Position and Trajectory Control of a Quadcopter Using PID and LQ Controllers

Axel Reizenstein
Master of Science Thesis in Electrical Engineering

Position and Trajectory Control of a Quadcopter Using PID and LQ Controllers

Axel Reizenstein

LiTH-ISY-EX--17/5075--SE

Supervisors:

Kristoffer Bergman
ISY, Linköpings universitet

Erik Ekelund
SAAB Dynamics

Examiner:

Daniel Axehill
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Copyright © 2017 Axel Reizenstein
It’s only a model.
Abstract

This thesis describes the work done to implement and develop position and trajectory control of a quadcopter. The quadcopter was originally equipped with sensors and software to estimate and control the quadcopter’s orientation, but did not estimate the current position. A GPS module, GPS antenna and a LIDAR have been added to measure the position in three dimensions. Filters have been implemented and developed to estimate the position, velocity and acceleration. Four controllers have been designed that use these estimates: one PID controller and one LQ controller for vertical movement, and a position controller and a trajectory controller for horizontal movement. The position controller maintains a constant position, while the trajectory controller maintains a constant velocity while travelling along a straight line. These position and trajectory controllers calculate the reference angles required to direct the thrust necessary to control the quadcopter’s movement. Additionally, an algorithm has been developed to decrease overshoot by predicting future trajectories. These controllers have proven to be successful at controlling the quadcopter’s position in all three dimensions, both in practice during outdoor flight and in simulations.
Acknowledgments

This thesis represents the culmination of the 5 most important and challenging years of my life. Completing this thesis can partially be credited to my own work, but the individuals without whom success would not have been certain or even possible are too many to list. I will, however, attempt to summarise.

My family, for offering support and for being there when they were needed most. Thank you for reminding me that at the end of the longest days, home awaits.

My examiner, Daniel, and supervisors, Kristoffer and Erik, for helping me and holding me to a high standard. I could not have corrected the faults and flaws of my work if you had not found them.

Everyone I worked with at SAAB Dynamics. Thank you for offering assistance and creating a wonderful working environment. I would especially like to thank John, Johan, Torbjörn, Anders and Björn.

My opponent, Petter, for his fascinating thesis and his thorough analysis of my own.

All my peers, for putting up with me for all these years. I hope this is not goodbye.

Linköping, June 2017
Axel Reizenstein
Contents

Notation xi

1 Introduction 1
  1.1 Background ................................................. 1
  1.2 Problem Formulation ........................................ 1
  1.3 Purpose .................................................... 2
  1.4 Related Work ................................................ 2
  1.5 Limitations .................................................. 2
  1.6 Outline of Thesis ............................................ 3

2 Preliminaries 5
  2.1 Euler Angles .................................................. 5
  2.2 Modelling and Control of Quadcopter Angles ................. 7
  2.3 Kalman Filter ............................................... 8
  2.4 LQ Control .................................................. 9

3 Platform 11
  3.1 Hardware .................................................... 12
  3.2 Program Architecture ....................................... 12
  3.3 LIDAR Measurements ....................................... 15

4 Modeling 17
  4.1 LIDAR Measurements ....................................... 17
  4.2 Model Simplification ....................................... 18
  4.3 Simulation Environment .................................... 26

5 Position Estimation 27
  5.1 Extended Kalman Filter .................................... 27
  5.2 Height Estimation .......................................... 27
  5.3 Height Kalman Filter ....................................... 28
    5.3.1 Model Structure of Height ............................. 29
    5.3.2 Static Thrust Estimation .............................. 30
    5.3.3 Noise Estimation ..................................... 32
## Notation

### Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[x\ y\ z]$</td>
<td>Inertial frame coordinates</td>
</tr>
<tr>
<td>$[X\ Y\ Z]$</td>
<td>Body-fixed coordinates</td>
</tr>
<tr>
<td>$[\phi\ \theta\ \psi]$</td>
<td>Euler angles roll, pitch and yaw</td>
</tr>
<tr>
<td>$[p\ q\ r]$</td>
<td>Angular rates in body-fixed coordinates</td>
</tr>
<tr>
<td>$\mathcal{R}$</td>
<td>Rotation matrix for vectors</td>
</tr>
<tr>
<td>$T$</td>
<td>Rotation matrix for angular velocities</td>
</tr>
<tr>
<td>$u_x$</td>
<td>Control signal for $x$</td>
</tr>
<tr>
<td>$[c_x\ s_x]$</td>
<td>$\cos(x)$ and $\sin(x)$</td>
</tr>
<tr>
<td>$\hat{x}$</td>
<td>Estimated value of $x$</td>
</tr>
<tr>
<td>[LAT LON]</td>
<td>Latitude and Longitude angles in radians</td>
</tr>
<tr>
<td>$\phi_m$</td>
<td>Phase margin</td>
</tr>
<tr>
<td>$\omega_c$</td>
<td>Cross-over frequency</td>
</tr>
</tbody>
</table>

### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PID</td>
<td>Proportional, Integral, Differential</td>
</tr>
<tr>
<td>LQR</td>
<td>Linear Quadratic Regulator</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>LIDAR</td>
<td>Light Detection And Ranging</td>
</tr>
<tr>
<td>ESC</td>
<td>Electronic Speed Controller</td>
</tr>
<tr>
<td>ICC</td>
<td>Inflight Compass Calibration</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
</tr>
<tr>
<td>NED</td>
<td>North-East-Down</td>
</tr>
</tbody>
</table>
This thesis details the work performed at the request of SAAB Dynamics to implement navigational functionality on a quadcopter. This chapter will summarise the purpose of this thesis, and the work it builds upon. A brief overview of the outline of the thesis will also be provided.

1.1 Background

The quadcopter design is efficient in that it allows for the ability to control the orientation and movement in three dimensions using only four moving parts. The downside is that the system has no natural stabilising elements, but instead relies on software algorithms for stability. Combined with the fact that the system is underactuated, this means that advanced estimation and controller algorithms are required to control the quadcopter (Zemalache et al., 2005).

The platform used in this thesis is provided by SAAB Dynamics. Previously, Kugelberg (2016) used black-box modeling to implement angle controllers. The navigational algorithms developed in this thesis utilise these angle controllers to allow for control of the quadcopter’s position. These navigational algorithms can be used by other students in the future to implement additional functionality.

1.2 Problem Formulation

One way to implement a trajectory controller is to control the angles in an inner control loop, with the trajectory controller as the outer loop (Bonna and Camino, 2015). This outer loop can be implemented using LQ or PID controllers. Evaluating and implementing these controllers requires models of the quadcopter’s movement (Glad and Ljung, 2006). Accurate state estimates are required as input
to these controllers, and can be generated using a Kalman filter or EKF (Gustafsson, 2012). Implementing these filters requires models of the estimated states, noise estimation and measurement equations.

Often, nonlinear models of the quadcopter is used to generate the controllers (Bangura and Mahony, 2012). Kugelberg (2016) assumed that the quadcopter was close to a hovering state to approximate the models used as linear, and by assuming that the quadcopter moves with a velocity low enough to neglect drag, models used for positioning can be approximated as linear as well.

### 1.3 Purpose

The goal of this thesis is to design and implement controllers of the quadcopter’s position and trajectory in an outdoor environment. These controllers will generate reference angles for the inner loop angle controllers. The controllers require estimates of the quadcopter’s position in three dimensions. To estimate the position new sensors will have to be integrated along with algorithms to generate position estimates from these new sensor measurements. Using these controllers, it will be possible to program the quadcopter to perform missions, which will consist of a set of coordinates or a path that the quadcopter will follow.

### 1.4 Related Work

The work done previously on the same platform by Kugelberg (2016) was focused on estimating the dynamics of the quadcopter using black box models in order to implement PID angle controllers. Additionally, various algorithms were implemented to handle control signal prioritisation, control signal saturation and integrator wind-up.

Various research has been done on methods to control the position and trajectory of a quadcopter. Abdolhosseini et al. (2013) explores the usage of an MPC controller to follow a reference trajectory. Santana et al. (2015) implements a GPS for outdoor waypoint following, using a base station to generate control signals.

### 1.5 Limitations

The position and velocity of the quadcopter is estimated in a local Cartesian coordinate system instead of global, spherical coordinates. This results in a limited range before the Cartesian estimates starts to deviate from the actual position.

Vertical position is measured using a LIDAR, which means that the vertical position will be estimated as height (above ground) instead of altitude (above sea level). Additionally, the area where the quadcopter operates will be flat with little or no inclination.

The angle controllers developed by Kugelberg (2016) assumes that the quadcopter is close to hovering. This assumption is kept in this thesis.
1.6 Outline of Thesis

Each task in this thesis builds upon previous ones. The chapters are thus arranged chronologically. Chapter 2 is a summary of the theory used in this thesis. Chapter 3 describes the hardware and program architecture of the platform’s current configuration. In Chapter 4, theoretical modeling of the quadcopter is performed. Chapter 5 explains how the algorithms estimating the position of the quadcopter were developed. The implementation of the position controllers is described in Chapter 6. Finally, Chapter 7 contains the discussion of conclusions and future work.
Preliminaries

The work done in this thesis involves modeling, control, state estimation and sensor fusion. This chapter will give a brief overview of the theory necessary to understand these methods. The notation and terminology used to describe the quadcopter will also be described.

2.1 Euler Angles

According to Nelson (1998) it is convenient to define a body-fixed coordinate system \([X \ Y \ Z]\) in the aircraft. In aeronautic applications, quantities such as acceleration, velocities, and angular rates are often, or at least partially, measured in relation to the aircraft. The body-fixed coordinate system can be used to relate measurements and estimations to the inertial system. In aeronautical applications this is commonly a North-East-Down (NED) coordinate system, with the \(X\) axis pointing fore, the \(Y\) axis pointing starboard and the \(Z\) axis pointing to the keel of the craft. The rotational velocity of the aircraft is measured by the angular rates \(p\) around the \(X\) axis, \(q\) around the \(Y\) axis and \(r\) around the \(Z\) axis. The aircraft’s velocities along the \(X\), \(Y\), and \(Z\) axes are denoted \(u\), \(v\) and \(w\) respectively. To relate the orientation of the local coordinate system relative to a global one, the Euler angles roll \((\phi)\), pitch \((\theta)\), and yaw \((\psi)\) can be used.

Nelson (1998) describes the three rotations needed to transform the global coordinate system \([x \ y \ z]\) to the local system \([X \ Y \ Z]\). Each rotation results in a new coordinate system, and the two intermediate coordinate systems are denoted \([x' \ y' \ z']\) and \([x'' \ y'' \ z'']\). \([X \ Y \ Z]\) is reached by rotating \(\psi\) radians around \(z\), \(\theta\) radians around \(y'\) and \(\phi\) radians around \(x''\). The position and velocity of the aircraft in the global coordinate systems is denoted \([p_x \ p_y \ p_z]\) and \([\dot{p}_x \ \dot{p}_y \ \dot{p}_z]\) respectively. The velocity can according to Nelson (1998) be expressed using the
rotation matrix $\mathcal{R}$ according to

\begin{align}
\begin{bmatrix}
\sin(\phi) & \cos(\phi) \\
\sin(\theta) & \cos(\theta) \\
\sin(\psi) & \cos(\psi)
\end{bmatrix}
\end{align}

\begin{equation}
\begin{bmatrix}
\sin(\phi) \\
\sin(\theta) \\
\sin(\psi)
\end{bmatrix}
\begin{bmatrix}
\cos(\phi) \\
\cos(\theta) \\
\cos(\psi)
\end{bmatrix}
\end{equation}

\begin{equation}
\begin{bmatrix}
\dot{\phi} \\
\dot{\theta} \\
\dot{\psi}
\end{bmatrix}
= \mathcal{R} \begin{bmatrix}
u \\
w
\end{bmatrix}.
\end{equation}

The derivatives of the Euler angles can be expressed using the angular rates as

\begin{equation}
\begin{bmatrix}
\dot{\phi} \\
\dot{\theta} \\
\dot{\psi}
\end{bmatrix}
\approx
\begin{bmatrix}
p \\
q \\
r
\end{bmatrix},
\end{equation}

as described by Nelson (1998). This expression for the derivatives requires that $\theta \neq \frac{\pi}{2}$. When $\phi$ and $\theta$ are close to 0, which corresponds to the quadcopter being close to hover, $T$ is approximately a unit matrix. In this case the relation between the angles and angular rates can be approximated as linear according to

\begin{equation}
\begin{bmatrix}
\dot{\phi} \\
\dot{\theta} \\
\dot{\psi}
\end{bmatrix}
\approx
\begin{bmatrix}
p \\
q \\
r
\end{bmatrix}.
\end{equation}

The body-fixed coordinate system and the angular rates of the quadcopter can be seen in Figure 2.1.


