Vision-Based Flight Control for a Quadrotor UAV

by

J.J. Rademeyer

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering in the Faculty of Engineering at Stellenbosch University

Study leaders:

Dr J.A.A. Engelbrecht
Prof. H.A. Engelbrecht

March 2020

Stellenbosch University https://scholar.sun.ac.za

UNIVERSITEIT·STELLENBOSCH·UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / Plagiarism *Declaration*

1 Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2 Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
I agree that plagiarism is a punishable offence because it constitutes theft.

3 Ek verstaan ook dat direkte vertalings plagiaat is.
I also understand that direct translations are plagiarism.

4 Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.

5 Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Copyright © 2020 Stellenbosch University
All rights reserved

Stellenbosch University https://scholar.sun.ac.za

Abstract

This thesis presents the development, implementation, and practical verification of a vision-based flight control and waypoint navigation system for a quadrotor unmanned aerial vehicle (UAV). The vision-based flight control system was developed to serve as a building block in a larger project to autonomously navigate an inspection drone relative to an inspection target in an indoor or GPS-denied environment. The intended application of this technology is to use autonomous drones to inspect large commercial airliners for external damage while the aircraft is parked in a maintenance hangar.

For the project, a vision-based UAV research platform was created using commercial off-the-shelf UAV hardware and open-source software. The Intel Aero RTF Drone was used as the research vehicle, the PX4 open-source software was used for flight control and state estimation, the Robotics Operating System (ROS) and the ArUco library was used for vision-based position and attitude determination, QGroundControl software was used for the ground control station, and the Gazebo software was used to create a simulation environment that supports both software-in-the-loop and hardware-in-the-loop simulations.

The vision-based flight control system was developed by modifying the PX4 flight control software to replace the existing GPS-based state estimator with our own vision-based state estimator, and adding vision-based pose estimation software that executes on a companion computer and determines the quadrotor position and attitude using external ArUco markers. The PX4 flight control architecture was also reverse-engineered and the controller gains were re-designed for the Intel Aero RTF flight dynamics. Finally, a waypoint scheduler was implemented to enable the quadrotor UAV to autonomously navigate a pre-determined set of position waypoints around an inspection target.

The vision-based flight control system was verified with laboratory experiments, simulations, and practical flight tests. The practical flight tests showed that the vision-based pose estimation reliably detects the ArUco markers and provides position and attitude measurements even during aggressive position and yaw angle steps. The vision-based state estimator successfully estimates the position, velocity, and attitude of the quadrotor UAV, and propagates the state when markers are temporarily lost from the camera's field of view. The flight tests also demonstrated that the vision-based flight control and waypoint navigation system provides stable and accurate position control for the quadrotor UAV and successfully navigates the vehicle to follow a given sequence of position

Stellenbosch University https://scholar.sun.ac.za

Uittreksel

Hierdie tesis beskryf die ontwerp, implementering, en praktiese verifikasie van 'n visie-gebaseerde vlugbeheer en wegpunt navigasie stelsel vir 'n vier-rotor onbemande vliegtuig (UAV). Die visie-gebaseerde vlugbeheerstelsel is ontwerp om te dien as 'n boublok in 'n groter projek om 'n inspeksie hommeltuig outonoom te navigeer rondom 'n inspeksie teiken in 'n binne-muurse of GPS-geweierde omgewing. Die praktiese toepassing vir die tegnologie is om outonome hommeltuie te gebruik om groot passassiersvliegtuie te inspekteer vir uitwendige skade terwyl die vliegtuig in 'n loods geparkeer is vir herstelwerk.

Vir die projek is 'n visie-gebaseerde UAV navorsingsplatform geskep deur gebruik te maak van kommersiële van-die-rak-af UAV hardeware en oopbron sagteware. Die Intel Aero RTF hommeltuig is gebruik as die navorsingsvoertuig, die PX4 oopbron sagteware is gebruik vir vlugbeheer en toestandsafskatting, die Robotics Operating System (ROS) sagteware en die ArUco biblioteek is gebruik vir visie-gebaseerde posisie en oriëntasie bepaling, die QGroundControl sagteware is gebruik vir die grondstasie, en die Gazebo sagteware is gebruik om 'n simulasie omgewing te skep wat beide sagteware-in-die-lus en hardeware-in-die-lus simulasies ondersteun.

Die visie-gebaseerde vlugbeheerstelsel is ontwikkel deur die PX4 vlugbeheer sagteware te wysig om die bestaande GPS-gebaseerde toestandafskatter te vervang met ons eie visie-gebaseerde toestandafskatter, en deur visie-gebaseerde lokalisering sagteware by te voeg wat uitvoer op 'n metgesel rekenaar en die voertuig se posisie en oriëntasie te bepaal vanaf eksterne ArUco merkers. Die PX4 vlugbeheer argitektuur is ook truwaarts uitgevind en die beheerder aanwinste is herontwerp vir die Intel Aero RTF vlugdinamika. Laastens is 'n wegpunt skeduleerder implementeer om die voertuig in staat te stel om 'n stel voorafbepaalde posisie wegpunte rondom 'n inspeksie teiken outonoom te navigeer.

Die visie-gebaseerde vlugbeheerstelsel is geverifieer met laboratorium eksperimente, simulasies, en praktiese vlugtoetse. Die praktiese vlugtoetse het gewys dat die visie-gebaseerde lokalisering die ArUco merkers betroubaar optel en posisie en oriëntasie metings verskaf selfs tydens aggressiewe posisie en gierhoek trapbewegings. Die visie-gebaseerde toestandafskatter skat suksesvol die posisie, snelheid en oriëntasie van die voertuig af, en propageer die toestand wanneer merkers tydelik uit die kamera se gesigsveld verdwyn. Die vlugtoetse het ook gedemonstreer dat die visie-gebaseerde vlugbeheer en wegpunt navigasie stelsel stabiele en akkurate posisiebeheer verskaf vir die vier-rotor UAV, en die voertuig suksesvol navigeer om 'n gegewe reeks posisie wegpunte te volg.

