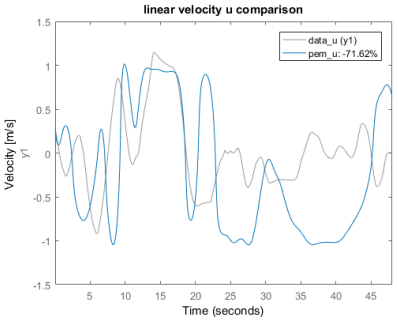


4.14b) Pitch angle fit

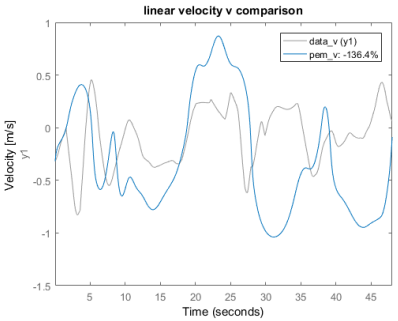


4.14c) Yaw angle fit

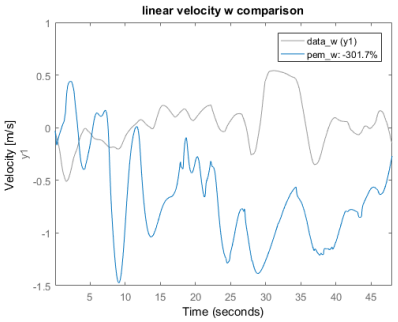
Figure 4.14 Fits for angles using PEM method on data gathered while flying in rate mode.



4.15a) Linear velocity u fit

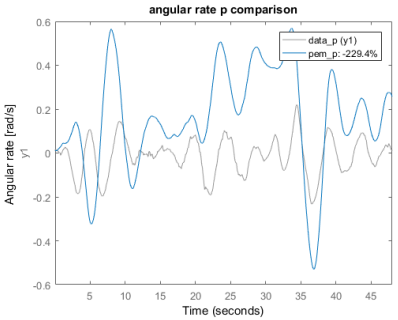


4.15b) Linear velocity v fit

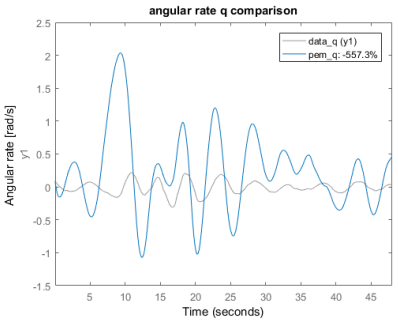


4.15c) Linear velocity w fit

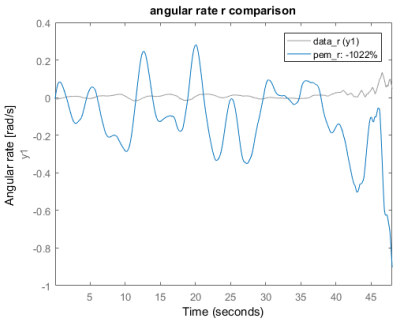
Figure 4.15 Fits for the linear velocities in the body frame, using PEM method on data gathered while flying in rate mode.



4.16a) Angular rate p fit



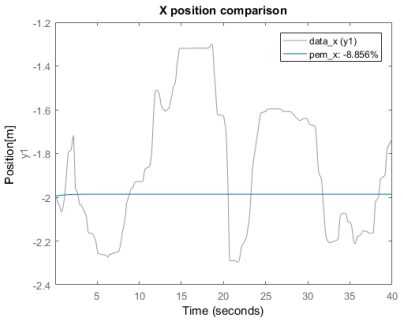
4.16b) Angular rate q fit



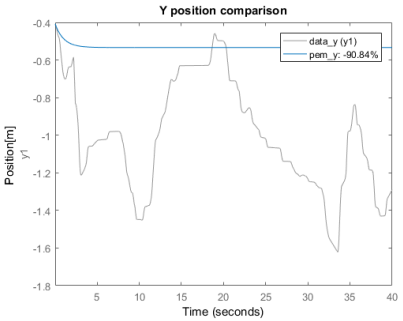
4.16c) Angular rate r fit

Figure 4.16 Fits for angular rates in the body system using PEM method on data gathered while flying in rate mode.

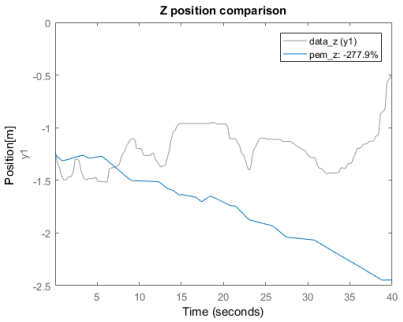
Since neither of the flight modes gave satisfactory results for the non-linear model using prediction error method, similar test were made for the linearised model described in Chapter 2, where the quadcopter is close to a hovering state. This was firstly done while flying in angle mode. See Figures 4.17 - 4.20. Since the platform was operated by a human pilot, small oscillations were likely to happen in all directions. As the linearised model was based on the angles ϕ and θ being close to zero, it was reasonable to assume that fits for them would therefore be quite low.



4.17a) X position fit

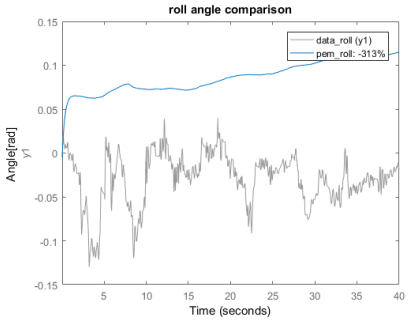


4.17b) Y position fit

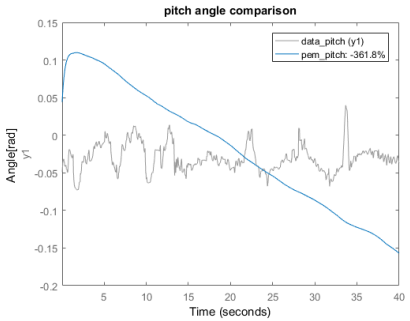


4.17c) Z position fit

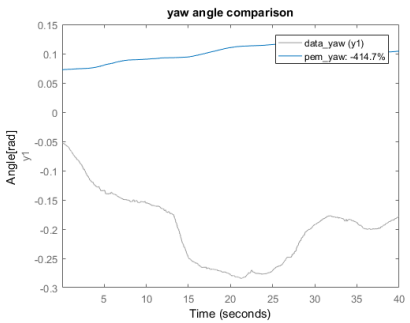
Figure 4.17 Fits for global coordinates in the linearised model, using PEM method on data gathered while flying in angle mode. Fits were clearly quite poor.