2.2 Modelling and Control of Quadcopter Angles

A quadcopter has four motors and rotors. A common configuration, which is used on the quadcopter in this thesis, is to have two motors rotating clockwise, and two motors rotating counter-clockwise. By having the motors on the diagonals rotating in the same direction it is possible to control the roll, pitch and yaw angles. The motors are numbered from 1 to 4, and each motor has its own separate motor signal, denoted $u_1$, $u_2$, $u_3$, and $u_4$ for each motor. The position and the direction of each motor can be seen in Figure 2.1. Each motor signal ranges from 0 to 1, where 0 denotes no power and 1 denotes max power. To control the angles and throttle, the control signals $u_{\text{throttle}}$, $u_{\text{roll}}$, $u_{\text{pitch}}$, and $u_{\text{yaw}}$ are used. Kugelberg (2016) implemented the relation between the motor signals and control signals as

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

(2.6)

In order to implement a fast and robust controller, the dynamics between the control signals and angular rates need to be known. Previously, work was done to estimate these relationships using black-box modelling. Time discrete models were estimated for the roll, pitch and yaw channels with the approximation that when the quadcopter is close to hovering, each channel only affects the cor-
responding rate and angle. The models for each channel, under the assumption that the quadcopter is close to hover and that (2.5) is valid, has the form

\[
\begin{align*}
A_{p}(z)p[k] &= -A_{1,p}(z)\phi[k] + B_{p}(z)u_{roll}[k] + C_{p}(z)e_p[k], \\
A_{\theta}(z)\theta[k] &= -A_{2,\theta}(z)q[k] + B_{\theta}(z)u_{pitch}[k] + C_{\theta}(z)e_\theta[k], \\
A_{q}(z)q[k] &= -A_{1,q}(z)\theta[k] + B_{q}(z)u_{pitch}[k] + C_{q}(z)e_q[k], \\
A_{r}(z)\psi[k] &= -A_{2,\psi}(z)r[k] + B_{\psi}(z)u_{yaw}[k] + C_{\psi}(z)e_\psi[k], \\
A_{r}(z)r[k] &= -A_{1,r}(z)\psi[k] + B_{r}(z)u_{yaw}[k] + C_{r}(z)e_r[k], \\
A_{x}(z) &= a_{x,0} + a_{x,1}z^{-1} + \ldots + a_{x,n}z^{-n}, \\
B_{x}(z) &= b_{x,1}z^{-1} + \ldots + b_{x,n}z^{-n}, \\
C_{x}(z) &= c_{x,0} + c_{x,1}z^{-1} + \ldots + c_{x,n}z^{-n}, \\
\end{align*}
\]

where \( e_x \) is noise. Equation 2.10 describes the shape of the polynomials in the black-box models. Using these models each channel could be simulated and PID controllers for the \( \phi \) and \( \theta \) angles and the rate \( r \) were determined.

### 2.3 Kalman Filter

If a linear state space model of a system is available, a Kalman filter can be used to obtain optimal estimates of the states from observations. A discrete model with the states \( x \), input signals \( u \) and observations \( y \) can be written as

\[
\begin{align*}
x[k + 1] &= F[k]x[k] + G_u[k]u[k] + G_v[k]v[k], \\
y[k] &= H[k]x[k] + D[k]u[k] + e[k], \\
\text{cov}(e[k]) &= R[k], \quad \text{cov}(v[k]) = Q[k],
\end{align*}
\]

where \( e \) is noise in the observations, originating from sensors, hardware and/or environmental disturbances among others. \( v \) is process noise, which is the magnitude of changes in the system that are not described by the model (Gustafsson, 2012). The matrices \( F, G_u, G_v, H \) and \( D \) can change over time, but in this thesis they are constant and will be written as such. The noise covariances \( R[k] \) and \( Q[k] \) are also assumed to be constant. Using this model, state estimates \( \hat{x}[k] \) and state covariances \( P[k] \) can be obtained. This is done in two steps, the time update that updates the state estimates according to the model, and the measurement update that updates the state estimates using observations (Gustafsson, 2012). The measurement update is simplified by first defining the innovation \( \epsilon \), the innovation
covariance $S$ and the Kalman gain $K$ as

$$
\epsilon[k] = y[k] - H\hat{x}[k|k-1] - Du[k],
$$
$$
S[k] = HP[k|k-1]H^T + R,
$$
$$
K[k] = P[k|k-1]H^TP^{-1}[k].
$$

The measurement update can then be calculated according to

$$
\hat{x}[k|k] = \hat{x}[k|k-1] + K[k]\epsilon[k],
$$
$$
P[k|k] = P[k|k-1] - K[k]HP[k|k-1].
$$

The time update can be expressed as

$$
\hat{x}[k+1|k] = F\hat{x}[k] + Gu[k],
$$
$$
P[k+1|k] = FP[k|k]F^T + GvQG^T.
$$

If the model is instead nonlinear, Gustafsson (2012) describes how an Extended Kalman Filter (EKF) can be used.

## 2.4 LQ Control

When controlling a system with a known, linear structure, an LQ controller can be implemented. According to Glad and Ljung (2006) and Lewis et al. (2012), an LQ controller minimises the quadratic cost $J$ by using the control law $u = -Kx$. For a continuous-time system with

$$
\dot{x} = Ax + Bu,
$$

the cost is

$$
J = \int_0^\infty (x^T Q x + u^T R u) dt,
$$

and for a discrete-time system with

$$
x[k+1] = Ax[k] + Bu[k],
$$

the cost is

$$
J = \sum_{k=0}^{\infty} (x^T Q x + u^T R u),
$$

assuming infinite time horizon and no mixed terms. $Q$ and $R$ are weight matrices, and are commonly diagonal matrices. The feedback gain $K$ that minimises $J$ is found by solving either the continuous-time algebraic Riccati equation or the discrete-time algebraic Riccati equation. The continuous-time algebraic Riccati equation is

$$
A^T P + PA - PBR^{-1}B^T P + Q = 0,
$$

(2.19)
and the discrete-time algebraic Riccati equation is
\[ A^T PA - PA^T PB(B^T PB + R)^{-1} B^T PA + Q = 0. \]  \hspace{1cm} (2.20)

For larger systems, the \( P \) that solves these equations is usually found numerically. For a continuous-time system the feedback gain that minimises \( J \) is
\[ K = R^{-1} B^T P, \]  \hspace{1cm} (2.21)

and for a discrete system it is
\[ K = (B^T PB + R)^{-1} B^T PA. \]  \hspace{1cm} (2.22)

(Glad and Ljung, 2006; Lewis et al., 2012).
The quadcopter platform in its current configuration can be seen in Figure 3.1. The positioning and control algorithms have been implemented on a computer, and this computer uses different methods and protocols to communicate with the remaining hardware. The hardware and the communication methods will be described in this chapter and the program architecture on the computer will also be explained.

Figure 3.1: Image of the quadcopter’s current configuration. The LIDAR can be seen in the left side of the image, and the GPS antenna and receiver antennas can be seen on top of the quadcopter. The blue PWM board can also be seen behind the LIDAR.
3.1 Hardware

The quadcopter consists of a carbon fibre frame with 4 motors, a battery and various electronic boards. The total mass of the quadcopter is 0.759 kilograms. The hardware currently installed is described in Table 3.1.

The components use different methods to communicate, described further in Figure 3.2.

3.2 Program Architecture

The programs running on the Raspberry are the central function, simply called main, and several smaller programs called daemons. The daemons run parallel to the central function and handle communication with external components. Table 3.2 lists the current daemons and Figure 3.3 shows how all programs interact.

Communication between programs is done using User Datagram Protocol (UDP). Reading from the UDP buffer is blocking, which means that a program attempting to read from the UDP buffer will pause until there is a message to read. This is used to set the frequency at which main runs. The IMU runs at a frequency of 100 Hz, and the EKF daemon reads from it using another protocol that is also blocking. As a result, main can only read data from the EKF daemon at a rate of 100 Hz, which is then the frequency of main.
Table 3.1: Summary of the hardware present on the quadcopter along with descriptions of their purpose and performance.

**Raspberry Pi**
The central computer that performs the majority of the calculations is a Raspberry Pi 3 model B. It is a microcomputer with a quad-core 1.2 GHz processor. It communicates with other modules using I2C, USB and serial communication (Raspberry Pi Foundation, 2017).

**ESC**
Each motor is powered and controlled by its own Electronic Speed Controller (ESC) that provides 3-phase alternating current. The ESC:s keeps each motor rotating at an RPM determined by the pulse width they receive from the PWM board.

**PWM board**
A PCA9685 PWM controller has been configured to send pulses to each ESC, with length ranging from about one to two milliseconds. A one millisecond pulse corresponds to the lowest speed, and two milliseconds to maximum. The PWM controller uses an internal 25 MHz clock to send these pulses at a frequency of about 400 Hz (NXP Semiconductors, 2015).

**Receiver**
To remotely control the quadcopter, an X8R receiver module on the quadcopter receives data transmitted from an X9D+ Taranis remote controller. The receiver has two antennas and receives data over eight 2.4 GHz channels (FrSky, 2015).

**Arduino**
The data from the receiver is transmitted using a serial port. Since the serial port on the Raspberry is connected to the IMU, the receiver is instead connected to an Arduino Micro. This Arduino is connected to the Raspberry by I2C, acting as slave. The Arduino also handles all analogue input signals.

**IMU**
The IMU on the quadcopter is an Xsens MTi-3. It is a board with a microcomputer and sensors, consisting of magnetometers, gyroscopes and accelerometers. These measure the magnetic field, angular rate and acceleration respectively, in three dimensions. The microcomputer can perform various filter algorithms, bias estimation and magnetometer calibration (Xsens, 2016).

**GPS**
An M8Q GPS module and an active GPS patch antenna has been attached to the quadcopter. The antenna is attached to the M8Q module with a coaxial cable, and the M8Q is attached to the Raspberry Pi using an adapter that converts the TTL-232RG serial communication to USB-protocol.

**LIDAR**
An SF30-B LIDAR is attached to the front of the quadcopter, pointing downwards. This allows for accurate, real time measurements of the height. The LIDAR is capable of outputting measurements with a resolution of three cm at 286 Hz from a micro-USB port, and even higher rates using serial communication. It also has the option for analogue output (LightWare Optoelectronics, 2016).
Figure 3.3: Flowchart describing the communication of programs on the Raspberry. Filled arrows show the progression of each program. Dashed lines show UDP communication, which is blocking.
### 3.3 LIDAR Measurements

Currently the USB and serial port on the LIDAR are damaged, so the analogue output port is used. The max range is currently set to 8 meters. The analogue to digital converter on the Arduino Micro outputs integers ranging from 0 to 1023, corresponding to a measured voltage from 0 to 5 V (Arduino, 2017). The SF30-B LIDAR in turn outputs an analogue value from 0 to 2.56 V corresponding linearly to a distance from 0 to 8 meters (LightWare Optoelectronics, 2016). Ideally, the conversion from integer to distance should then be

\[
\text{distance} = \frac{\text{integer} \cdot 8 \cdot 5}{1023 \cdot 2.56}. \quad (3.1)
\]
The models and controllers implemented by Kugelberg (2016) can successfully follow reference angles. However, there are still noticeable oscillations, primarily in the pitch controller, that would be advantageous to eliminate. Additionally, the step response of the models of the angular rates \( p \) and \( q \) resulted in oscillations towards infinity, which is not a realistic behaviour and results in unrealistic behaviours when simulating the complete model. This chapter will describe the work done to obtain models that better describe the actual behaviour of the system, and how these models were used to implement new controllers. The method used to determine a linear model to convert LIDAR measurements to distances will also be shown.