Stellenbosch University https://scholar.sun.ac.za

Contents

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

List of Figures

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

List of Tables

Stellenbosch University https://scholar.sun.ac.za

Nomenclature

Abbreviations

2D	Two Dimensional
3D	Three Dimensional
6DOF	Six Degree of Freedom
Aero	Intel® Aero Ready To Fly Drone
CoG	Centre of Gravity
DCM	Directional Cosine Matrix
EKF	Extended Kalman Filter
EV	External Vision
FCU	Flight Control Unit
GPS	Global Positioning System
HITL	Hardware-in-the-loop
IBVS	Image-based Visual Servoing
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
NED	North East Down
OBC	On-board Computer
PBVS	Position-based Visual Servoing
PID	Proportional Integral Differential
PnP	Perspective-n-Point
pub/sub	Publish/Subscribe
ROS	Robotic Operating System
RTOS	Real-time Operating System
SITL	Software-in-the-loop
SLAM	Simultaneous Localisation and Mapping
UAV	Unmanned Aerial Vehicle
VIO	Visual Inertial Odometry
VO	Visual Odometry

Aircraft Dynamics

m	Mass of Multi-rotor
g	Earths Gravitational acceleration
I_{xx}, I_{yy}, I_{zz}	Mass Moments of Inertia

Stellenbosch University https://scholar.sun.ac.za

T	Motor Thrust
T_c	Motor Thrust Command
T_m	Maximum Motor Thrust
τ	Motor Response Time Constant
d_{arm_x}, d_{arm_y}	Distance from CoG to Motor in the i direction
r_D	Rotor Drag
R_{LD}	Lift to Drag Ratio
F_D	Drag Force
ρ	Density of Air
C_D	Coefficient of Drag
A_x, A_y, A_z	Frontal Area in the i direction
$\mathbf{x}$	State Space Representation of Dynamic Model
$\mathbf{u}$	Input in State Space Form

Mass Moment of Inertia Experiment

d	Distance Between Ropes
l	Length of Ropes
t_p	Time of Oscillation Period

Control System

K_P	Proportional Gain
K_I	Integral Gain
K_D	Derivative Gain
T_f	Filter Time Constant
u	Input to Control Loop
y	Output to Control Loop
D_P, D_ϕ, D_V, D_E	Controller Transfer Function
G_P, G_ϕ, G_V, D_E	Plant Transfer Function
F_T	Total Thrust Force

Visual Odometry

f	Focal Length
c_x, c_y	Camera Centre
P	Principal Point
r	Radius of Image Plane for Distortion
$L(r)$	Distortion Factor
R	Rotation Matrix
T_{vec}	Translation Vector
r_{vec}	Rotation Vector
T_{vec}	Translation Vector
β	Vector Angle of Axis Angle

Stellenbosch University https://scholar.sun.ac.za

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations for their contributions to this project:

- My supervisors, Japie and Herman Engelbrecht, for their insights and guidance throughout this project. Thank you for always being accessible and all the effort you put in to help solve the problems that arose. I would not have been able to complete this thesis without your guidance.
- Denel Aeronautics and North-West University for providing a postgraduate bursary and financial support for this research project under the Technology and Human Resources for Industry Programme (THRIP) for 2018/2019.
- The safety pilot, Micheal Basson, for his expertise in flying and ensuring that the multi-rotor landed safely in all the tests.
- Anton Erasmus for all his help with practical set-ups and all the hours spent on the software.
- Christian Muller for always being of assistance with questions concerning the project at hand.
- My family, for the their support and aid throughout my studies.
- My wife, Juan-Mari, for her support and always having the time to listen to my progress and frustrations.

Stellenbosch University https://scholar.sun.ac.za

Chapter 1

Introduction

1.1 Background

A commercial aeroplane that flies through a storm must be inspected for damage. Damage may be caused by lightning strikes or hail of which the pilot might be unaware. Such inspections can ground an aeroplane for up to 4 hours while every surface is manually examined for signs of damage. The examination generally consists of an individual moving around the aeroplane with a ladder or a mobile elevating work platform. This unscheduled down time is inconvenient and expensive for airlines. Therefore, the use of a quadrotor inspection drone is proposed as a possible solution to reduce the time loss. The inspection drone could fly around the aeroplane with a high-resolution camera and capture images of all the surfaces in a 20-minute flight. The images could be painted onto a 3D CAD model of the aeroplane and a human inspector or computer algorithm could then examine the surfaces. Machine learning algorithms combined with computer vision could speed up the process of identifying problem areas and could highlight them for the human inspector's attention. This could be done while the aeroplane is being refuelled and prepared for the next flight, minimising the overall down time of the aircraft.

A serious problem with replacing an inspector with an unmanned aerial vehicle (UAV), is that UAVs are not allowed to fly on most commercial airports. The inspection would therefore have to be carried out inside a hangar, where the global positioning system (GPS) signal will be severely degraded, if at all available. An autonomous inspection drone would have to navigate itself around the circumference of the aircraft without the aid of GPS sensors. Alternative localisation techniques, such as vision-based localisation, will have to be employed.