4.18a) Roll angle fit

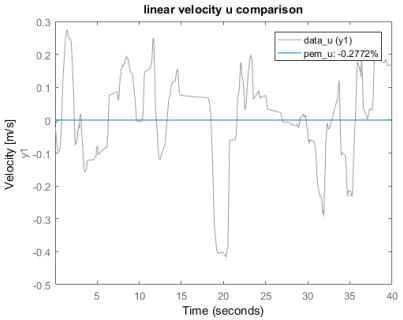


4.18b) Pitch angle fit

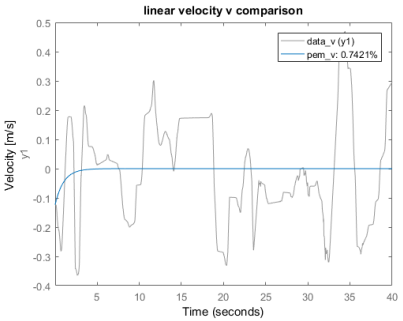


4.18c) Yaw angle fit

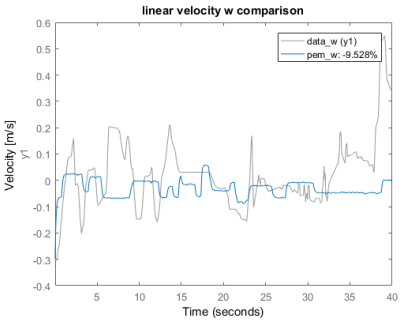
Figure 4.18 Fits for angles in the linearised model, using PEM method on data gathered while flying in angle mode.



4.19a) Linear velocity u fit

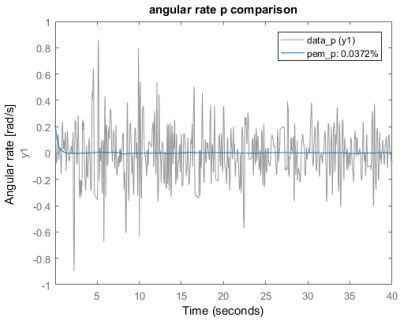


4.19b) Linear velocity v fit

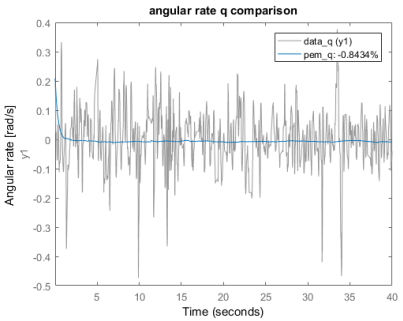


4.19c) Linear velocity w fit

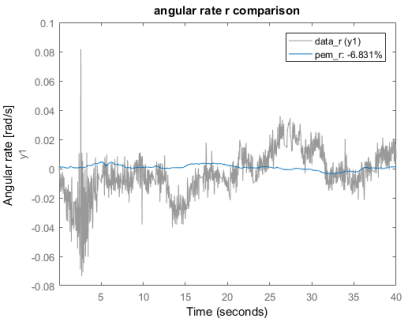
Figure 4.19 Fits for the linear velocities in the body frame, in the linearised model, using PEM method on data gathered while flying in angle mode.



4.20a) Angular rate p fit



4.20b) Angular rate q fit

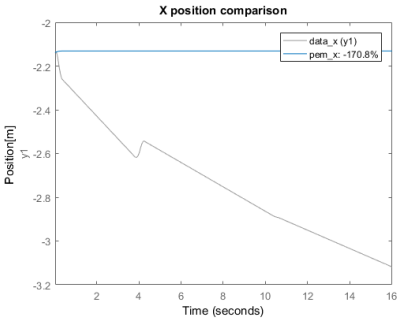


4.20c) Angular rate r fit

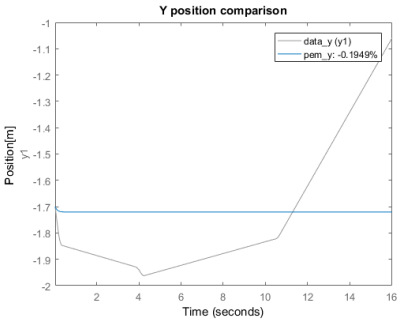
Figure 4.20 Fits for angular rates in the body system, in the linearised model, using PEM method on data gathered while flying in angle mode

Just like the case with the non-linear model, the overall fits were quite poor.

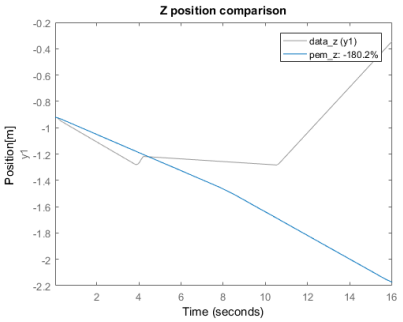
Also here it was decided to try out the experiments also while flying in rate mode. See Figures 4.21 - 4.24.



4.21a) X position fit

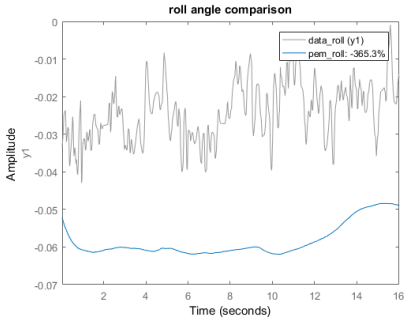


4.21b) Y position fit

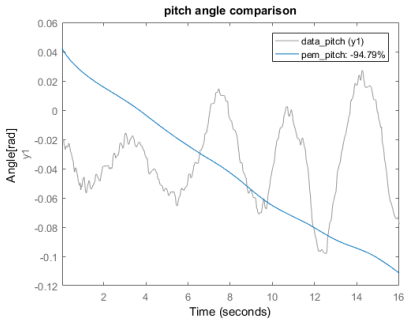


4.21c) Z position fit

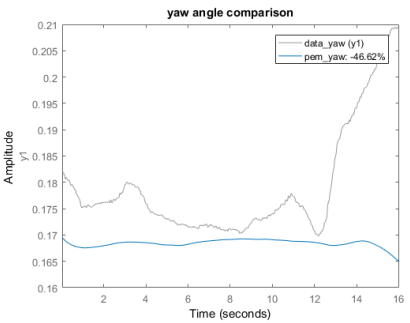
Figure 4.21 Fits for global coordinates in the linearised model, using PEM method on data gathered while flying in rate mode. The fits were evidently poor.