### 4.1 LIDAR Measurements

The ideal conversion from LIDAR measurement to distance is described in (3.1). However, there are several intermediate steps affecting the measurement received by the Raspberry that might affect the accuracy. To investigate these possible errors, the integer acquired by the analogue conversion in the Arduino was recorded at distances measured manually. The integers obtained at these distances can be seen in Figure 4.1. The distances calculated from these measurements using the theoretical method did not correspond well to the true distances. However, the measurements seem to be a linear function of the actual distance, so the parameters of a linear function was estimated by minimising the sum of square errors of

\[
\text{distance} = k \cdot \text{integer} + m. \tag{4.1}
\]

Using linear least squares results in \( k = 0.012686 \) and \( m = -0.093157 \). A new set of data was collected to validate the function, and Figure 4.2 shows the distances
calculated using the theoretical conversion and the linear function acquired using least squares.

![Analog conversion](image)

**Figure 4.1:** Integers obtained by connecting the LIDAR analogue port to the Arduino’s analogue to digital converter.

![LIDAR distances](image)

**Figure 4.2:** Distances calculated from LIDAR measurements using different linear parameters, plotted against the actual distance at which they were obtained. There appears to be a small, constant offset at distances after 2 meters.

### 4.2 Model Simplification

To be able to control the position of the quadcopter, fast control of the angles is required. The current PID controllers result in some oscillatory behaviour, so LQ controllers have been implemented using the models for the angular rates.
The time discrete black-box models in roll, pitch and yaw given in (2.7), (2.8) and (2.9) can be converted to continuous transfer functions on the form

\[
\begin{align*}
\phi(t) &= A_p \phi(s)p(t) + B_u \phi(s)u_{roll}(t) + C_e \phi(s)e_\phi(t), \\
p(t) &= A_p \phi(s)p(t) + B_u \phi(s)u_{roll}(t) + C_e \phi(s)e_p(t), \\
\theta(t) &= A_q \theta(s)q(t) + B_u \theta(s)u_{pitch}(t) + C_e \theta(s)\epsilon_\theta(t), \\
q(t) &= A_q \theta(s)q(t) + B_u \theta(s)u_{pitch}(t) + C_e \theta(s)e_q(t), \\
\psi(t) &= A_r \psi(s)r(t) + B_u \psi(s)u_{yaw}(t) + C_e \psi(s)e_\psi(t), \\
r(t) &= A_r \psi(s)r(t) + B_u \psi(s)u_{yaw}(t) + C_e \psi(s)e_r(t),
\end{align*}
\]  

(4.2)

(4.3)

(4.4)

where \(A_{x_2y}, B_{x_2y} \) and \(C_{x_2y} \) are transfer functions from \(x \) to \(y\), \(e_x \) is noise, and \(u_{roll}, u_{pitch} \) and \(u_{yaw} \) are the control signals. Kugelberg (2016) compared the black-box models to theoretical transfer functions. These theoretical transfer functions are given by

\[
\begin{align*}
\phi &= G_\phi(s)u_{roll}, & p &= G_p(s)u_{roll}, \\
G_\phi(s) &= \frac{15000}{s(s + 30)(s + 1)}, & G_p(s) &= \frac{15000}{(s + 30)(s + 1)}, \\
\theta &= G_\theta(s)u_{pitch}, & q &= G_q(s)u_{pitch}, \\
G_\theta(s) &= \frac{9000}{s(s + 30)(s + 1)}, & G_q(s) &= \frac{9000}{(s + 30)(s + 1)}, \\
\psi &= G_\psi(s)u_{yaw}, & r &= G_r(s)u_{yaw}, \\
G_\psi(s) &= \frac{750}{s(s + 30)(s + 0.7)}, & G_r(s) &= \frac{750}{(s + 30)(s + 0.7)}.
\end{align*}
\]  

(4.5)

(4.6)

(4.7)

Figure 4.3 shows the step responses of the black-box models and the theoretical transfer functions for a control signal with amplitude 0.1. The black-box model of \(r \) seems to have dynamics similar to its theoretical transfer function, while the black-box models for \(p \) and \(q \) result in oscillations with an infinitely increasing amplitude. Since the angles can be calculated from the angular rates according to (2.4), models for them are not required. Thus, only models for the angular rates are necessary, and according to the theoretical models the dynamics of the angular rates should not depend on the current angle. If this dependency on the angles is removed, and a step is done in the transfer function from control signal to angular rates, the response is stable and has an appearance that more resembles the theoretical response (see Figure 4.4).

The coefficients of the transfer functions from control signal to angular rate converted to continuous time can be seen in Table 4.1. These transfer functions have a large number of coefficients, and would result in a state space model with
many states that can not be measured directly. By creating simplified transfer functions containing only the roots and poles closest to the imaginary axis, an LQ controller can be created that controls only the dominating dynamics.

Table 4.1: The coefficients of the numerators and denominators of the continuous transfer functions from the motor signals to the angular rates \( p \), \( q \) and \( r \).

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Numerator</th>
<th>Denominator</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s^0 )</td>
<td>6.018 \cdot 10^7</td>
<td>2.844 \cdot 10^7</td>
</tr>
<tr>
<td>( s^1 )</td>
<td>3.309 \cdot 10^8</td>
<td>1.435 \cdot 10^6</td>
</tr>
<tr>
<td>( s^2 )</td>
<td>5.998 \cdot 10^6</td>
<td>1.252 \cdot 10^6</td>
</tr>
<tr>
<td>( s^3 )</td>
<td>4.329 \cdot 10^4</td>
<td>2.596 \cdot 10^4</td>
</tr>
<tr>
<td>( s^4 )</td>
<td>1.895 \cdot 10^2</td>
<td>7.247 \cdot 10^1</td>
</tr>
<tr>
<td>( s^5 )</td>
<td>0.000 \cdot 10^0</td>
<td>1.000 \cdot 10^0</td>
</tr>
</tbody>
</table>

The transfer functions for \( p \) and \( q \) have one root and a pair of poles close to the imaginary axis, while the transfer function for \( r \) has only a pole close to the imaginary axis. For \( p \) the root and poles are \(-0.1824\) and \(-0.3397 \pm 4.7907i\), and for \( q \) the root and poles are \(-0.2114\) and \(-0.1717 \pm 4.1681i\). Using the coefficients in Table 4.1, the static gain for \( p \), \( q \) and \( r \) can be calculated as 2.1154, 1.5584 and 26.9412, respectively. A transfer function with static gain \( g \), a root in \( r \), and poles in \( p_1 \) and \( p_2 \) can be calculated according to

\[
\frac{\frac{g}{r}s + g}{\frac{1}{p_1}p_2 s^2 + (\frac{1}{p_1} + \frac{1}{p_2})s + 1}.
\]
Figure 4.3: The step response acquired using theoretical and black-box models for $p$, $q$ and $r$, with a step amplitude of 0.1. The only black-box model with a convergent step response is $r$. 
A continuous time transfer function on the form

\[ y = \frac{a \cdot s + b}{c \cdot s^2 + d \cdot s + 1} u, \]  
(4.9)

can be written on state space form as

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{bmatrix} =
\begin{bmatrix}
-\frac{d}{c} & \frac{1}{c} \\
-1 & 0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
+ \begin{bmatrix}
a \\
c
\end{bmatrix} u,
\]

\[ y = x_1. \]  
(4.10)

This results in the state space models

\[
\begin{bmatrix}
\dot{\tilde{\alpha}} \\
\dot{\tilde{\beta}}
\end{bmatrix} =
\begin{bmatrix}
-0.6794 & 23.0661 \\
-1 & 0
\end{bmatrix}
\begin{bmatrix}
\tilde{\alpha} \\
\tilde{\beta}
\end{bmatrix}
+ \begin{bmatrix}
267.4527 \\
2.1154
\end{bmatrix} u_{roll},
\]

\[ \alpha = \int_0^t (2.1154u_p - p) dt, \]  
(4.11)

\[
\begin{bmatrix}
\dot{\tilde{\gamma}} \\
\dot{\tilde{\delta}}
\end{bmatrix} =
\begin{bmatrix}
-0.3433 & 17.4029 \\
-1 & 0
\end{bmatrix}
\begin{bmatrix}
\tilde{\gamma} \\
\tilde{\delta}
\end{bmatrix}
+ \begin{bmatrix}
128.2951 \\
1.5584
\end{bmatrix} u_{pitch},
\]

\[ \beta = \int_0^t (1.5584u_q - q) dt. \]  
(4.12)

where \( \alpha \) and \( \beta \) are states that are not measured. Using (4.11) and (4.12) \( \alpha \) and \( \beta \) can be expressed as

\[ \alpha = \int_0^t (2.1154u_p - p) dt, \]  
(4.13)

\[ \beta = \int_0^t (1.5584u_q - q) dt. \]  
(4.14)

The transfer function for \( r \) has a pole at -0.8929. A transfer function on the form

\[ y = \frac{a}{b \cdot s + 1} u, \]  
(4.15)

can be expressed on state space form as

\[ \dot{x} = \frac{1}{b} x + \frac{a}{b} u, \]  
(4.16)

resulting in the state space model

\[ \dot{r} = -0.8929 r + 26.9412 u_{yaw}. \]  
(4.17)

Figure 4.4 shows the step responses of the transfer functions from the black-box model and simplified state space models. The figure shows that they are nearly identical.
Figure 4.4: Response of a step with amplitude 0.1 in the motor control signal for both the original and the simplified models of $p$, $q$ and $r$ without angle dependencies. The models for $r$ have less oscillatory dynamics and higher gains than the models for $p$ and $q$. 
Assuming that the quadcopter is close to hovering, states for the Euler angles are implemented according to

\[
\begin{bmatrix}
\dot{\phi} \\
\dot{p} \\
\dot{\alpha}
\end{bmatrix} = 
\begin{bmatrix}
0 & 1 & 0 \\
0 & -0.6794 & 23.0661 \\
0 & -1 & 0
\end{bmatrix}
\begin{bmatrix}
\phi \\
p \\
\alpha
\end{bmatrix} + 
\begin{bmatrix}
0 \\
267.4527 \\
2.1154
\end{bmatrix} u_{\text{roll}},
\]

(4.18)

\[
\begin{bmatrix}
\dot{\theta} \\
\dot{q} \\
\dot{\beta}
\end{bmatrix} = 
\begin{bmatrix}
0 & 1 & 0 \\
0 & -0.3433 & 17.4029 \\
0 & -1 & 0
\end{bmatrix}
\begin{bmatrix}
\theta \\
q \\
\beta
\end{bmatrix} + 
\begin{bmatrix}
0 \\
128.2951 \\
1.5584
\end{bmatrix} u_{\text{pitch}},
\]

(4.19)

\[
\begin{bmatrix}
\dot{\psi} \\
\dot{r}
\end{bmatrix} = 
\begin{bmatrix}
0 & 1 & 0 \\
0 & -0.8929 & 0
\end{bmatrix}
\begin{bmatrix}
\psi \\
r
\end{bmatrix} + 
\begin{bmatrix}
0 \\
26.9412
\end{bmatrix} u_{\text{yaw}}.
\]

(4.20)

Various LQ controllers for \([\phi \ p]\) and \([\theta \ q]\) were calculated by solving the continuous-time algebraic Riccati equation (2.19). However, none of the generated LQ gains were able to stabilise control of the angles during practical tests, and the quadcopter was unable to get airborne. Hence, the old PID controllers have been kept for roll and pitch.