Another potential solution could be for a human pilot to remotely control the inspection drone so that autonomous navigation is not required. However, this would mean that a qualified drone pilot would have to be available for all inspections, and also introduces the risk of human error that could lead to damage of the aircraft. The automation of the inspection drone would allow the inspections to be performed by any technical person, and would not require them to be skilled in the remote control of drones.

Therefore, a need exists for a GPS-less flight control and waypoint navigation system to enable an autonomous inspection drone to navigate itself along a pre-determine set of position waypoints relative to an inspection target, in an indoor or GPS-denied environment.

Stellenbosch University https://scholar.sun.ac.za

1.2 Research Goal

The primary goal of this research project is to design, implement, and verify a vision-based flight control system to control the position of a quadrotor relative to an inspection target in a GPS-denied environment. For this project, a marker-based pose estimation approach must be used to accurately determine the position and attitude of the quadrotor relative to the inspection target. This research will serve as a stepping stone for future research on markerless pose estimation and vision-based flight control.

A secondary goal of this research project is to develop a UAV research platform using commercial off-the-shelf UAV hardware and open-source software. This will replace the Electronic System Laboratory's (ESL's) in-house developed avionics, ground control station, and hardware-in-the-loop simulation environment. The ESL's UAV research hardware was developed in the early 2000's when commercial UAV autopilots were not available yet. However, the ESL's electronic components have since become obsolete and its manufacturing is no longer supported. Therefore, a need has been identified to migrate to commercially available UAV hardware, and to develop a new UAV research platform that includes the UAV avionics, a ground control station, and support for hardware-in-the-loop simulation.

1.3 Objectives

The goals of this project are broken up into smaller objectives, namely:

1. To select and procure a suitable off-the-shelf quadrotor UAV: an airframe, a flight control unit, an on-board computer, and a suite of sensors including an inertial measurement unit (IMU), GPS sensor, magnetometer, barometric sensor, and camera module.

2. To select suitable open-source flight control software to serve as the basis for the vision-based flight control system.

3. To create an integrated UAV system consisting of the UAV, the ground control station, and the hardware-in-the-loop simulation environment.

4. To establish a mathematical model of the quadrotor UAV flight mechanics that can be used for flight control design and analysis, and for simulations.

5. To reverse engineer the flight control system and the state estimators implemented by the open-source flight control software.

6. To re-design the flight controllers gains based on the flight dynamics of the chosen commercial off-the-shelf quadrotor UAV, if necessary.

7. To design and implement a vision-based localisation algorithm that executes in real time on-board the quadrotor UAV.

8. To design and implement a vision-based state estimator that estimates the position, velocity, and attitude of the quadrotor UAV without using GPS sensor measurements.

Stellenbosch University https://scholar.sun.ac.za

9. To design and implement a vision-based flight control and waypoint navigation system for the quadrotor UAV.

10. To verify the correct operation of the vision-based localisation, state estimation, flight control, and waypoint navigation using laboratory experiments, simulation tests, and practical flight tests.

1.4 Methodology

To reach the aim and objectives of this project the following approach was taken. First, a quadrotor UAV was sourced that (a) allowed access to the flight control software, (b) had at least one camera on-board and (c) had on-board computational power for image processing and position and orientation (pose) estimation. The quadrotor UAV had to be controllable in flight and, therefore, a control system was required. Different open-source control software were investigated and the final choice had to be made between the in-house flight control software of Stellenbosch University's Electronic Systems Laboratory (ESL) research group and the ArduPilot and PX4 open-source flight control software suites. The ESL's in-house software was developed for the ESL's in-house avionics hardware and would have to be ported to the chosen commercial off-the-shelf hardware, the ArduPilot open-source software did not provide support for hardware-in-the-loop simulation. Consequentially, the PX4 open-source software remained as the only viable option and was chosen as the flight control software. PX4's controllers were analysed and were found to be two-degrees-of-freedom successive loop closure PID controllers with various safety checks. The architecture was fairly complex and it was necessary to create a simplified representation to design gains for these controllers.

First, a dynamic model of the quadrotor UAV in flight was derived. The dynamic model was then linearised around hover and used to design gains for the controller that could be directly implemented into the PX4 controllers. The designed gains were tested using Software-in-the-loop (SITL) simulations. Finally, practical flight tests were performed to verify the correct operation of the flight controllers on the real quadrotor UAV.

The next step was to localise the quadrotor UAV in an environment without GPS. After a thorough investigation of the available literature, it was found that cameras and consequently computer vision was a feasible solution. However, in computer vision there are two main methods of localisation that are commonly used. The first is simultaneous localisation and mapping (SLAM), where the environment is unknown and the vehicle is required to simultaneously locate itself while mapping its environment. The second method is localisation and requires full or partial knowledge of the environment and only needs to be localised. The second method was chosen, because the shape of the inspection target will be known and the inspection waypoints will be pre-determined. According to the literature (that will be reviewed in Chapter 2 Section 2.3.2), the two most popular solutions for localisation in a known environment is marker-based and marker-less localisation, where the marker is any identifiable object placed in the environment with the sole purpose of facilitating localisation. Marker-based localisation was chosen to mediate time constraints on this project because it was faster to implement. The localisation technique requires the camera model to be derived and the measurement noise to be characterised. Open-source software for marker-based localisation was investigated, as it was the most efficient solution given the time constraints of the project. The ArUco

Stellenbosch University https://scholar.sun.ac.za

artificial reality libraries were chosen because they are considered the current industry standard. The system was implemented in hardware and tested. This project made use of marker-based localisation, as the scope of the project was too large to attempt more complex methods for this phase in the development of the larger process. However, the system was implemented in a modular way to allow the localisation algorithms of this project to be easily substituted and refined for future systems.