4.22a) Roll angle fit

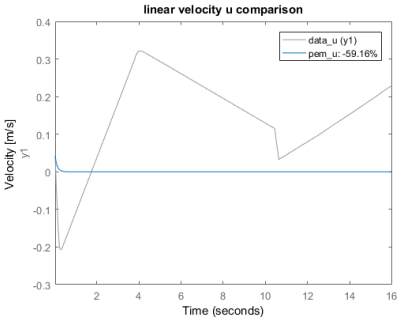


4.22b) Pitch angle fit

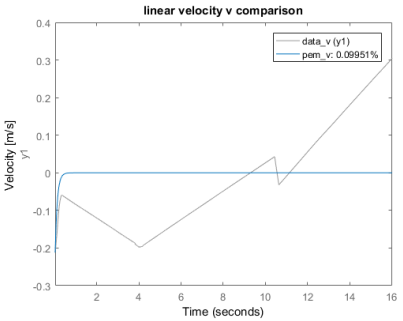


4.22c) Yaw angle fit

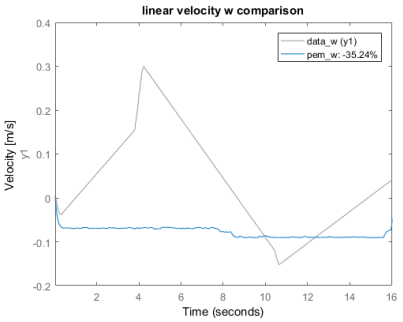
Figure 4.22 Fits for angles in the linearised model, using PEM method on data gathered while flying in rate mode.



4.23a) Linear velocity u fit

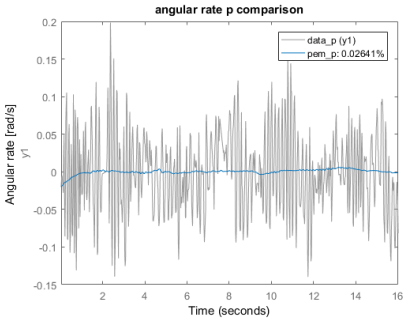


4.23b) Linear velocity v fit

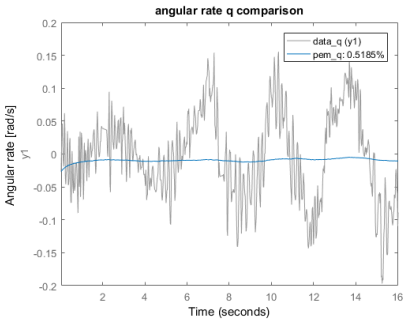


4.23c) Linear velocity w fit

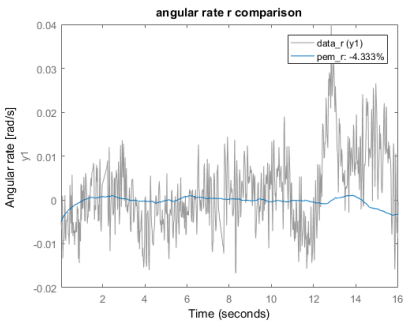
Figure 4.23 Fits for the linear velocities in the body frame, in the linearised model, using PEM method on data gathered while flying in rate mode.



4.24a) Angular rate p fit



4.24b) Angular rate q fit



4.24c) Angular rate r fit

Figure 4.24 Fits for angular rates in the body system, in the linearised model, using PEM method on data gathered while flying in rate mode

It may be argued that fits for the linearised model were better than in the non-

linear case, but not to such a degree that any conclusion about unknown system variables could be drawn.

4.4 Extended Kalman Filter (EKF)

The extended Kalman filter is the non-linear counterpart to the classical Kalman filter. Unlike the standard filter, the EKF is not an optimal predictor. Despite this, and the fact that theoretical support for the EKF is poor, it is widely used [Wan, 2006],[Huang et al., 2008]. In theory it linearises the system around each new working point, thus making it linear at said point of interest. The filter consists of a prediction part and an update part. The predictive part uses previous state estimations in order to generate an estimation of the states at the current timestep. The update phase then combines knowledge about the current prediction and the current observation to refine the estimation of the states of interest. In discrete time, the algorithm in its entirety may be represented as

Prediction (4.14)

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k)$$
$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + G_k Q_k G_k^T$$

Update

$$\tilde{y}_k = z_k - h(\hat{x}_{k|k-1})$$
$$S_k = H_k P_{k|k-1} H_k^T + R_k$$
$$K_k = P_{k|k-1} H_k^T S_k^{-1}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$
$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

where

$$F_k = \frac{\partial f}{\partial x}|_{\hat{x}_{k-1|k-1}, u_k}$$
$$H_k = \frac{\partial h}{\partial x}|_{\hat{x}_{k|k-1}}$$

The different parameters describe insecurities in the model, the measurements, predicted states according to the model and how good the initial guesses for the system states are. Matrices Q and R are covariance matrices that correspond to process and measurement noise respectively. Basically, they tell how much one trusts the model and the measurements, with higher values in the matrices meaning that there is a lot

of noise and lower values indicating absence of noise. Finding suitable covariance matrices is not an exact science, and it may often require a bit of experimentation.

4.5 Parameter estimation using an EKF

Due to the poor fits from the PEM, other means of parameter estimation were examined. It is possible to use an EKF for estimating unknown parameters [Larsson, 2013], this by extending the state vector such that

$$\tilde{x}_k = \begin{bmatrix} x_k \\ \Gamma_k \end{bmatrix} \tag{4.15}$$

where Γ_k are the unknown parameters at time $k \cdot dt$. The sought parameters may be found in Equations 2.7, 2.9, 4.9 and 4.11, and Γ may be represented as $\Gamma = [I_{xx}\ I_{yy}\ I_{zz}\ M_\phi\ M_\theta\ M_\psi\ D_u\ D_v\ D_w\ D_p\ D_q\ D_r]^T$. The added parameters were modelled using constant velocity, see [Gustafsson, 2012], resulting in

$$\hat{x}_{k+1}(\Gamma) = \begin{bmatrix} f(x_k(\Gamma), u_k, \Gamma) \\ \Gamma_k \end{bmatrix}. \tag{4.16}$$

There were 12 states in total, but as half of them depended on derivatives of the other half, there was no need to model their noise separately. I.e. torque affects angular rates, which in turn affects the angles. Hence there was no need for a model with separate noises in angular rates and angles, as the noise in the torque estimation would propagate to the angles per default. Thus the Q matrix would be of size 18×18, 6 separate states with noise, plus the 12 parameters to estimate. The matrix G was of the size 24×18, modelled using constant velocity as aforementioned.