When implementing the LQ controller for \([\psi \ r]\), an additional state representing the integration of the error, \(\int_0^t \psi \, dt = \Psi\) has been added, resulting in the state space model

\[
\begin{bmatrix}
\dot{\Psi} \\
\dot{\psi} \\
\dot{r}
\end{bmatrix} = 
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & -0.8929
\end{bmatrix}
\begin{bmatrix}
\Psi \\
\psi \\
r
\end{bmatrix} + 
\begin{bmatrix}
0 \\
26.9412
\end{bmatrix} u_{\text{yaw}}.
\]

(4.21)

To generate an LQ gain, (2.19) and (2.21) was used with the model in (4.21). The \(Q\) and \(R\) matrices

\[
Q = 
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0.1
\end{bmatrix}, \quad R = 10,
\]

(4.22)

were tested, resulting in the LQ controller

\[
K_r = 
\begin{bmatrix}
0.3162 & 0.4939 & 0.1903
\end{bmatrix}, \quad u_{\text{yaw}} = -K_r \begin{bmatrix}
\Psi - \Psi_{\text{ref}} \\
\psi - \psi_{\text{ref}}
\end{bmatrix},
\]

(4.23)

where \(\psi_{\text{ref}}\) is the reference for \(\psi\) and \(\Psi_{\text{ref}} = \int_0^t \psi_{\text{ref}} \, dt\). The Bode Diagram for this controller can be seen in Figure 4.5.

The phase margin \(\phi_m\) is 72.258 degrees, and the cross-over frequency is 5.36 rad/s. This LQ controller resulted in stable control of the yaw angle, and has replaced the previous PI controller. The measured yaw angle and reference during a test flight can be seen in Figure 4.6.
4.2 Model Simplification

Figure 4.5: Bode Diagram of the open loop from $\psi_{ref}$ to $\psi$.

Figure 4.6: Plot of the measured yaw angle when following a reference yaw angle using the LQ controller.
4.3 Simulation Environment

In order to test algorithms and controllers, a complete model of the quadcopter has been created in Simulink. It contains all models for the angular rates and determines the Euler angles using integration and the rotation matrix for angular rates according to (2.4). Using a quadratic model of drag from Ljung and Glad (2004) and Newtons second law, the position $p$, velocity $v$ and acceleration $a$ in the global coordinates $[x y z]$ are simulated according to

$$a_x = \frac{T_x - \alpha v_x^2}{m}, \quad v_x = \int_0^t a_x \, dt, \quad p_x = \int_0^t v_x \, dt,$$

$$a_y = \frac{T_y - \alpha v_y^2}{m}, \quad v_y = \int_0^t a_y \, dt, \quad p_y = \int_0^t v_y \, dt,$$

$$a_z = \frac{T_z + mg - \alpha v_z^2}{m}, \quad v_z = \int_0^t a_z \, dt, \quad p_z = \int_0^t v_z \, dt,$$ (4.24)

where $\alpha$ describes the drag and $m$ is the mass of the quadcopter. The thrust $T_x$, $T_y$ and $T_z$ are the amounts of the quadcopter’s thrust parallel to the different coordinate vectors. The angular velocity of the motors has a rise time due to inertia and drag. This rise time has been estimated as approximately 33 ms by SAAB Dynamics and is included in the simulation. The angular rates $p$, $q$ and $r$ are calculated using the previously determined black-box models. The Euler angles are not calculated using black-box models, but are instead calculated as the integration of the rotated rates according to (2.4). Using this Simulink structure, different quadcopter models and controllers can be tested and evaluated.
Estimation of the quadcopter’s states requires the implementation of sensor fusion algorithms. Horizontal position estimation is done using an EKF program that can access the IMU and GPS to estimate the quadcopter’s states. However, this EKF does not support the use of a LIDAR, so height estimation using LIDAR measurements has been implemented. Additionally, the magnetometers on the IMU have been calibrated to compensate for external disturbances.

5.1 Extended Kalman Filter

The EKF available is a program written in C that uses measurements from an IMU and optionally a GPS to estimate Euler angles and angular rates along with position, speed, and acceleration in three dimensions. The position format is latitude and longitude in radians along with altitude. The output angles are Euler angles, and the angular rates, velocities and accelerations are given in the quadcopter’s body-fixed coordinate system.

5.2 Height Estimation

Measurements of the quadcopter’s height can be done using the accelerometers and the LIDAR. Both these measurements require knowledge of the current orientation of the quadcopter to acquire the vertical components of the measured acceleration and distance. Assuming that vertical velocities are small enough that drag can be neglected, all dynamics are linear and a Kalman filter can be used to estimate vertical velocity and smooth the height measurements. This Kalman filter uses the LIDAR measurements and the EKF output. This means that the acceleration has already been subject to filtering, and if the EKF is not fast
enough the time delays could result in an unstable system.

In Figure 5.1, the raw acceleration is compared to the difference between raw and filtered acceleration, and the cross-covariance of the raw and filtered acceleration can be seen in Figure 5.2. The raw accelerometer data is not free acceleration since it includes the gravitational acceleration. To obtain comparable signals, the x-component of the rotated gravitational acceleration has been subtracted from the raw measurements.

![Figure 5.1: The plot to the left shows the raw acceleration measured during a test flight, and the plot to the right shows the difference between the raw and filtered acceleration. As the difference is orders of magnitude smaller than the measurements, the raw and estimated acceleration are almost identical.](image)

As the difference between raw and filtered acceleration is small, and the cross-covariance has a dominating peak at 0 lags, it seems that time delays caused by the filter are shorter than the sample time of the controller. Hence, it is safe to use this filtered signal. Additionally, the EKF estimates the bias for all accelerometers which makes the measurements more accurate.

### 5.3 Height Kalman Filter

In order to acquire an estimate of the vertical velocity \( v \), a smoothed estimate of the height \( h \) and an estimate of the thrust required to hover \( u_0 \), a Kalman filter was implemented using measurements from the accelerometers and the LIDAR. A state space model has been derived, and a noise analysis has been done to estimate the noise covariance.
5.3 Height Kalman Filter

Figure 5.2: The covariance between filtered and raw acceleration, showing a clear peak at 0 lags.

5.3.1 Model Structure of Height

The relation between the acceleration $a$, velocity $v$ and height $h$ is linear. The change in acceleration is considered to be process noise. Measurements for acceleration and height are available. A continuous model can be written as

$$
\begin{align*}
\dot{x}(t) &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} v(t), \\
y(t) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} e_1(t) \\ e_2(t) \end{bmatrix},
\end{align*}
$$

(5.1)

where $e_1(t)$ is measurement noise in the LIDAR, $e_2(t)$ is measurement noise in the accelerometers and $v(t)$ is process noise. Transforming this model to discrete time results in

$$
\begin{align*}
\dot{x}[k + 1] &= \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & \frac{T}{2} \\ 0 & 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} \frac{T^3}{2} \\ \frac{T^2}{2} \\ T \end{bmatrix} v[k], \\
y[k] &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} e_1[k] \\ e_2[k] \end{bmatrix},
\end{align*}
$$

(5.2)

where $e_1[k]$ is measurement noise in the LIDAR, $e_2[k]$ is measurement noise in the accelerometers and $v[k]$ is process noise (Gustafsson, 2012). The distance and acceleration measurements are in the quadcopter’s local coordinates, and are rotated into the global coordinate system. Hence, the noise of the measurements depend on the current orientation and will vary with time. For the sake of sim-
plicity, it is assumed that the influence of the angles is negligible and that the noise can be approximated as Gaussian.

The LiDAR is located 0.122 meters in front of the centre of the IMU. This distance and the current orientation of the quadcopter needs to be considered when calculating the height. Using the rotation matrix $\mathcal{R}$ defined in (2.2), the unit vectors of the quadcopter’s body-fixed coordinate system $[X_Q Y_Q Z_Q]$ and the unit vectors of the global coordinate system $[x_g y_g z_g]$ can be expressed as

$$
X_Q = \mathcal{R}x_g, \quad Y_Q = \mathcal{R}y_g, \quad Z_Q = \mathcal{R}z_g.
$$

The height $h$ can be calculated from the LiDAR distance $d$ as

$$
h = d(Z_Q \cdot z_g) + 0.122(X_Q \cdot z_g),
$$

where the scalar product is used to determine the cosine of the angle between the vectors. The vertical acceleration $a$ in global coordinates is similarly acquired by rotating the measurements from the quadcopter according to

$$
a_Q = \begin{bmatrix} a_X & a_Y & a_Z \end{bmatrix}^T,
\quad
a = -\mathcal{R}a_Q \cdot z_g,
$$

where $a_X$, $a_Y$ and $a_Z$ are accelerometer measurements in the quadcopter. The sign is inverted since quadcopter acceleration is measured in an NED system and the height is measured upwards from the ground.

### 5.3.2 Static Thrust Estimation

The LiPo-battery currently used can power the quadcopter for approximately ten minutes during flight. It was discovered that as the battery is depleted, the thrust signal necessary for constant height, and thus the point around which the height controller should operate, increases. This static thrust is also different for different batteries, and will change if weight is added or removed from the quadcopter. While an integrating action in the controller avoids stationary height errors, it is more efficient to constantly be aware of the current operating point. Hence, a state has been included in the Kalman filter that estimates the current static thrust.

There are various methods to estimate the thrust generated from the motors. The thrust can be expressed as proportional to the square of the motor angular velocity, or as a function of various factors such as relative air velocity (Luukkonen, 2011; Spakovszky, 2007). Previous models of the quadcopter platform by Kugelberg (2016) assumed that the thrust was proportional to the motor signal, implying that the square of the RPM is proportional to the motor signal. This assumption will be used in this thesis as well. At motor signal 0 the total thrust is also 0, and at motor signal $u_0$ it is $mg$. Thus the relation between the throttle
control channel $u_{throttle}$ and the force from the motors $F_{Thrust}$ can be written as

$$F_{Thrust} = \frac{u_{throttle}mg}{u_0}.$$  (5.6)

The vertical acceleration $a$ in global coordinates can be expressed as

$$a = \frac{mg - F_{Thrust}cc_\phi c_\theta}{m} = g(1 - \frac{u_{throttle}cc_\phi c_\theta}{u_0})$$  (5.7)

and $u_0$ can be expressed as

$$u_0 = \frac{u_{throttle}cc_\phi c_\theta}{1 - \frac{a}{g}}.$$  (5.8)

This expression will only be useful as long as there is some amount of thrust directed upwards. If this is not the case, software fail-safes have been included to prevent the update of the static thrust state in the measurement update. However, being in a free fall is not a situation that should occur during normal operation. The progression of this state during a test flight, initialised with value 0.75 and uncertainty 0.001, can be seen in Figure 5.3.

**Figure 5.3:** Measurements of static thrust signal compared to the filtered estimate during a test flight.
Using the model in (5.2) and the measurements in (5.4), (5.5) and (5.8), the complete model used in the Kalman filter is

\[
x = \begin{bmatrix} h & v & a & u_0 \end{bmatrix}^T,
\]

\[
y = \begin{bmatrix} d(Z_Q \cdot z_g) + 0.122(X_Q \cdot z_g) \\
-z_g \cdot R a_Q \\
u_{thrust}^c \phi \theta \\
1 \end{bmatrix},
\]

\[
x[k + 1] = \begin{bmatrix} 1 & T & T^2 & 0 \\
0 & 1 & T & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} T^3 \\
T^2 \\
T \\
0 \end{bmatrix} \begin{bmatrix} v_1[k] \\
v_2[k] \\
v_3[k] \\
0 \end{bmatrix},
\]

(5.9)

The update of the static thrust is conditional. Close to the ground the quadcopter is subject to what is called the ground effect, which increases the lift (Danjun et al., 2015). Hence, the first condition is that the quadcopter needs to be more than 0.3 meters above the ground. The second condition, to avoid division by 0 is that \(a \neq g\). Finally, the quadcopter needs to be close to hovering, so the last condition is that the angles and angular rates in pitch and roll are within certain intervals. These intervals are \(|\phi| < 0.2, |\theta| < 0.2, |p| < 0.5, |q| < 0.5\). If these conditions are not met, the measurement of \(u_0\) and the final row in \(H\) is set to 0.