An estimator was implemented to fuse the vision-based position and attitude measurements with the other sensors measurements. The PX4 software includes a GPS-based estimator and a limited vision-based estimator, which was replaced with our own vision-based state estimator that uses the full three-axis attitude measurement.

The complete system was integrated with the different software components loaded on the flight control unit and the on-board computer of the quadrotor UAV. The integrated system was then tested through a series of practical flight tests. The tests were completed in consecutive steps, where each of the steps tested a larger part of the system. The controller and estimator were first tested with GPS, where-after the waypoint scheduler was added and tested with GPS as well. When both tests were successful, the vision-based system was activated and tested. The quadrotor UAV was flown by a safety pilot before switching to the waypoint scheduler for the final test, which was aimed at proving the integration of all the different subsystems.

1.5 Thesis Outline

Chapter 1 provided the background and motivation for the research, stated the research goals, enumerated the project objectives, and gave an overview of the methodology.

Chapter 2 presents the literature review. The literature review includes the basic dynamics of a quadrotor UAV, the typical sensors that are used, the different vision-based localisation techniques, and the different open-source software packages that are available.

Chapter 3 presents a system overview with a physical representation of the quadrotor UAV. This is followed by detailed descriptions of the hardware and software interactions, the software-in-the-loop and hardware-in-the-loop simulation environments, and the flight tests.

Chapter 4 establishes the mathematical model of the quadrotor vehicle's flight mechanics, and linearises the model for controller design.

Chapter 5 discusses the flight control system design by providing an overview of the flight control architecture, followed by the detailed designs of the individual controllers, and finally the step responses to verify the correct design.

Chapter 6 describes the camera model that is used for the image processing, followed by an overview of the pose estimation process, and the sensor noise characterisation.

Chapter 7 provides an overview of the state estimator, describes the changes that were made to enable vision-based state estimation, and describes the laboratory tests that were used to verify the correct operation of the vision-based pose estimation.

Chapter 8 describes the flight tests that were performed and presents and discusses the flight test results.

Chapter 9 summarises the conclusions and makes recommendations for future research.

Stellenbosch University https://scholar.sun.ac.za

Chapter 2

Literature Review

This chapter presents a literature review of previous research relevant to vision-based flight control of quadrotor UAVs. First, basic multi-rotor dynamics is covered, and an overview is provided of the on-board sensors that are generally available on quadrotors. Next, different techniques for vision-based localisation, the difference between marker-less and marker-based localisation, and different techniques for vision-based control are covered. Finally, an overview is given of the available open-source software packages for flight control and simulation.

2.1 Basic Multi-rotor Dynamics

Multi-rotors can be in large number of different configurations. However, the core principle is that at least two actuators/motors are used with a rotating propeller. The number of propeller and motor pairs used is determined by its practicality. This project made use of a quadrotor in a cross-configuration, thereby constraining the configuration down to use 4 motor and propeller pairs. The two most common configurations for quadrotors are plus- and cross-configurations. As the name suggests, the plus-orientation resembles a plus-symbol while flying forward (as shown in Figure 2.2), and the cross-configuration resembles the letter "X" (as shown in Figure 2.1).