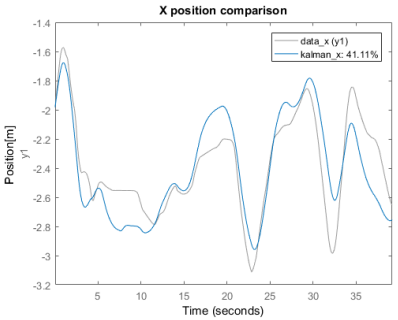
Whereas PEM completely thrusts the model dynamics and the measurements, the EKF may be tuned such that it estimates both states and sought parameters. As the sought parameters were mostly constants (drag coefficients may change due to airflow and a few other parameters, but that effect was deemed negligible in this thesis), it seemed reasonable to assume that they were "less noisy" than the states. After some trial and error it was found that matrices

$$Q = diag(\overbrace{1000}^{6} \ldots \overbrace{0.001}^{12})$$

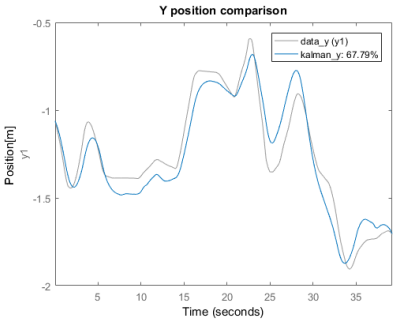
and

$$R = diag(\underbrace{\overbrace{10}^{3}}_{camera} \ldots \underbrace{\overbrace{1}^{3}}_{IMU} \ldots \underbrace{\overbrace{10}^{3}}_{camera} \ldots \underbrace{\overbrace{1}^{3}}_{IMU})$$

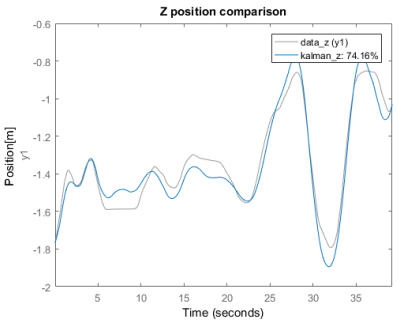
yielded decent results, as may be seen in Figures 4.25 - 4.28.



4.25a) X position fit

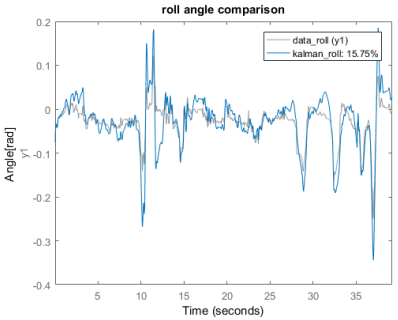


4.25b) Y position fit

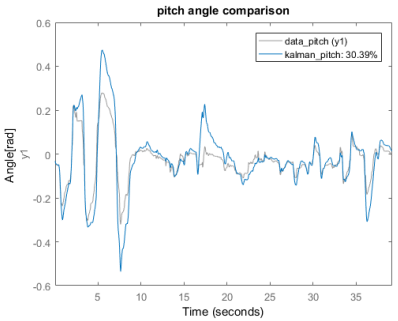


4.25c) Z position fit

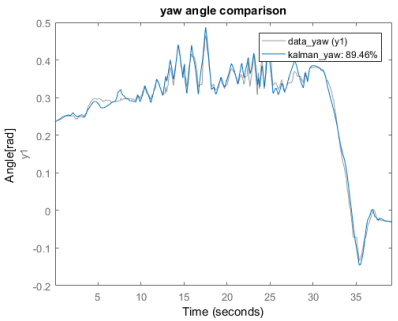
Figure 4.25 Fits for global coordinates in using an EKF. Data gathered while flying in angle mode. Compared to the fits achieved using the prediction error method, a significant improvement was noted.



4.26a) Roll angle fit

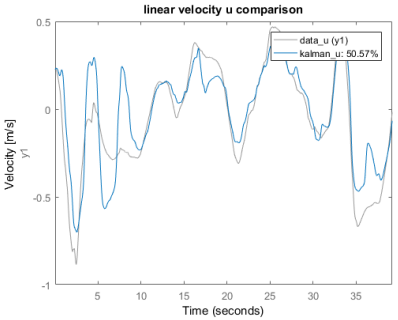


4.26b) Pitch angle fit

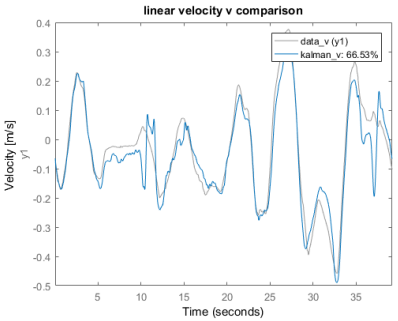


4.26c) Yaw angle fit

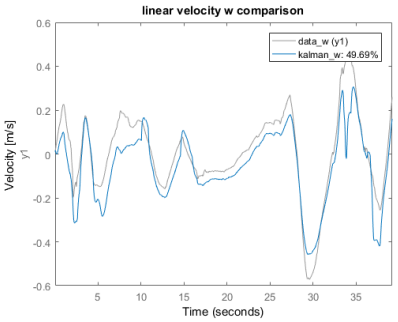
Figure 4.26 Fits for angles using an EKF. Data gathered while flying in angle mode. Roll and pitch angle estimation were mediocre at best.



4.27a) Linear velocity u fit

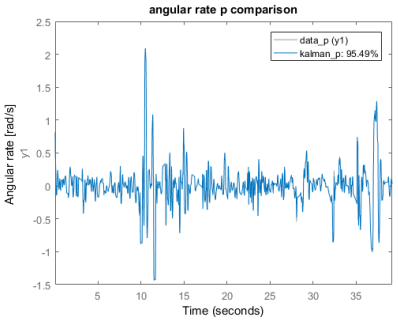


4.27b) Linear velocity v fit

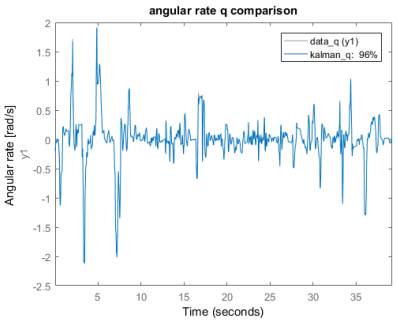


4.27c) Linear velocity w fit

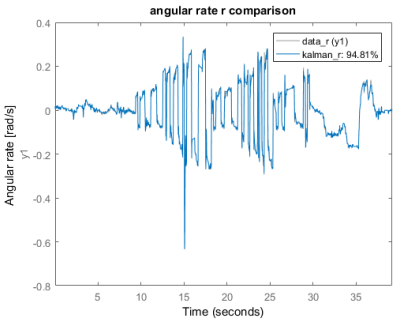
Figure 4.27 Fits for the linear velocities in the body frame, using an EKF. Data gathered while flying in angle mode.