### 5.3.3 Noise Estimation

To acquire the covariance matrix \(R\), the covariance of the measurements were estimated individually. When the quadcopter is in flight, the variations in height means that the quadcopter does not maintain a height constant enough to estimate the noise covariance. Hence, the noise estimation in height measurements was done while the quadcopter was stationary on the ground. The data used to estimate the measurement noise in acceleration and static thrust was however done in flight. The static thrust can only be calculated while the quadcopter is in flight, and vibrations from the motors will result in noise that is only present during flight.

A segment of data from when the quadcopter remained stable for about 14 seconds was used to estimate noise covariance in acceleration and static thrust, and 50 seconds of recorded data was used to estimate the noise covariance in the height measurement. The autocovariance of these segments was estimated to determine if the noise was coloured or white. Histograms of the segments were used to determine whether the noise is Gaussian. The data can be seen in Figure 5.4, the autocovariance of the data can be seen in Figure 5.5, and the histograms can be seen in Figure 5.6.
Figure 5.4: The data used to estimate measurement noise covariances. Some oscillatory elements are present in the acceleration and $u_0$ measurements.
Figure 5.5: The autocovariance of the data used to estimate measurement noise covariances.
According to Gustafsson et al. (2001), the autocovariance of white noise should be 0 except for a peak at 0 lags. The autocovariance of the height measurements is noise with low variance except for a peak at 0 lags, and thus the noise of the height measurements is said to be white. In the autocovariances of the acceleration and $u_0$ measurements however there are recurring peaks, suggesting that the noise is coloured. However, as the peaks at 0 lags dominate over the recurring peaks, the noise in these measurements is still approximated as white. The noise
covariances achieved from the estimations are

\[
\begin{align*}
\text{cov}(e_1) &= 3.7268 \cdot 10^{-5}, \quad \text{cov}(e_2) = 2.8153 \cdot 10^{-1}, \quad \text{cov}(e_3) = 1.6978 \cdot 10^{-3}. \\
\end{align*}
\] (5.10)

Comparing the histograms with bell curves calculated using the covariances and means of the signals shows that the noise in height measurements closely resembles Gaussian noise, and the noise in acceleration and \(u_0\) is approximately Gaussian. Thus, the noise in height measurements is expected to be possible to fairly accurately be approximated as white Gaussian noise.

The calculated covariances are used as the diagonal elements in the \(R\) matrix of the Kalman filter, described in (2.11). The covariance of the process noise was not determined through measurements, but was instead selected as values that resulted in good state estimates during post processing of measurements. Setting the diagonal values of the \(Q\) matrix in the Kalman filter in (2.11) to \([10 \ 0.00001]\) results in satisfactory estimations.

### 5.4 Magnetometer Calibration

To obtain an estimate of the yaw angle that does not drift over time, the magnetometers on the IMU are used. These are however sensitive to disturbances from magnetic alloys and magnetic fields from wires or electromagnets (Konvalin, 2009). The magnetometers can be calibrated to compensate for these disturbances, as long as the disturbances remain stationary relative to the magnetometers. The magnetometers will still be affected by disturbances not located on the quadcopter, but these effects are negligible when flying in an open field.

To calibrate the magnetometers, a program on the IMU called Inflight Compass Calibration (ICC) is used (Xsens, 2016). The EKF program has been modified in order to activate this function when the GPS is not active. The ICC gathers data from the magnetometers for 100 seconds. According to the Xsens Knowledge Base (2017), the magnetometers should be moved through as many orientations as possible during this time for accurate calibration. After 100 seconds the ICC will calculate calibration parameters which are then saved on the IMU.

Currently the calibration is performed by a user holding the quadcopter by hand, in order to reach more extreme angles. The motors generate a noticeable magnetic field while running, so the calibration procedure is done while the motors are running at an RPM close to what is required to hover. However, the motors require less power and generate a weaker magnetic field without propellers attached. Therefore, four plastic bars with shape and weight similar to the propellers are attached during calibration as a safer alternative than using the actual propellers (see Figure 5.7). With these mock-up propellers and the motors running, the conditions during calibration are similar to the conditions during flight.
5.4 Magnetometer Calibration

To analyse the result of a calibration, the magnetic norm can be studied. Ideally, the norm of the magnetic measurements should always be 1 regardless of orientation (Xsens Knowledge Base, 2017). If the magnetometers are correctly calibrated, the norm should not deviate substantially from 1 when moving the quadcopter through various orientations. The norm of the magnetometers while varying the quadcopter orientation before and after calibration can be seen in Figure 5.8.

![Figure 5.7: A comparison of an actual propeller and the substitute used during calibration.](image)

![Figure 5.8: The magnetic norm while changing the quadcopter orientation before and after calibration. The calibration takes effect at around 108.47 seconds.](image)

The figure shows the magnetometer norm while the motors are running, both before and after calibration. It can be seen that there is an offset in the magnetometer norm before and after calibration. However, after calibration the magnitude of this offset is reduced, and the variation of the norm when altering the orientation has been significantly reduced. A side effect is that the magnetometers are
improperly calibrated when the motors are off and may result in incorrect heading estimation when the motors are inactive. However, as long as the heading is estimated accurately while the quadcopter is in flight the performance should not be affected.
The control of the position has been divided into two parts, horizontal and vertical. To control the vertical position, a PID controller and an LQ controller have been implemented. For horizontal movement, a position controller that maintains constant position and a trajectory controller that maintains constant velocity have been implemented. The vertical controllers generate a control signal for thrust, and the horizontal controllers generate reference angles used by the angle controllers. The estimates used by the controllers and the control signals they generate can be seen in Figure 6.1.

6.1 Height Control

To maintain a height reference, two different controllers have been implemented: one LQ controller and one PID controller. Both controllers use two feed-forward components: the static thrust and the angle of the quadcopter. Feed-forwarding the static thrust means that the controllers do not have to compensate for the gravitational acceleration and only control deviations from the reference height. A larger roll and/or pitch angle will require more thrust to stay airborne, and feed-forwarding this angle means the controllers will not have to compensate as much for this.

The signal generated by the controllers, $u_{\text{controller}}$, is added to the signal that maintains constant acceleration, $u_0$. The throttle signal $u_{throttle}$ is then generated by compensating for the current roll and pitch angles according to

$$u_{throttle} = \frac{u_0 + u_{\text{controller}}}{z_g \cdot Z_Q},$$

where $z_g$ and $Z_Q$ are described by (5.3).
6.1.1 PID

By using the height and vertical velocity estimated by the Kalman filter given by (5.9), a PID controller for the height has been implemented according to

\[ e = h_{\text{ref}} - h, \]  
\[ u_{\text{PID}} = K_p e + K_I \int_0^t (e) \, dt - K_D v. \]

To prevent integrator wind-up, conditional integration is used. The condition for integration is that \( u_{\text{throttle}} \) must be in the interval \( u_0 \pm 0.25 \). This interval was chosen since \( u_0 \approx 0.75 \) and \( u_{\text{throttle}} \) is limited to \([0, 1]\), and the interval where integration is allowed should be approximately symmetrical around \( u_0 \) in order to handle positive and negative height deviations equally. To determine the sta-
bility of the system, the relation between \( u_{throttle} \) and the height \( h \) has been determined. The following calculations are done assuming that \( \phi \) and \( \theta \) are small, and that the angle feed-forward will handle angular deviations. Assuming that the thrust from the motors is a linear function of the control signal, the thrust force is 
\[
F_{\text{throttle}} = K_t u_{\text{throttle}},
\]
for some parameter \( K_t \). When hovering, \( u_{\text{throttle}} = u_0 \), and 
\[
F_{\text{throttle}} = mg.
\]
Hence,
\[
K_t u_0 = mg \quad \Rightarrow \quad K_t = \frac{mg}{u_0} \quad (6.4)
\]
Also, using (6.1) and \( F = ma \),
\[
a = \frac{F}{m} = \frac{K_t u_{\text{throttle}} - mg}{m} = \frac{K_t u_0 + K_t u_{\text{PID}} - mg}{m} = \frac{K_t u_{\text{PID}}}{m} = \frac{u_{\text{PID}} g}{u_0}. \quad (6.5)
\]
Using \( u_{\text{PID}} \) described in (6.3), a differential equation can be written as
\[
a = \ddot{h} = \frac{g}{u_0}(K_p (h_{\text{ref}} - h) + \int_0^t (h_{\text{ref}} - h) dt - K_D \dot{h}) \implies \]
\[
\frac{u_0}{g} \ddot{h} + K_p h + K_I \int_0^t h dt + K_D \dot{h} = K_p h_{\text{ref}} + K_I \int_0^t h_{\text{ref}} dt. \quad (6.6)
\]
This can be rewritten into a transfer function according to
\[
\left( \frac{u_0}{g} s^3 + K_D s^2 + K_p s + K_I \right) h = (K_p s + K_I) h_{\text{ref}} \implies \]
\[
h = \frac{K_p s + K_I}{\frac{u_0}{g} s^3 + K_D s^2 + K_p s + K_I} h_{\text{ref}}. \quad (6.7)
\]
\( u_0 \) is not constant but is commonly in the interval of 0.7-0.8. The transfer function in (6.7) can be used to determine the roots and poles of the closed loop system, as well as simulating a step response for various values of \( u_0 \). Various parameters were tested, and the simulated step response with the values \( K_p = 0.2 \), \( K_I = 0.05 \) and \( K_D = 0.2 \) resulted in a step response that was deemed sufficiently fast with acceptable overshoot. Step responses for different values of \( u_0 \) can be seen in Figure 6.3, and Bode Diagrams can be seen in Figure 6.2.
Figure 6.2: The Bode Diagram of the PID controller for different values of $u_0$. As $u_0$ increases, the phase is constant but the magnitude decreases, decreasing the phase margin.

Figure 6.3: The simulated step response after a step in the reference height using a PID controller for different values of $u_0$. As $u_0$ increases, the settling time decreases but the overshoot increases.

The phase margin $\phi_m$ and cross-over frequency $\omega_c$ for different values of $u_0$ can be seen in Table 6.1. While the phase margin decreases as $u_0$ increases, the size
of the phase margin still allows for sufficient tolerance against time delays (Glad and Ljung, 2006).

Table 6.1: The phase margins and cross-over frequencies for different values of $u_0$. As $u_0$ increases, $\phi_m$ and $\omega_c$ decreases.

<table>
<thead>
<tr>
<th>$u_0$</th>
<th>$\phi_m$ [deg]</th>
<th>$\omega_c$ [Rad/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>72.99</td>
<td>3.34</td>
</tr>
<tr>
<td>0.75</td>
<td>69.06</td>
<td>2.71</td>
</tr>
<tr>
<td>0.9</td>
<td>65.31</td>
<td>2.29</td>
</tr>
</tbody>
</table>

The result of a test flight using these parameters can be seen in Figure 6.4.

Figure 6.4: The measured height while following a reference using a PID controller. The angles were controlled manually to maintain a constant position.

The PID controller successfully follows a reference height, although it suffers from a large overshoot at takeoff. This may be caused by the ground effect resulting in unexpectedly powerful lift before the quadcopter has cleared the ground.