Stellenbosch University https://scholar.sun.ac.za

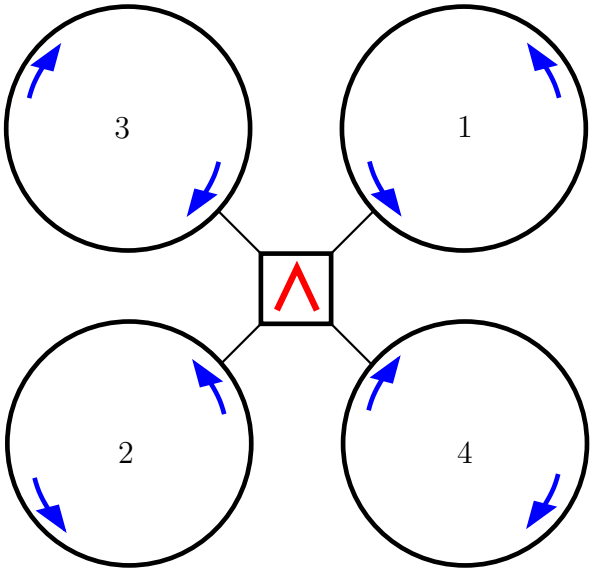

Figure 2.1: Cross Configuration

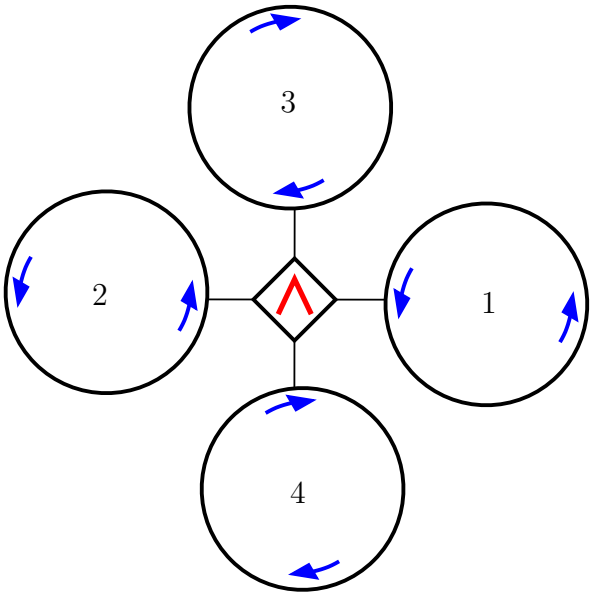

Figure 2.2: Plus Configuration

At this point it is necessary to turn to the basic principles of quadrotor dynamics. The first important state - which is also the basis of many of the following concepts - is when the quadrotor is hovering. Hovering is defined as the state in which the quadrotor is suspended in the air with no movements (other than the rotors) or rotations. Therefore, all forces and moments acting on the body sum to zero. A spinning propeller produces a thrust directly along the axis around which the propeller is rotating, as well as a moment around the axis in the opposite direction. Each motor's thrust force and moment are tightly coupled. Consider the cross-orientation: when each pair of motor and propellers is represented by a single force and moment, a free-body diagram can be drawn as seen in Figure 2.3. When all of the motors provide the exact same thrust and the combined thrust is that of the quadrotor's weight, the quadrotor will hover.

Stellenbosch University https://scholar.sun.ac.za

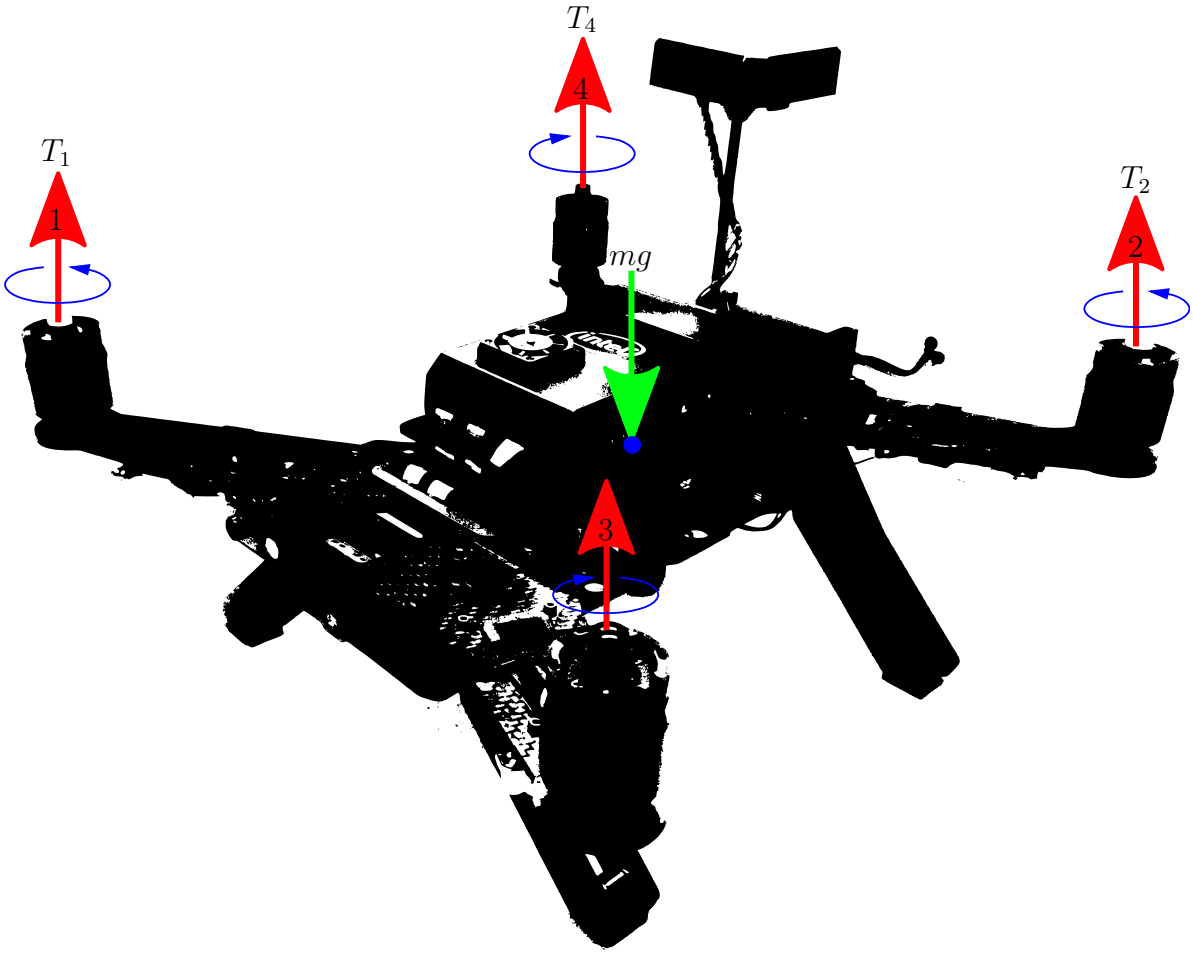

Figure 2.3: Free-body diagram of a quadrotor

Motion of the quadrotor is then achieved by a change in the moment or thrust of the motors from this equilibrium state, which disturbs the balance and causes the quadrotor to move. An upward movement is caused by an equal amount of increase in the thrust of each motor and a downward movement is caused by an equal amount of decrease in thrust. The propellers are configured so that propeller 1 spins in the opposite direction from 3 and 4. Similarly, propeller 3 spins in the opposite direction from propeller 1 and 2 (see Figure 2.3). This is to balance the net moments on the quadrotor. Therefore, to yaw the vehicle a pair of motors (e.g. 1 and 2) must spin faster while the other pair (e.g. 3 and 4) must spin slower. This keeps the net thrust constant while causing a net moment, resulting in a constant height and rotation around the axis of gravitation. The same principle holds for roll and pitch where, depending on the orientation, motor pairs provide more or less thrust to cause rotation. As the body rotates, more general thrust is required to keep the quadrotor at the same altitude while moving in the direction of the desired angle. Therefore, to move forward the quadrotor will produce more thrust to the back motors (e.g. 2 and 4) and less to the front ones (e.g. 1 and 3). As the body starts to rotate forward, a thrust increase in all motors will cause the quadrotor to start accelerating forward. The control system will perform the opposite of the thrust commands to slow the quadrotor down, stop it or reverse it.