4.28a) Angular rate p fit



4.28b) Angular rate q fit

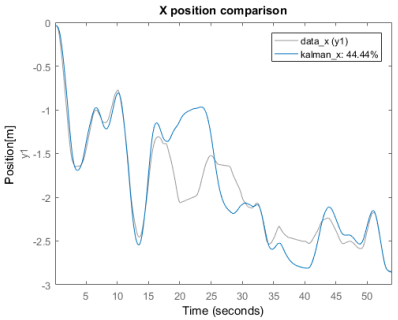


4.28c) Angular rate r fit

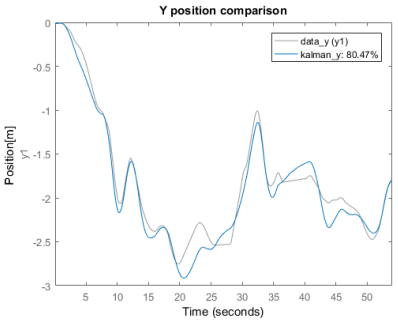
Figure 4.28 Fits for angular rates using an EKF. Data gathered while flying in angle mode.

Overall, the EKF approach was much more successful than the PEM. As the

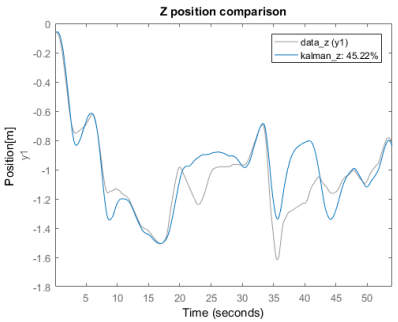
Kalman filter estimates the unknown parameters over time, it naturally means that they are not constant throughout the process. Estimated values were used in another data set, but this time the sought parameters were set to constants at the beginning. The states on the other hand, were still able to be estimated using the filter and the measurements. See Figures 4.29 - 4.32.



4.29a) X position fit

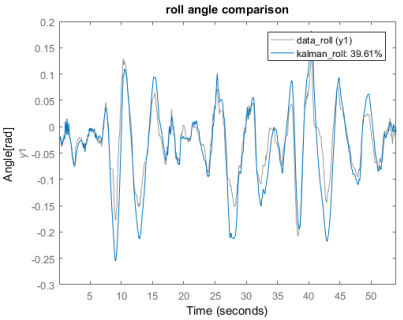


4.29b) Y position fit

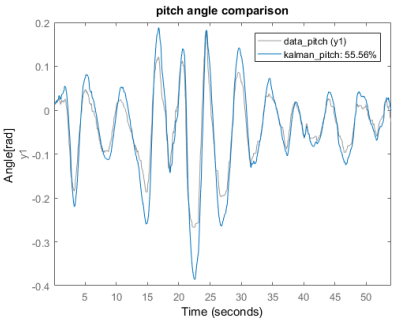


4.29c) Z position fit

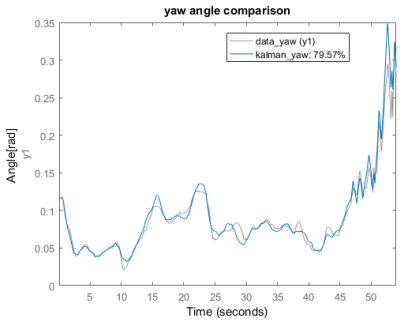
Figure 4.29 Fits for global coordinates when using an EKF with constant parameters.



4.30a) Roll angle fit

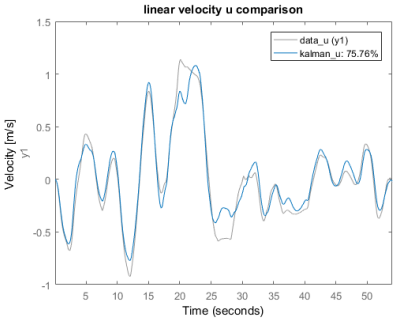


4.30b) Pitch angle fit

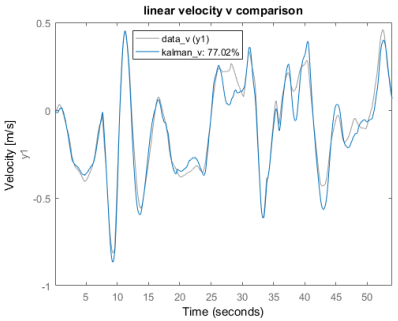


4.30c) Yaw angle fit

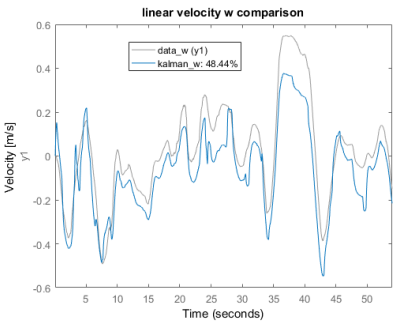
Figure 4.30 Fits for angles using an EKF with constant parameters.



4.31a) Linear velocity u fit

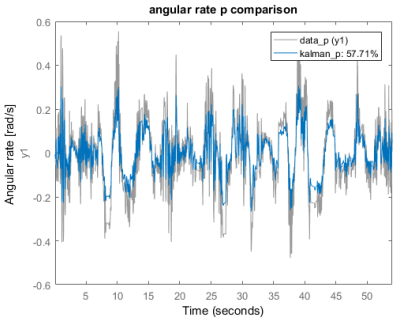


4.31b) Linear velocity v fit

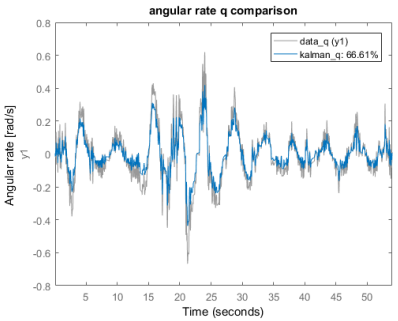


4.31c) Linear velocity w fit

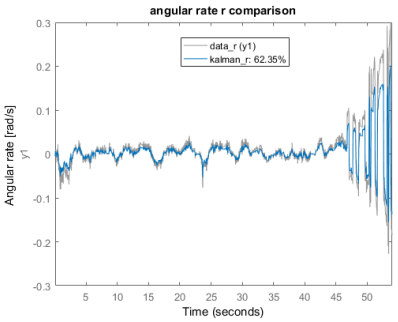
Figure 4.31 Fits for the linear velocities in the body frame, using an EKF with constant parameters.