6.1.2 LQR

To create an LQ controller of the height, a state space model is required. In the model, drag is neglected and the input is a force instead of a control signal. The reason for this is that since $u_0$ is not constant, the relation between the states and control signal is not linear. However, the relation between the states and the generated force is linear. The model is given by

$$
\begin{bmatrix}
\dot{I}_h \\
\dot{h} \\
\dot{v}
\end{bmatrix}
= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}
\begin{bmatrix}
I_h \\
h \\
v
\end{bmatrix}
+ \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m} \end{bmatrix} F_d,
$$

(6.8)
where $F_d$ is the deviation from the hovering thrust and $I_h$ is the integral of $h$. An LQR feedback gain $K_{F_d}$ can be calculated after selecting suitable $Q$ and $R$ matrices according to Section 2.4. Using (6.4), $u_{LQR}$ can be calculated as

$$F_d = u_{LQR}mg u_0 \Rightarrow u_{LQR} = \frac{F_d u_0}{mg}, \quad (6.9)$$

which gives that

$$u_{LQR} = -K_{LQR} \begin{bmatrix} I_h & h & v \end{bmatrix} = -\frac{u_0}{mg} K_{F_d} \begin{bmatrix} I_h & h & v \end{bmatrix}. \quad (6.10)$$

To find a suitable gain matrix, the step response for different matrices $R$ and $Q$ was simulated. Since the control signal is generated using feedback from the velocity, height and integration of the height, the resulting LQ controller is basically a PID controller with $K_P = K_{LQR}(2)$, $K_I = K_{LQR}(1)$ and $K_D = K_{LQR}(3)$. Therefore, the transfer function given by (6.7) can be used to simulate the step response. Since $K_{LQR}$ is proportional to $u_0$, the step response will be the same regardless of the value of $u_0$.

Using $Q$ with diagonal elements $[5 \ 100 \ 10]$ and $R = 5$ resulted in the gain matrix $K_{F_d} = [1 \ 5.1239 \ 3.1270]^T$, and the step response in Figure 6.5 was acquired. The accompanying Bode Diagram can be seen in Figure 6.6. The phase margin $\phi_m$ was 68.99 degrees and and the cross-over frequency $\omega_c$ was 4.34 rad/s, showing a resistance against time delays that does not vary with $u_0$.

![Simulated step in height reference](image)

**Figure 6.5:** The simulated step response after a step in the reference height using an LQ controller.
6.1 Height Control

**Figure 6.6:** The Bode Diagram for the LQ controller. The Bode Diagram and phase margin does not depend on $u_0$.

**Figure 6.7:** The measured height while following a reference using an LQ controller. The angles were controlled manually to maintain a constant position.

Comparing the simulated step responses from the LQ and PID controller, the LQ controller is faster and has a smaller overshoot. Although the LQ controller has a longer settling time, it is not affected by variations in $u_0$. Depending on $u_0$, 
the PID controller has a higher or lower phase margin than the LQ controller. The measured height and reference height during a test flight can be seen in Figure 6.7. Compared to the PID controller, the LQ controller seems to have less overshoot at takeoff, and after takeoff appears to be faster without overshooting.

6.2 Position Controller

Control of the quadcopter’s horizontal position is performed by calculating the reference angles necessary to direct the thrust required to steer the quadcopter towards the desired position. The measurements used are the quadcopter’s rotated velocity along with longitude and latitude measurements. The controllers interaction with the remaining system can be seen in Figure 6.1.

6.2.1 Assumptions

When developing the models and expressions for the quadcopter’s translational dynamics, a number of simplifications and assumptions have been made. It is assumed that the quadcopter always is close enough to its starting location that the small-angle approximation can be used to approximate longitude and latitude angles as Cartesian coordinates. It is also assumed that the quadcopter is moving slowly enough that drag can be neglected. To make calculations regarding the direction of the quadcopter’s thrust simpler, it is assumed that all thrust is directed along the quadcopter’s negative Z axis and that the magnitude of the thrust along the global negative z axis is \( mg \), where \( m \) is the mass of the quadcopter.

6.2.2 Local Map

Latitude and longitude measurements are received from the GPS. The latitude is the angle from the equator to the current position, and longitude is the angle from a latitude line passing through Greenwich, England to the current position (Ordnance Survey, 2016). The circle parallel to the equator at the current latitude is called a parallel. The circumference of the parallels decrease closer to the poles. While the latitude and longitude measurements are spherical coordinates, in a small enough area they can be approximated as Cartesian coordinates using the small-angle approximation. However, two things need to be taken into consideration when doing this approximation: the radius at the starting position and the relation between longitude and latitude.

The current radius is calculated using the starting latitude and longitude \( \text{LAT}_s \) and \( \text{LON}_s \). By assuming that the Earth is an ellipsoid with a polar radius \( r_{po} \) and equatorial radius \( r_{eq} \) of approximately 6356.8 km and 6378.1 km respectively, the current radius is calculated as

\[
r = \sqrt{(r_{eq} \cos(\text{LAT}_s))^2 + (r_{po} \sin(\text{LAT}_s))^2}.
\]  
\[(6.11)\]
The conversion from longitude to position depends on the current latitude. At higher latitudes an increase or decrease in longitude corresponds to a shorter distance than on low latitudes, since the parallel has a smaller circumference. By defining a coordinate system with origin at the quadcopter’s starting position, with the $x$ axis pointing north and the $y$ axis pointing east, the small-angle approximation can be used to convert the difference between the current latitude $\text{LAT}$ and $\text{LAT}_s$ as the $x$ position, and the difference between the current longitude $\text{LON}$ and $\text{LON}_s$ as the $y$ position. The radius of the parallel that $\text{LON}$ moves along is $r \cos(\text{LAT})$, so the quadcopter’s position can be expressed as

\begin{align}
\hat{p}_x &= r(\text{LAT} - \text{LAT}_s), \quad (6.12) \\
\hat{p}_y &= r \cos(\text{LAT})(\text{LON} - \text{LON}_s), \quad (6.13)
\end{align}

where $\text{LAT}$ and $\text{LON}$ are the current latitude and longitude measurements.

### 6.2.3 LQ Controller

The control of the position is done in two steps. First, LQ controllers are used to calculate the thrust in the $x$ and $y$ directions of the local map to maintain a position. Then, using the assumption that there is always a thrust force with magnitude $mg$ directed upwards in the local map, angles to direct the desired amount of thrust in the $x$ and $y$ directions can be calculated.

The state space models for horizontal motion, neglecting the effect of drag, are

\begin{align}
\begin{bmatrix}
\dot{p}_x \\
\dot{v}_x
\end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ v_x \end{bmatrix} + \begin{bmatrix} 0 \\ g \end{bmatrix} u_x, \\
\begin{bmatrix}
\dot{p}_y \\
\dot{v}_y
\end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_y \\ v_y \end{bmatrix} + \begin{bmatrix} 0 \\ g \end{bmatrix} u_y, \quad (6.14)
\end{align}

where $u_x$ and $u_y$ are thrust forces in the $x$ and $y$ direction respectively, normalised with the lift force so that $u = 1$ corresponds to a thrust force $mg$. Various LQ feedback gains were implemented according to (2.19) and (2.21), using the model in (6.14) and different $R$ and $Q$ matrices. After evaluating the performance of these controller it was determined that using a $Q$ matrix with diagonal elements $[1 \ 0.5]$ and $R = 20$ resulted in a controller with desirable behaviour. With these matrices, the LQ controllers are

\begin{align}
    u_x &= -\begin{bmatrix} 0.2236 & 0.2657 \end{bmatrix} \begin{bmatrix} p_x \\ v_x \end{bmatrix}, \\
    u_y &= -\begin{bmatrix} 0.2236 & 0.2657 \end{bmatrix} \begin{bmatrix} p_y \\ v_y \end{bmatrix}, \quad (6.15)
\end{align}
6.2.4 Reference Angle Calculations

Using the definitions in (2.1) and (2.2), rotating the quadcopter’s negative $Z$-axis, which is the direction of the thrust, into the global coordinate system yields

$$
\mathcal{R} \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -s_\psi s_\phi - c_\psi c_\phi s_\theta \\ c_\psi s_\phi - s_\psi c_\phi s_\theta \\ -c_\theta c_\phi \end{bmatrix} \implies \mathcal{R} \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -s_\psi s_\phi - c_\psi c_\phi s_\theta \\ c_\psi s_\phi - s_\psi c_\phi s_\theta \\ c_\theta c_\phi \end{bmatrix}.
$$

(6.16)

This means that if the normalised thrust of the quadcopter is \( \frac{1}{c_\theta c_\phi} \), the normalised thrust in the global coordinates \([x\ y\ z]\) is given by (6.16). Since the signals from the controller in (6.15) represent desired normalised thrust in \(x\) and \(y\), the angles \( \phi_{ref} \) and \( \theta_{ref} \) need to be calculated so that

$$
u_x = \frac{s_\psi s_\phi + c_\psi c_\phi s_\theta}{c_\theta c_\phi},
$$

(6.17)

$$
u_y = \frac{-c_\psi s_\phi + s_\psi c_\phi s_\theta}{c_\theta c_\phi}.
$$

(6.18)

The reference angle for \( \psi \) is controlled by a user with a remote controller, and current measurements of \( \psi \) are used to calculate \( \phi_{ref} \) and \( \theta_{ref} \).

The quadcopter has limited motor power available, so the assumption that there is always a force \( m g \) directed upwards in global coordinates is only valid within a certain interval. To ensure that the quadcopter does not attempt to keep itself at an angle where it can not maintain its height, \( K \) is defined as

$$
K = \min(K_{max}, u_x^2 + u_y^2),
$$

(6.19)

where \( K_{max} \) is a constant that will be determined later. To preserve the length and direction of the desired thrust, the following two conditions must be satisfied:

$$
\frac{u_x}{u_y} = \frac{s_\psi s_\phi + c_\psi c_\phi s_\theta}{-c_\psi s_\phi + s_\psi c_\phi s_\theta},
$$

(6.20)

$$
\left( \frac{s_\psi s_\phi + c_\psi c_\phi s_\theta}{c_\theta c_\phi} \right)^2 + \left( \frac{-c_\psi s_\phi + s_\psi c_\phi s_\theta}{c_\theta c_\phi} \right)^2 = K.
$$

(6.21)

It can be shown that

$$
\left( s_\psi s_\phi + c_\psi c_\phi s_\theta \right)^2 + \left( -c_\psi s_\phi + s_\psi c_\phi s_\theta \right)^2 = s_\psi^2 s_\phi^2 + 2s_\psi s_\phi s_\theta c_\phi c_\phi + c_\phi^2 c_\phi^2 c_\theta^2 + c_\phi^2 s_\phi^2 - 2s_\psi s_\phi s_\theta c_\psi c_\phi + s_\phi^2 c_\phi^2 s_\theta^2 =
$$

$$
(s_\phi^2 + c_\phi^2) s_\phi^2 + (s_\psi^2 + c_\psi^2) c_\phi^2 s_\theta^2 = s_\phi^2 + c_\phi^2 s_\theta^2,
$$

(6.22)
so (6.21) can be rewritten into

\[
\begin{align*}
    s_\phi^2 + c_\phi^2 s_\theta^2 &= Kc_\phi^2 c_\phi^2 = K \left(1 - s_\theta^2\right) c_\phi^2 = K \left(c_\phi^2 - s_\phi^2 c_\phi^2\right), \\
    s_\phi^2 c_\phi^2 (1 + K) &= K \left(1 - s_\phi^2\right) - s_\phi^2 \quad \Rightarrow \quad s_\phi^2 c_\phi^2 = \frac{K}{1 + K} - s_\phi^2.
\end{align*}
\]

(6.23)

Equation 6.20 can be rewritten as

\[
    c_\phi s_\theta \left(u_x s_\psi - u_y c_\psi\right) = s_\phi \left(u_y s_\psi + u_x c_\psi\right),
\]