The quadrotor is an inherently unstable system and therefore needs feedback control for stable flight. A robust control system capable of maintaining a quadrotor in the

Stellenbosch University https://scholar.sun.ac.za

air requires feedback from the on-board sensors to stabilise and control the quadrotor vehicle. Measuring the speed, thrust or moment of each propeller is difficult because the thrust and moment of each propeller is relative to the speed of the motor's rotation. The rotation, in turn, is proportional to the current applied to it. In practice no two motors will consistently provide the same thrust for the same current. Therefore, the resulting motion resulting from the thrusts and moments acting on the quadrotor must be measured with on-board sensors to enable feedback control of the quadrotor's translational and rotational motion. The variables that are required for feedback control include the angular velocity, the attitude, the translational velocity, and the translational position. The sensors that are typically used for feedback control of a quadrotor will be reviewed in the next section.

2.2 Sensors

To navigate in three-dimensional space, an autonomous vehicle must be able to determine its pose (position and attitude) in real time. While there is a wide array of sensors available, the most commonly used are inertial measurement units (IMUs) and GPS receivers. The simplest technique is to use an IMU to determine the attitude and a GPS receiver to determine the position. However, this project is intended for use inside a hangar where the GPS signal will either be severely degraded or completely absent. Therefore, auxiliary sensors are required to aid the GPS in positioning or to completely replace the GPS. Balamurugan, Valarmathi and Naidu [35] performed a survey on the state-of-the-art localisation techniques used in a GPS-denied environment and the implementation of a system that uses a camera as an auxiliary sensor with additional validation checks on the GPS signal. Zhou et al. [33] used a GPS to position a vehicle outside a building and used vision-based localisation to manoeuvre a quadrotor UAV into a building through an open window.

Cameras are lightweight and inexpensive sensors that provide a high degree of accuracy and bandwidth and have been used more than a decade for localisation. The placement of the cameras defines the system's capabilities and reliability. One common approach is to use external cameras placed in the environment to determine an object's position and attitude, which is a process called motion capture (MoCap). Identifying markers on the object aid in the determination of the object's pose and the pose information is then transmitted to the object through telemetry. Tisse, Fauvel and Durrant-Whyte [53] have shown the success of such a system. However, the risk of losing telemetry and damaging the inspection target (aeroplane) is too great for such a system and therefore cameras mounted on-board the vehicle are better suited for the purposes of this project.

The literature contains many different orientations and camera configurations for on-board cameras. Some of the more common configurations are monocular[1][13][20][21][22][29][34][35][36], stereo[2][14][24][42] and multi-camera arrays [3][28]. There are some hybrid camera configurations such as those implemented by Shen et al. [25][26] where a monocular camera with a high frame rate is used, aided by a secondary stereo camera with a lower frame rate. This retains the simplicity and robustness of a monocular

Stellenbosch University https://scholar.sun.ac.za

camera, while mitigating the ambiguity of a monocular camera using stereo vision.

2.3 Localisation

2.3.1 SLAM

The visual odometry (VO) system determines the pose of a vehicle by comparing each image frame to either a previous frame or a reference map of the environment. There are two ways in which an image frame can be analysed: either directly or indirectly. Engel, Koltun and Cremers [52] defines "indirect methods" as a combination of the sensor measurement values into primitive geometric shapes (such as lines, curves, corners, etc.) used for pose estimation. Direct methods use the raw sensor measurements as inputs to a probabilistic model to estimate the pose. Regardless of the method of analysis, the pose is the problem that requires solving. When the environment is unknown, the only option is to generate a map while simultaneously localising in the environment -a process known as Simultaneous Localisation and Mapping (SLAM).

A map of the environment can either be sparse or dense, where "sparse" uses a select number of independent points (mostly corners) to describe the environment and "dense" maps attempt to use all the pixels in the 2D image plane to describe the environment [52]. The SLAM solutions considered must be able to run in real-time and on-board a UAV's constrained hardware. Dense map solutions make use of an RGB-D camera (a monocular RGB-camera with a sensor to provide depth for each pixel) and have shown good representations of the environment, as demonstrated by [43] [44] [45] [46] [47] [48] [49]. The sparse map solutions provided by [40] [41] [42] [10] [12] [27] [33] use RGB-cameras and showed working examples of actual flight. A more extensive survey of the current state-of-the-art SLAM solutions was done by Cadena et al. [51] and answered the question of whether the problem of SLAM has been solved as follows:

> "To achieve truly robust perception and navigation for long-lived autonomous robots, more research in SLAM is needed. As an academic endeavor with important real-world implications, SLAM is not solved." ([51], pp. 1326).

Considering this statement, along with the large computational footprint used by SLAM algorithms and the fact that the map will be partially known (a 3D model of the inspection target is available), made it impractical to investigate SLAM solutions any further. However, SLAM could be a solution to the localisation problem for future work.

2.3.2 Indirect Visual Odometery

When a map of the environment exists, the problem of pose estimation is reduced to a localisation problem. Indirect methods for localisation can be subdivided into marker-based and marker-less localisation, where "marker-based" refers to some identifiable markers placed in the environment of which the pose is exactly known and "marker-less" is the identifying of features in the image and relating it to a model of the environment to determine the pose.