4.32a) Angular rate p fit



4.32b) Angular rate q fit



4.32c) Angular rate r fit

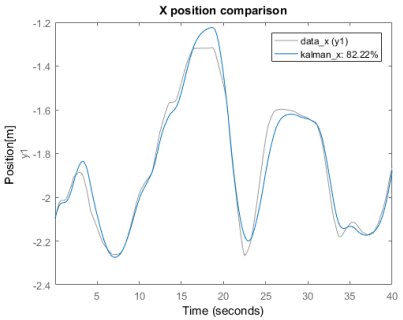
Figure 4.32 Fits for angular rates using an EKF with constant parameters.

The parameters were found to lie in the vicinity of values found in Table 4.1.

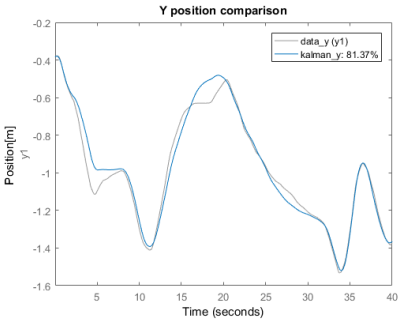
Parameter	*Value*	*Unit*
I_{xx}	0.006	Kgm^2
I_{yy}	0.007	Kgm^2
I_{zz}	0.008	Kgm^2
M_ϕ	$2 \cdot 10^{-7}$	Nm
M_θ	$3 \cdot 10^{-7}$	Nm
M_ψ	10^{-7}	Nm
D_u	1	Kg^2/m
D_v	1.2	Kg^2/m
D_w	2	Kg^2/m
D_p	1	$Kgm^2/rad \cdot s$
D_q	1.1	$Kgm^2/rad \cdot s$
D_r	1.2	$Kgm^2/rad \cdot s$

Table 4.1 Estimated parameters using the method with an extended Kalman filter.

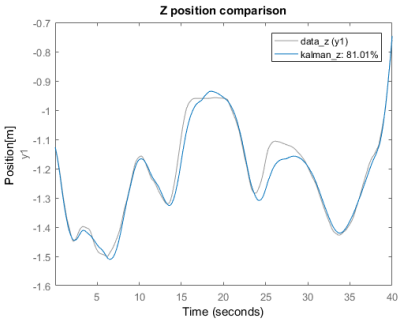
For good measure, the parameters were tried out also on a data set where the quadcopter was close to a hovering state, see Figures 4.33 - 4.36.



4.33a) X position fit

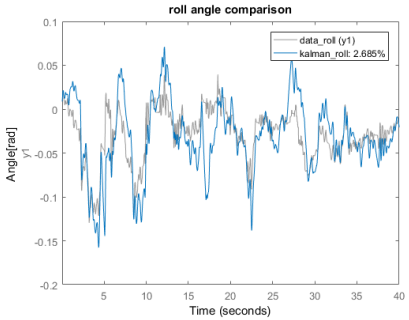


4.33b) Y position fit

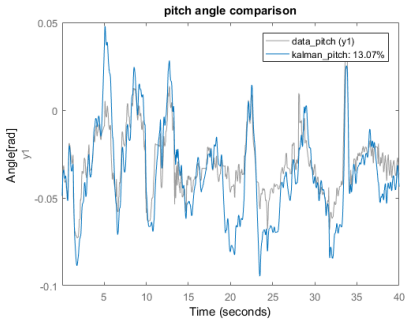


4.33c) Z position fit

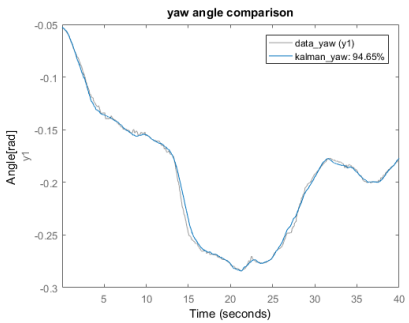
Figure 4.33 Fits for global coordinates when using an EKF with constant parameters. Flight was close to a hovering state.



4.34a) Roll angle fit

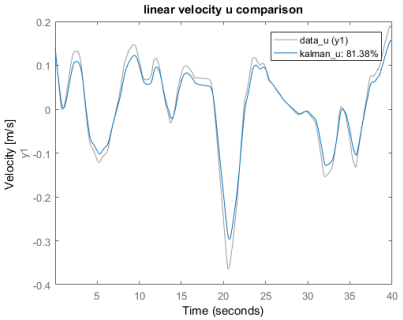


4.34b) Pitch angle fit

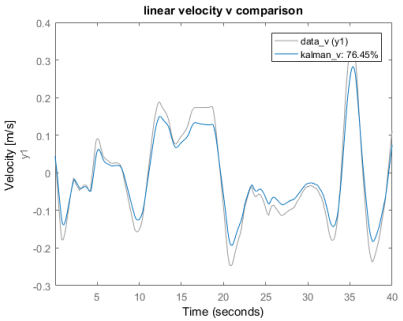


4.34c) Yaw angle fit

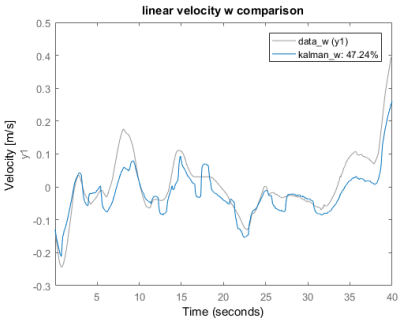
Figure 4.34 Fits for angles using an EKF with constant parameters while close to hovering. As was to be expected, fits for roll and pitch were quite low as the model was linearised around them both being zero.



4.35a) Linear velocity u fit

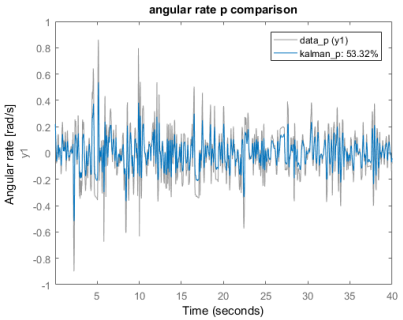


4.35b) Linear velocity v fit

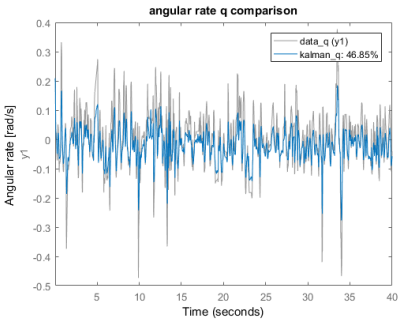


4.35c) Linear velocity w fit

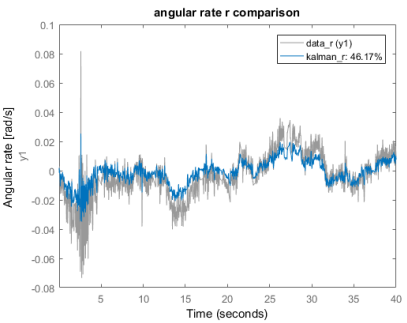
Figure 4.35 Fits for the linear velocities in the body frame, using an EKF with constant parameters.