(6.25)

and by squaring the left and right hand sides, (6.24) can be substituted into the left hand side, which yields

\[
\begin{align*}
    \left(\frac{K}{1 + K} - s_\phi^2\right) \left(u_x s_\psi - u_y c_\psi\right)^2 &= s_\phi^2 \left(u_y s_\psi + u_x c_\psi\right)^2, \\
    \frac{K}{1 + K} \left(u_x s_\psi - u_y c_\psi\right)^2 &= s_\phi^2 \left(u_x^2 + u_y^2\right).
\end{align*}
\]

(6.26)

which can be simplified and rewritten into

\[
    \frac{K}{1 + K} \left(u_x s_\psi - u_y c_\psi\right)^2 = s_\phi^2 \left(u_x^2 + u_y^2\right).
\]

(6.27)

With this, the reference angle \(\phi_{\text{ref}}\) can be calculated as

\[
    \phi_{\text{ref}} = \sin^{-1}\left(\frac{K}{\sqrt{(1 + K) \left(u_x^2 + u_y^2\right)}} \left(u_x s_\psi - u_y c_\psi\right)\right),
\]

(6.28)

unless \(u_x^2 + u_y^2 = 0\), in which case \(\phi_{\text{ref}}\) is set to 0. Equation 6.23 can be rewritten as

\[
    s_\phi^2 + c_\phi^2 (1 - c_\phi^2) = 1 - c_\phi^2 c_\phi^2 = Kc_\phi^2 c_\phi^2 \quad \Rightarrow \quad c_\phi^2 c_\phi^2 = \frac{1}{1 + K}.
\]

(6.29)

The angle \(\theta_{\text{ref}}\) can be calculated by using the calculated value of \(\phi_{\text{ref}}\) according to

\[
    s = \text{sign} \left(u_y s_\psi + u_x c_\psi\right),
\]

\[
    \theta_{\text{ref}} = s \cos^{-1} \left(\frac{1}{c_\phi \sqrt{1 + K}}\right).
\]

(6.30)

To determine the value of \(K_{\text{max}}\), the angle between the global z axis and the quadcopter’s Z axis is defined as \(\gamma\), and the max angle the position controller will output is defined as \(\gamma_{\text{max}}\). The normalised thrust vector is assumed to have magnitude 1 along the z-axis. When the angles are saturated, the normalised thrust vector will have absolute length \(T_{\text{max}}\), and the length of the normalised thrust vector in the [x y] plane will be \(T_{\text{max}} \sin (\gamma_{\text{max}})\). From Pythagoras theorem it follows that

\[
    T_{\text{max}}^2 = 1^2 + T_{\text{max}}^2 \sin^2 (\gamma_{\text{max}}) \quad \Rightarrow \quad T_{\text{max}}^2 = \frac{1}{1 - \sin^2 (\gamma_{\text{max}})}.
\]

(6.31)
meaning that $K_{\text{max}}$ should be

$$K_{\text{max}} = \frac{\sin^2 (\gamma_{\text{max}})}{1 - \sin^2 (\gamma_{\text{max}})}.$$  \hspace{1cm} (6.32)

An illustration of the forces can be seen in Figure 6.8.

![Illustration of the normalised forces at the maximum allowed angle $\gamma_{\text{max}}$.](image)

**Figure 6.8:** Illustration of the normalised forces at the maximum allowed angle $\gamma_{\text{max}}$.

As mentioned in Section 6.1.1, the static thrust signal $u_0$ is usually between 0.7 and 0.8. This means that the quadcopter needs about 70 to 80% of maximum thrust to maintain height. The maximum angle where this amount of thrust can be directed upwards is $\cos^{-1} (0.8) \approx 36.86^\circ$. The maximum angle allowed is currently set to 10° to allow for large safety margins in case of overshoot in the angle controllers, and to add robustness towards external influences on the angles.

The performance of the position controller during a step in reference position can be seen in Figure 6.9 and Figure 6.10. These figures show that using the position controller, the quadcopter can travel to the new position while avoiding overshoot. However, the lack of integral action makes the position controllers susceptible to stationary errors if external forces such as wind are present. As can be seen in Figure 6.10 the quadcopter does not reach the desired $y$ coordinates, suggesting that there is more wind in the $y$ direction than the $x$ direction.
6.3 Position Following

In order to allow an operator to control the position of the quadcopter, the LQ controllers have been modified such that they follow a reference point instead of
maintaining position $[0 \ 0]$, according to

$$u_x = -\begin{bmatrix} 0.2236 & 0.2657 \end{bmatrix} \begin{bmatrix} p_x - r_x \\ v_x \end{bmatrix},$$  \hspace{1cm} (6.33)

$$u_y = -\begin{bmatrix} 0.2236 & 0.2657 \end{bmatrix} \begin{bmatrix} p_y - r_y \\ v_y \end{bmatrix},$$  \hspace{1cm} (6.34)

$$u_z = -\begin{bmatrix} 0.2236 & 0.2657 \end{bmatrix} \begin{bmatrix} p_z - r_z \\ v_z \end{bmatrix},$$  \hspace{1cm} (6.35)

The reference position $[r_x \ r_y]$ is changed by using the left stick on the remote controller. The vertical position of the stick controls the speed of $r_x$, and the horizontal position of the stick controls the speed of $r_y$. The maximum speed of $r_x$ and $r_y$ is 1 m/s. The Bode Diagram of the reference following position controller can be seen in Figure 6.11.

![Bode Diagram Positioning](image)

**Figure 6.11**: Bode diagram of the the position controller with reference following. The phase margin is 72.868 degrees and the cross-over frequency is 2.73 rad/s.

The Bode Diagram shows that the phase margin is 72.87 degrees and the cross-over frequency is 2.73 rad/s.

The position controller can also follow a path by defining a list of coordinates. The quadcopter will move to each coordinate by setting it as the reference position, starting with the first coordinate in the list. When the quadcopter is within a certain distance of the current reference position the next coordinate in the list is
set as the reference position. To illustrate this, a path in a figure eight shape was created. The distance between coordinates was approximately 0.1 meters, and the distance where the reference position is updated was set to 1 meter. The measured position and velocity while following this path can be seen in Figure 6.12 and Figure 6.13, respectively.

**Figure 6.12**: The measured position while following reference points spaced approximately 0.1 meter apart, roughly shaped like a figure eight. The stationary errors suggest the presence of wind from the northwest.

The quadcopter manages to maintain a velocity that piecewise seems to be within 0.1 m/s of some constant velocity. The velocity can be increased or decreased by increasing or decreasing the threshold, as the LQ controller will generate a greater control signal when the destination is further away. The quadcopter mostly manages to stay within 0.5 meters of the path. Since there is no integral action, external forces such as wind will result in stationary errors.
6.4 Trajectory Control

While the position controller is capable of moving between predefined points it does so either with varying speed or requiring a large amount of predefined positions. To allow for greater control over how a mission is performed, a trajectory controller has been implemented. This trajectory controller moves between user defined destinations while maintaining a set reference speed.

6.4.1 Constant Velocity Along Line

The basic principle of the trajectory controller is similar to the position controller, with two LQ controllers determining the thrust in orthogonal directions. Unlike the position controller, these forces are not north-south and east-west, but instead parallel and perpendicular to a reference line. The first LQ controller in the trajectory controller maintains a constant velocity along the line, and the other controls the distance to this current line. The line is drawn between the previous and the current destination. The height controller still maintains the reference height of the current destination, while the trajectory controller follows the line in the horizontal directions. The continuous-time state space model for the movement along a trajectory can be expressed as

$$
\begin{bmatrix}
    \dot{p}_\perp(t) \\
    \dot{v}_\perp(t) \\
    \dot{v}_\parallel(t)
\end{bmatrix}
= 
\begin{bmatrix}
    0 & 1 & 0 \\
    0 & 0 & 0 \\
    0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
    p_\perp(t) \\
    v_\perp(t) \\
    v_\parallel(t)
\end{bmatrix}
+ 
\begin{bmatrix}
    0 & 0 \\
    g & 0 \\
    0 & g
\end{bmatrix}
\begin{bmatrix}
    u_\perp(t) \\
    u_\parallel(t)
\end{bmatrix},
$$

(6.36)

where $p_\perp(t)$ and $v_\perp(t)$ is the distance and velocity perpendicular to the line to be followed while $v_\parallel(t)$ is the velocity parallel to the line. The corresponding
discrete-time model is
\[
\begin{bmatrix}
p_\perp[k+1] \\
v_\perp[k+1] \\
v_\parallel[k+1] \\
a_\parallel[k+1]
\end{bmatrix} =
\begin{bmatrix}
1 & T & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
p_\perp[k] \\
v_\perp[k] \\
v_\parallel[k] \\
a_\parallel[k]
\end{bmatrix}
+ \begin{bmatrix}
gT^2/2 \\
gT \\
g \\
gT/2 \\
g
\end{bmatrix}
\begin{bmatrix}
u_\perp[k] \\
u_\parallel[k]
\end{bmatrix},
\tag{6.37}
\]
where \(p_\perp[k]\) and \(v_\perp[k]\) is the distance and velocity perpendicular to the line to be followed while \(v_\parallel[k]\) is the velocity parallel to the line and \(T\) is the sample time. Either of these models can be used to determine an LQ controller. To avoid implementing what is essentially a P-controller for the parallel velocity, measurements of the parallel acceleration \(a_\parallel\) is also used. The model in (6.37) was rewritten as
\[
\begin{bmatrix}
p_\perp[k+1] \\
v_\perp[k+1] \\
v_\parallel[k+1] \\
a_\parallel[k+1]
\end{bmatrix} =
\begin{bmatrix}
1 & T & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
p_\perp[k] \\
v_\perp[k] \\
v_\parallel[k] \\
a_\parallel[k]
\end{bmatrix}
+ \begin{bmatrix}
gT^2/2 \\
gT \\
g \\
gT/2 \\
g
\end{bmatrix}
\begin{bmatrix}
u_\perp[k] \\
u_\parallel[k]
\end{bmatrix}.
\tag{6.38}
\]
This means that the update of \(v_\parallel[k]\) is determined by the mean of the expected acceleration from \(u_\parallel[k]\) and the previous value of the acceleration \(a_\parallel[k]\), which can be measured.

Each destination is determined by coordinates in the global coordinate system. The line to be followed is defined by the previous destination \([d_{k-1,x}, d_{k-1,y}]\) and the current destination \([d_{k,x}, d_{k,y}]\). Using linear algebra, the unit vectors \(l_\parallel\) and \(l_\perp\) are calculated. \(l_\parallel\) describes the direction of the line in global coordinates, and \(l_\perp\) is the tangent of the line. They are calculated according to
\[
l_\parallel = \frac{1}{\|d_k - d_{k-1}\|} \begin{bmatrix}d_{k,x} - d_{k-1,x} \\ d_{k,y} - d_{k-1,y}\end{bmatrix},
\tag{6.39}
\]
\[
l_\perp = \frac{1}{\|d_k - d_{k-1}\|} \begin{bmatrix}-d_{k,y} + d_{k-1,y} \\ d_{k,x} - d_{k-1,x}\end{bmatrix}.
\tag{6.40}
\]
If \(d_k = d_{k-1}\) there is no horizontal line to follow, and the position controller is used to maintain the \([x, y]\) coordinates of the destination while the target height is reached with the height controller.