[3]Monocular – single camera mounted facing forward or downwards that can measure infrared (IR), colour (RGB), black and white (monochrome) or high definition.

[3]Stereo – Two monocular cameras that are facing in the same direction.

[3]Multi-camera array – Two cameras facing in different directions or more than two cameras.

Stellenbosch University https://scholar.sun.ac.za

Marker-less

Bleser, Wuest and Stricker[21], Jin, Favaro and Soatto [22] and Veth, Raquet and Pachter [24] showed how marker-less pose estimation can be done for augmented reality, where Shen, Mulgaonkar, Michael and Kumar implemented a marker-less localisation system onto a UAV [25] [26].

Marker-based

The markers placed in the environment can be almost anything. The most successful markers are asymmetrical markers, so that detecting the marker will produce a unique pose estimate. The easiest markers to implement are square co-planar markers. Metni and Hamel [15], Jayatilleke and Zhang [18], Bacik et al. [38] and Mac et al. [20] showed the effectiveness and practical results to be expected from square co-planar markers. Benini, Rutherford and Valavanis [32] did a survey on the state-of-the-art marker-based libraries for a more comprehensive list of available square markers. Nguyen et al. [36] showed how circular markers can achieve better accuracy than the square markers. Nitschke [29] and Singh et al. [37] increased the range from which the co-planar markers can accurately be used for pose estimation, by nesting markers within each other (Nitshcke for square markers and Singh et al. for circular markers). Roozing and Göktogan [19] argued that markers placed in any environment are intrusive and not aesthetically appealing and, therefore, created invisible IR reflective markers. When using co-planar markers an ambiguity exists because rotation and translation are tightly coupled. The result is an estimation error because all the points of the marker lie on a single plane. Konomura and Hori used pyramid shaped markers to address this problem [16], while Vogt et al. used a cluster of cylinders of varying height [17].

The ArUco library, created by Rafael Muños and Sergio Garrido [39], was used because it is the current standard for square co-planar markers [36][37]. Even though the other marker types have shown great success and accuracy, the author decided to use the ArUco library. Future projects may use more advanced or less intrusive methods. The methods used specifically for the detection of the markers and pose estimation will be discussed in Chapter 6.

2.4 Visual Control

Visual servoing is a technique used to control a vehicle by using visual information. There are two approaches to visual servoing, namely position-based and image-based. Both approaches have been successfully used and will now be discussed.

The first approach is the image-based visual servoing (IBVS) controller, which is designed in the 2D image plane, where the pixels are directly used to compute the control commands. This offers a reduced computational complexity and eliminates any camera calibration errors. However, the translational and rotational movements are highly coupled, resulting in a nonlinear plant [54], thereby requiring a more complex controller. IBVS is well suited for use when the environment is unknown or when following a target. Mebarki, Lippiello and Siciliano showed the design process and implementation of an IBVS system [55].

The second approach is the position-based visual servoing (PBVS) controller which requires a pre-controller step, where features are extracted from the image plane and

Stellenbosch University https://scholar.sun.ac.za

matched to a known map of the environment. The pre-controller step provides a pose estimate in Cartesian coordinates to the controller. This results in the control law remaining in Cartesian coordinates and decoupling the rotational and translational movements for a linearisable plant [56]. The classical control architecture can therefore be used, such as using cascaded control loops.

A third approach consists of hybrids of these two approaches. One such hybrid was designed by Malis, Chaumette and Boudet and is aptly named "2 ½ visual servoing" [57]. This system decouples the rotational and translational movements by using the pre-controller step of PBVS for rotational information and the pixels in 2D image plane for the translational information. This hybrid approach allows for lower computational complexity than the PBVS, while still using a linear controller.

Even though all three approaches discussed above were viable, the PBVS approach was best suited to the requirements of this project, since it is the more stable type when a map of the environment is known. Consequently, the use of nonlinear controllers is eliminated and the only requirement is a good camera calibration process.

2.5 Flight Control Software packages

The previous sections have emphasised the importance of a good control system for stable and reliable flight. At the beginning of this project there were three options to investigate. The first option was Electronic System Laboratory's (ESL's) in-house developed flight control software that has been used in many previous projects. The other two available options for control software was the open-source software packages, ArduPilot and PX4. Both PX4 and ArduPilot are commonly used in industry and are well documented, with active development teams continually improving the systems.

All the software packages have cascaded control architectures, where the different states are decoupled. Estimation is performed by an extended Kalman Filter (EKF), where ArduPilot and PX4 also provide other estimation architectures. The major differences are described below.

2.5.1 ESL's In-house Flight Control Software

The ESL flight control software is intended to execute directly on the PIC microcontroller hardware without an operating system. This allows close control of the timing and lowers the computational and memory footprint on the processor on which it executes. It allows for easier development and simpler code structures. Furthermore, the ground station software intended for use during flight tests, was created to interact with the aerial vehicle. Software-in-the-loop (SITL) and hardware-in-the-loop (HITL) simulations environments have been developed, which allow customisable simulation models to be used for different types of vehicles. The documentation for most of the software is contained in previous thesis reports in the Stellenbosch library and in inline comments in the source code. However, the ESL's in-house developed autopilot hardware has reached its end-of-life, and its manufacturing is no longer supported. A need has therefore been identified to develop a UAV research platform using commercial off-the-shelf UAV hardware and open-source software, and to migrate the ESL's in-house flight control and guidance systems to the new platform.