4.36a) Angular rate p fit



4.36b) Angular rate q fit



4.36c) Angular rate r fit

Figure 4.36 Fits for angular rates using an EKF with constant parameters.

Aside from the roll and pitch angles, fit for the linearised model were, overall, quite high.

5

Control design

This chapter concerns theory and implementation of regulators used to control the quadcopter.

5.1 PID control

Simplicity combined with good performance renders PID one of the most common controller methods. A few different ways of describing it exists. The most usual one is the parallel form, which may be written as

$$u(t) = K_p e(t) + K_i \int_{t_0}^{t} e(t)dt + K_d \frac{de(t)}{dt}. \tag{5.1}$$

$u(t)$ is the control signal, $e(t)$ is the control error defined as $r(t) - y(t)$, where $r(t)$ is the reference output and $y(t)$ is the current output. K_p, K_i, and K_d are the gains for the proportional, integral and derivative part respectively. As computers do not run in continuous time, Equation 5.1 needs to be discretized in order to be implemented on the quadcopter platoform. This leads to

$$u_k = K_p e_k + K_i \sum_{j=1}^{N} e_j T_s + K_d \frac{e_k - e_{k-1}}{T_s} \tag{5.2}$$

, where the added parameter T_s is the sampling time. Block diagram for a simple process with a PID regulator is found in Figure 5.1.

5.2 Implementation aspects of PID

A common problem when implementing PID controllers is integral windup. If a reference of any form for some reason can not be achieved (e.g. heavy wind to one side), there will be constant errors, causing the integral part of the PID to accumulate. If the reference at a later point can be achieved (e.g. the wind drops) the

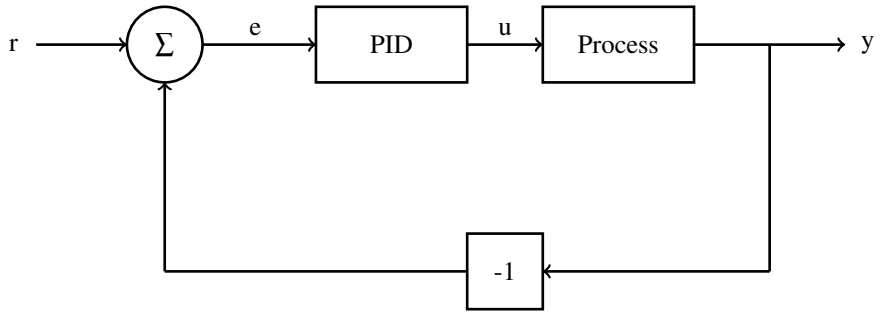


Figure 5.1 Block diagram of a simple PID regulator

integral part may have accumulated such that it results in an overshoot. Several solutions were implemented to prevent this:

- When flying in self-levelling mode the integral parts were not updated until the throttle was sufficiently high. Otherwise they might build up before even starting to fly if the platform was not perfectly balanced before take off.
- If the throttle was very low, all integral parts would be set to zero. This way it was possible to land and take off again without integral parts affecting the current flight using error measurements from the previous one.
- If a control signal was saturated or if an error was particularly high, the corresponding integral part would not be updated.

Control signals in discrete time may be very sharp, resembling square waves. As these changes happen during a very short time span, the derivative parts of PID controller may become unreasonably large. High frequency noise, which is common in most processes, also adds to derivative gain. To avoid these problems it is possible to apply a first order filter such that the derivative gain does not amplify high frequency content. With these aspects considered, the PID algorithm will be represented by

$$\begin{aligned} P_k &= K_p e_k \\ I_k &= I_{k-1} + K_i e_k T_s \\ D_k &= \frac{K_d}{K_d + N T_s K_p} D_{k-1} + \frac{K_d N}{K_d/K_p + N T_s} (y_k - y_{k-1}) \end{aligned} \tag{5.3}$$

, where N is a design parameter affecting the maximum derivative gain. Suitable PID configuration will differ depending on if flight is done in angle mode or rate mode. Parameter values that were found to have good performance can be seen in Table 5.1.

	Angle mode			Rate mode		
	Roll	Pitch	r rate	p rate	q rate	r rate
K_p	0.15	0.15	0.1	0.05	0.05	0.1
K_i	0.54	0.54	0.15	0.15	0.15	0.15
K_d	0.006	0.006	0.0	0.0	0.0	0.0
N	5.0	5.0	0.0	0.0	0.0	0.0

Table 5.1 Suitable PID parameters depending on whether flight was performed using angle mode or rate mode. Both modes control rotational rate around the z-axis. Yaw angle is never directly regulated. N is zero by default if $K_d = 0$. Introducing derivative gain in rates merely lead to oscillations and was thus neglected.

6

Conclusions and discussion

System identification of the quadcopter, using motion capture proved to be more difficult than anticipated, no matter whether a non-linear or a linear model was used. Reasons as to why the prediction error method provided very poor fits were not wholly concluded. It might be that the already provided Kalman filter used in the rosnode when fetching camera data needs to be tuned significantly. As is seen in Figure 4.8, it seems to suffer from shortcomings from time to time. It may also be that the mathematical model describing the dynamics of the system needs to be reevaluated.
The EKF approach proved to be a much more reliable way to go in this project. That may be because it does not completely trust the state measurements of the camera system and the IMU, as opposed to the PEM. Results provided in Table 4.1 are not considered absolute, but they appear to work satisfactory with the model and measurements.
When the EKF were to estimate the parameters, fits of roll and pitch angles were decent at best whereas fits of angular rates were excellent. In the experiment where estimated parameters were set to constants, fits for roll and pitch improved while fits for angular rate deteriorated. This might be a side effect of the estimations of torques being the most volatile of the parameters to estimate. It is unclear why the torques seem to be very difficult to estimate. It may be a side effect of them being very small, leading way for numerical problems.
PID controllers perform satisfactory both in angle mode as well as in rate mode.