The acceleration and velocity parallel and perpendicular to the line can be calculated using these unit vectors and scalar products according to
\[
a_\parallel = l_\parallel \cdot \begin{bmatrix}a_x \\ a_y\end{bmatrix},
\tag{6.41}
\]
\[
v_\parallel = l_\parallel \cdot \begin{bmatrix}v_x \\ v_y\end{bmatrix},
\tag{6.42}
\]
\[
v_\perp = l_\perp \cdot \begin{bmatrix}v_x \\ v_y\end{bmatrix},
\tag{6.43}
\]
where the quadcopter’s velocity and acceleration in global coordinates is expressed with \(v_x, v_y\) and \(a_x, a_y\). The distance from the line \(p_\perp\) and the distance to the desti-
nation \( p_\parallel \) can be calculated as

\[
p_\perp = l_\perp \cdot \begin{bmatrix}
p_x - d_{k-1,x} \\
p_y - d_{k-1,y}
\end{bmatrix},
\]

(6.44)

\[
p_\parallel = l_\parallel \cdot \begin{bmatrix}
d_{k,x} - p_x \\
d_{k,y} - p_y
\end{bmatrix}.
\]

(6.45)

Using the model in (6.38), a \( Q \) matrix with diagonal elements \([1 0.5 5 1]\) and an \( R \) matrix with diagonal elements \([10 20]\), the LQ gain \( K_{\text{line}} \) was calculated as

\[
K_{\text{line}} = \begin{bmatrix}
0.2207 & 0.2634 & 0 & 0 \\
0 & 0 & 0.1301 & 0.065
\end{bmatrix},
\]

(6.46)

resulting in the control signals

\[
\begin{bmatrix}
u_\perp \\
u_\parallel
\end{bmatrix} = -K_{\text{line}} \begin{bmatrix}
p_\perp \\
v_\perp \\
v_\parallel - v_{\text{ref}} \\
a_\parallel
\end{bmatrix},
\]

(6.47)

where \( u_\perp[k] \) and \( u_\parallel[k] \) are the control signals perpendicular and parallel to the line, and \( v_{\text{ref}} \) is the reference velocity. These are expressed in the global coordinate system as

\[
\begin{bmatrix}
u_x \\
u_y
\end{bmatrix} = l_\perp u_\perp + l_\parallel u_\parallel.
\]

(6.48)

References for \( \phi \) and \( \theta \) can then be calculated using (6.28) and (6.30). If the line is horizontal the reference for \( \psi \) is set such that the heading of the quadcopter is parallel to the current line, according to

\[
\psi = \text{atan2}(l_\parallel, y, l_\parallel, x),
\]

(6.49)

where \( l_\parallel, x \) and \( l_\parallel, y \) are the \( x \) and \( y \) components of \( l_\parallel \), and \( \text{atan2} \) is the four quadrant inverse tangent that returns an angle in the interval \([−\pi \pi]\).

### 6.5 Missions

Each mission is defined by a matrix with four columns, and with one row for each destination the quadcopter shall visit. For each destination, these columns represent the \([x \ y \ z]\) coordinates of the destination and the reference velocity when moving to the destination. The quadcopter will move to all these coordinates at the reference speed given. When \( p_\parallel \leq 0 \) the destination is reached, and the current destination is updated. The matrix describing the mission for the quadcopter to perform is stored as a text file, which the quadcopter will load when booting. A test flight was performed using the mission described by Table 6.2.
Table 6.2: The coordinates and reference velocities when testing the trajectory controller.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-6</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-6</td>
<td>-6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>-6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1.5</td>
<td>2</td>
</tr>
<tr>
<td>-6</td>
<td>6</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>-6</td>
<td>-6</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>-6</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 6.14: Reference and measured velocity during the test flight. There is a significant and sudden decrease in velocity when the quadcopter reaches the current waypoint and performs a turn.

The measured position during the test flight can be seen in Figure 6.15 and the measured velocity can be seen in Figure 6.14. These figures show that the controller is capable of following a reference trajectory at a reference velocity. There is some position overshoot at breakpoints, which increases at higher velocities. There is a sharp decrease in speed when the quadcopter reaches its current destination.
Figure 6.15: Measured position during a test flight. The quadcopter started at [0 0], flew in a square twice and then returned to [0 0]. The deviation from the first segment of the trajectory may be a result of reduced control at takeoff.

6.5.1 Constant Velocity When Turning

In order to avoid overshoot at breakpoints, a predictive transition has been implemented. This predictive transition calculates the upcoming heading, and begins transitioning before the current destination has been reached.

At the distance $d_{\text{turn}}$ from the breakpoint a turn that takes $t_{\text{turn}}$ seconds to complete is initiated. During the turn, two sets of control signals $[u_x \ u_y]$ are calculated, $u_{\text{current}}$ and $u_{\text{next}}$. Both are computed using (6.47) and (6.48), with $u_{\text{current}}$ using the current line, and $u_{\text{next}}$ using the upcoming line. The weighted
control signal is

\[ u_{\text{weighted}} = (1 - W)u_{\text{current}} + W u_{\text{next}}, \]  
(6.50)

\[ \psi_{\text{weighted}} = (1 - W)\psi_{\text{current}} + W \psi_{\text{next}}, \]  
(6.51)

\[ W = \frac{t - t_{\text{start}}}{t_{\text{turn}}}, \]  
(6.52)

where \( t \) is the current time and \( t_{\text{start}} \) is the time the turn was started. \( W \) is defined in \([0 \; 1]\), and when \( t - t_{\text{start}} = t_{\text{turn}} \) the turn is completed and the current line to follow is updated.

For flexibility, the distance and time to turn depends on the size of the turn and the current velocity. The distance when the turn must be started depends on how much of the current velocity must be eliminated when the breakpoint is reached. The baseline for these values is the constant \( t_{90} \), which is the preferred time it should take to perform a 90 degree turn.

When performing a 90 degree turn with an initial reference velocity \( v_{\text{ref}} \), starting the turn at \( \frac{v_{\text{ref}}}{t_{90}} \) meters from the breakpoint should be enough to eliminate the velocity perpendicular to the new line when the breakpoint is reached. However, a smaller turn means that not all of the current velocity needs to be eliminated, and the time to turn is shorter. \( t_{\text{turn}} \) is approximated as being proportional to the angle of the turn \( \psi_{\text{turn}} \), hence

\[ t_{\text{turn}} = \frac{|t_{90} \psi_{\text{turn}}|}{\pi}, \]  
(6.53)

\[ d_{\text{turn}} = \frac{v_{\text{ref}}}{t_{\text{turn}}}. \]  
(6.54)

The value of \( t_{90} \) has been selected as one second. Using the controller in (6.47) and the model in (6.14), a flight with and without this algorithm could be simulated. \( t_{90} \) was set to one second, and \( v_{\text{ref}} \) was set to 2 m/s for all segments. The simulated paths can be seen in Figure 6.16, and the simulated velocities can be seen in Figure 6.17.
In this simulation, predictive turning completely eliminates overshoot. The velocity clearly diverges from the reference velocity at breakpoints, but this divergence is smaller and slower with predictive turning. As a result, the simulated mission is completed faster when using predictive turning, however the waypoints are not reached completely. Whether predictive or normal turning is preferred depends on if reaching the destination completely is more important than minimising deviations from the reference velocity.
Figure 6.17: Simulated absolute velocity while flying along a path with $v_{ref} = 2$ and $t_{90} = 1$, with and without predictive turning. The velocity deviation is decreased with predictive turning, resulting in faster mission completion.

Table 6.3: The coordinates and reference velocities when testing the predictive turning algorithm.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>-6</td>
<td>6</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>-6</td>
<td>-6</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>-6</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1.5</td>
<td>2</td>
</tr>
</tbody>
</table>

Using $t_{90} = 1$, a test flight was performed using the predictive turning algorithm and the mission in Table 6.3. The resulting trajectory can be seen in Figure 6.18, the velocity can be seen in Figure 6.19, and the yaw angle can be seen in Figure 6.20.

Figure 6.18 shows that the quadcopter starts turning when it is 2 meters from the next breakpoint, which is correct since $t_{90} = 1$, $v_{ref} = 2$ and all turns are 90 degrees. When the velocity exceeds the reference velocity there is some overshoot, but otherwise the overshoot is mostly or completely eliminated. Figure 6.19 shows that turning still results in velocity variations, and there seems to be some relation between the amplitude of the velocity variation and the size of the overshoot.

Figure 6.20 shows that the reference yaw angle is followed relatively well except during the last turn, although there is significant overshoot.
Comparing the simulated results and the test flights with and without predictive turning shows that predictive turning has less overshoot and velocity variations than ordinary turning, but more than the simulated results. When the overshoot is properly eliminated, the resulting trajectory and velocity variations are similar to the simulation, with the velocity decreasing with approximately 0.5 m/s. When the overshoot is not eliminated the amplitude of the overshoot seems to at least be reduced compared to normal turning.

It should be noted that during this test, the predictive turning algorithm appears to more efficiently reduce overshoot along the $y$ axis than the $x$ axis. Whether this is due to external influences or the implementation requires further investigation.

**Figure 6.18:** Trajectory of the quadcopter when testing the predictive turning algorithm. The trajectory is marked red when the predictive turning algorithm is active.
Figure 6.19: Velocity of the quadcopter when testing the predictive turning algorithm. At some breakpoints the velocity variations have been significantly reduced.

Figure 6.20: Yaw angle of the quadcopter when testing the predictive turning algorithm, showing how the transition at breakpoints is now a ramp instead of a step.
Conclusions

The implemented controllers have been tested both in simulations and in practical tests using the position estimations. This chapter will comment on the result of the algorithms, and the possibilities for future development.

7.1 Result

The goal of this thesis was to allow for a position controller to allow the quadcopter to perform missions. These missions are defined by a sequence of coordinates that the quadcopter will attempt to reach and velocities to follow when moving to the destinations. The implementation of the trajectory controller was successful, with the quadcopter being capable of maintaining the reference velocities and travelling to the defined coordinates. An algorithm to decrease position overshoot and improve velocity reference tracking was developed and showed promising results during simulations and practical tests. Both height controllers could follow a reference and the position controller could follow a step in the reference position and maintain it, but with a minor stationary error.

One problem when verifying the performance of the quadcopter is that there are no external measurements of the quadcopter's position available. If the quadcopter's position estimates drift over time while the quadcopter is in flight, the quadcopter will compensate for the resulting position errors by adjusting its position accordingly. As such, even though the quadcopter will move in the real world, the logs will show that it maintained its position, and was unaffected by the drifting errors. While data can be logged while the quadcopter is stationary, and an operator can visually detect abnormal position compensations from the quadcopter, drifts that occur during trajectory following can be difficult to detect without an external measurement to compare with.

Using the models estimated by Kugelberg (2016) it was possible to implement
an LQ controller for $\psi$, but attempts to do the same for $\phi$ or $\theta$ were unsuccessful. The reason for this may be that while data could be gathered for all angles of $\psi$, the angles of $\phi$ and $\theta$ needed to be in a small interval to maintain stability. This limitation on the data collection may have resulted in models that did not describe the dynamics accurately enough to create an LQ controller. It is also possible that the assumptions and simplifications made in this thesis are not valid, and that it is possible to implement a controller using a complete model.

7.2 Future Work

The ability to estimate and control the position of the quadcopter can be built upon further and be used in future experiments. New black-box models for angles and angular rates could be estimated that included the effect of the quadcopter’s horizontal velocities.

The positioning and trajectory controllers can be combined with different position estimation methods. An alternative for the future is to evaluate the performance indoors, using external cameras and image processors to generate position and orientation measurements.

Horizontal and vertical positioning estimation is done by separate programs. In the future it could be advantageous to integrate the LIDAR measurements into the EKF for better estimates.

Since the quadcopter no longer requires a user with a remote controller, implementing controllers for the angular rates instead of angles could result in improved navigational capabilities. This will however require fast controllers or the integration of the angular rate errors will result in significant angle deviations.

The current position controllers assume that there is always a force $mg$ directed upwards. While this is true in most cases, large changes in $u_{throttle}$ could affect the performance of the position controllers. Feedforwarding the current throttle signal could improve the precision of the position controllers. Additionally, the height controllers could be improved by determining a more accurate model of how the thrust depends on $u_{throttle}$.

An algorithm could be developed that, given a set of coordinates, determines the set of trajectories that results in, for example, the shortest path or fastest route required to visit all coordinates.

There is significant room for improvement in the predictive turning algorithm. By implementing an optimal control algorithm, with finite time horizon and final conditions on velocity and distance to the next line, the overshoot and velocity variations could be further decreased.