7

Future work

At the beginning of this thesis, it was planned that the system identification was to lead way for trajectory generation and path following. As the hardships of parameter estimation began to show, it became clear at a certain point that there would unfortunately not be enough time left for this to be tried out. It could on the other hand act as a natural follow-up to this work. In the future it might be useful to try to estimate more parameters by experiments where camera measurements are not necessary, if nothing else than to provide a hint such that a harder constraint may be imposed on the guessing interval. Also, it would most likely be useful to decouple the model in order to estimate the drag coefficients. With a height regulator and an angle regulator it should be possible to go straight forward in one direction at a constant height and angle until a constant velocity is achieved, and thereafter calculate the drag coefficients D_u and D_v. Similar experiments should be possible with a regulator for angular rate in order to evaluate the dampening coefficients for angular rates. Instead of using an extended Kalman filter, it might be worth trying an unscented Kalman filter, as it may overcome a few of the limitations imposed by EKF's.
As of now the code run on the quadcopter is heavily interrupt based. Using threads and semaphores may help to improve performance.

A

Notations and abbreviations

Notations

Notation	Description
[x y z]	Inertial frame coordinates
$[\phi\ \theta\ \psi]$	Euler angles, roll, pitch and yaw
[u v w]	Linear velocities in boxy-fixed frame
[p q r]	Angular velocities in boxy-fixed frame
u_x	Control signal for x
$[c_x\ s_x\ t_x]$	cosine(x), sine(x) and tangent(x)
$\hat{x}$	estimate value of x

Abbreviations

Abbreviation	Description
PID	Proportional, Integral, Derivative
ESC	Electronic Speed Controller
NED	North-East-Down (coordinate system)
IMU	Inertial Measurement Unit
ROS	Robot operating System
GUI	Graphical User Interface
SISO	Single Input Single Output
MIMO	Multiple Input Multiple Output

Bibliography

Bergman, K. and J. Ekström (2014). *Modeling, Estimation and Attitude Control of an Octorotor Using PID and L1 Adaptive Control Techniques*. URL:http://www.diva-portal.org/smash/get/diva2:725353/FULLTEXT01.pdf. MA thesis. Linköping University.

Bernard, D. D. C., F. Riccardi, M. Giurato, and M. Lovera (2017). *A dynamic analysis of ground effect for a quadrotor platform*. Polytechnic university of Milan.

Camera system rosnode. Accessed 2018-04-12. URL: URL : \url{https://github.com/KumarRobotics/motion_capture_system}.

Chéron, C., A. Dennis, V. Semerjyan, and C. YangQuan (2010). *A multifunctional hil testbed for multirotor vtol uav actuator*. Published in International conference on mechatronics and embedded systems and applications.

Droneomega propeller guide. Accessed 2018-05-23. URL: URL : \url{http://www.droneomega.com/quadcopter-propeller/}.

Fogelberg, J. (2013). *Navigation and Autonomous Control of a Hexacopter in Indoor Environments*. URL:http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=4359940&fileOId=4359943. MA thesis. Lund University.

Förster, J. (2015). *System identification of the Crazyflie 2.0 nano quadrocopter*. URL: http://mikehamer.info/assets/papers/Crazyflie%20Modelling.pdf. Eidgenössische Technishe Hochschule Zürich.

Gustafsson, F. (2012). *Statistical sensor fusion - second edition*. Studentlitteratur AB. ISBN: 978-91-44-07732-1.

How to choose propeller for a mini quad. Accessed 2018-05-31. URL: URL : \url{https://oscarliang.com/choose-propellers-mini-quad/}.

Huang, G., A. Mourikis, and S. Roumeliotis (2008). *Analysis and improvement of the consistency of extended kalman filter based slam*.

Kugelberg, I. (2016). *Black-box modeling and attitude control of a quadcopter*. URL: https://liu.diva-portal.org/smash/get/diva2:908582/FULLTEXT02.pdf. MA thesis. Linköping University.

Landry, B. (2015). *Planning and control for quadrotor flight through cluttered environments*. URL:http://groups.csail.mit.edu/robotics-center/public_papers/Landry15.pdf. MA thesis. Massachusetts Institute of Technology.

Larsson, R. (2013). *System Identification of Flight Mechanical Characteristics*. URL:http://liu.diva-portal.org/smash/get/diva2:622859/FULLTEXT01.pdf. PhD thesis. Linköping University.

Levine, W. S. (2011). *Control system fundamentals - Second edition*. Taylor and Francis group. ISBN: 978-1-4200-7363-8.

Ljung, L. (1999). *System Identification: Theory for the user, second edition*. Prentice Hall. ISBN: 0-13-881640-9.

Månsson, C. and D. Stenberg (2014). *Model-based design development and control of a wind resistant multirotor uav*. Accessed 2018-05-29, URL: http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=4610294&fileOId=4610297.

Nilsson, R. (1998). *Flight stability and automatic control - second edition*. McGraw-Hill. ISBN: 0-07-115838-3.

Orwiler, B. (1969). *Oscilloscope vertical amplifiers*. Tektronix.

Reizenstein, A. (2017). *Position and trajectory control of a quadcopter using PID and LQ controllers*. URL: https://liu.diva-portal.org/smash/get/diva2:1129641/FULLTEXT01.pdf. MA thesis. Linköping University.

ROS - concepts description. Accessed 2018-02-07. URL: URL : \url{http://wiki.ros.org/ROS/Concepts}.

Thornton, S. and J. Marion (2004). *Classical dynamics of particles and systems - fifth edition*. Brooks/Cole, Thomson learning. ISBN: 0-534-40896-6.

Wan, E. (2006). *Sigma-point filters: an overview with applications to integrated navigation and vision assisted control*.

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

Document name
MASTER'S THESIS

Date of issue
September 2018

Document Number
TFRT-6058

Author(s)
Martin Ottenklev

Supervisor
Kristoffer Bergman, SAAB Dynamics
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden
Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Evaluating motion capture as a means of system identification of a quadcopter

Abstract

This thesis describes the method of identifying unknown parameters that affect the dynamics of a given quadcopter, also known as grey box system identification. This was primarily done utilising an inertial measurement unit and a motion capture camera system. A system of equations describes the dynamics of the quadcopter, and was later coupled with data gathered while flying, in order to use different methods of system identification. As quadcopters are unstable, the first task was to design a stabilising regulator, making stable flight possible, and thus gathering flight data.

A few parameters regarding motor dynamics were evaluated via simple experiments with tools including a tachometer, a scale and a microphone. When it comes to flight dynamics, the first method of identification was to use a prediction error method which, given data regarding input signals, output signals and a mathematical model, tries to evaluate unknown parameters by minimising the error between state measurements and estimated states based on the earlier mentioned model, in each timestep. This method proved to be unsuccessful, for reasons partly unknown, and was later changed for a method utilising an extended Kalman filter, which gave more reliable results. Possible explanations to this phenomena may include that the Kalman filter implemented beforehand in the camera system may need to be retuned and that the aforementioned mathematical model needs to be reevaluated.

Estimated parameter values works well with the model, but that is not so say that there is not room for improvement.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-75

Recipient's notes

Security classification

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