Stellenbosch University https://scholar.sun.ac.za

2.5.2 ArduPilot

ArduPilot was created to run on a real-time operating system (RTOS). However, the system was designed to abstract the RTOS to function as a bareback system, which allowed new developers and hobbyist to easily learn the architecture. The documentation for ArduPilot provides recommendations for periphery software packages that are well supported. These include ground station software and simulation environments. At the time of this project, the ArduPilot's development team announced their decision to forego the use and support of Hardware-in-the-loop (HITL) simulations. An overview of the software architecture can be found online [59].

2.5.3 PX4

PX4 is also designed to run on an RTOS and, unlike ArduPilot, the software makes full use of the operating system. A publish-and-subscribe pattern software was implemented allowing for a modular approach, enabling easy addition of functions and adaptation of existing functions. PX4's documentation also provides recommendations for peripheral software packages. PX4 has full support for both SITL and HITL simulations and provides a lockstep SITL simulation environment. The software documentation can be found online [60]. PX4 was chosen as the flight control system, because it has full support for HITL simulations and has an active community with well written documentation.

2.6 Simulation Environments

PX4 provides three open-source software packages as simulation environments, namely Gazebo, jMAVSim and AirSim. Each one of these environments has easy integration with PX4 and supports ROS. All three have support for HITL simulations and allow easy integration. However, jMAVSim does not allow the addition of extra sensors such as cameras. Consequently, it cannot simulate obstacles or a visual inspection. AirSim has cameras already included and allows for obstacles to be placed in the environment, but uses the Unreal Engine 4, which needs a strong Graphics Processing Unit (GPU) to run [58].

Gazebo allows easy addition of cameras and obstacles and, even though the rendering of the visualisation can be computationally expensive, it has two separate servers for the simulation. One server renders the visualisation and the other simulates the physics. This allows the physics simulator to work independently from the rendering, thereby freeing up valuable computation time. Gazebo was the logical choice for the simulation environment.

2.7 Vision-Based Pose Estimation Software

The previous sections covered the software and hardware that were chosen for this project, namely PX4 for the flight control and state estimation, Gazebo for the simulations, and an off-the-shelf quadrotor vehicle. Gazebo uses ROS packages for communications to PX4 and the camera used could stream the images captured as a ROS topic. Therefore, ROS was well suited to serve as the environment for the image processing.

Stellenbosch University https://scholar.sun.ac.za

Despite its name, Robotic Operating System (ROS) is not an operating system, but rather a middleware software package. Like PX4 and Gazebo, it uses a publish-and-subscribe pattern and a topic is referred to as a ROS topic. A ROS topic can be any data type, which includes images. Each class is called a node. For a ROS node to be executable, a master node is required. The master node simply initialises the system and manages the ROS topics. MAVROS was chosen to serve as the master node for the software because it provides functionality to convert ROS topics into MAVLink messages, and vice versa (Mavlink is the communication protocol used to communicate with PX4).

2.8 Summary

To summarise, a quadrotor in a cross-configuration was selected and flight is achieved through differential control of the motor and propeller pairs. A camera was chosen as the sensor to replace the GPS for localisation due to it being inexpensive and accurate. SLAM algorithms were investigated, but due to the higher complexity and computational capacity required and the map of the inspection target being known, it was decided to use localisation only. Marker-based and marker-less localisation methods were discussed and marker-based localisation was chosen for this project. Different visual control strategies were compared and Position-Based Visual Servoing (PBVS) was chosen, because it is more stable when a map of the environment is available. Open-source software was chosen for the flight control system, visual processing system and simulation programs. The reason being it allows for faster development and has a community of developers supporting it. The software packages selected were PX4 for the flight control system, ROS and ArUco for the image processing, and Gazebo for the simulations.

Stellenbosch University https://scholar.sun.ac.za

Chapter 3

System Overview

This chapter provides an overview of the UAV system that was created using commercial off-the-shelf UAV hardware and open-source software, to achieve the objectives of this research project. The components of the system included: the quadrotor UAV, the flight control and guidance system, the ground control station, the simulation environment (supporting both software-in-the-loop and hardware-in-the-loop simulations), and the vision-based pose estimation system. The components were individually designed and verified through simulations and physical testing. Once a component was successfully tested it was added to the system. The entire system was created in a single simulation environment and the integration was tested. Due to hardware limitations, which will be discussed below, the entire system could not be tested through hardware-in-the-loop (HITL) simulation. However, GPS-based estimation and control could be tested and the visual inertial odometry system without control could be tested separately. Both showed successful results. Since all the simulations showed that flight will be feasible, flight tests were conducted to prove the integration of the system.

Every step will be discussed in more detail in the Chapters to follow, though some concepts and layouts are complicated. Therefore, it is necessary to describe these concepts and layouts in this Chapter. The concepts and layouts of relevance here are: (1) The hardware specification and restrictions; (2) How the software was implemented on the hardware; (3) The communication flow between the different software packages; (4) How the SITL and HITL simulations were implemented; and (5)A brief overview of the flight test setup.

3.1 Hardware

The flying platform is the key component, since most of the software is reliant on the hardware (either directly such as computational power or indirectly such as response time to commands). The Intel® Aero Ready-To-Fly drone (Intel® Aero RTF drone) was chosen as best suited for the purpose of this project. A full list of its specifications is given in appendix A. Important aspects that are relevant for this thesis are: the STM32 F427V chip used as the flight control unit (FCU); the quad-core Intel® Atom computer used as the on-board computer (OBC); the Intel® RealSense camera module; and the GPS. The figure below is a visual representation of how these components are placed on the vehicle.

Stellenbosch University https://scholar.sun.ac.za

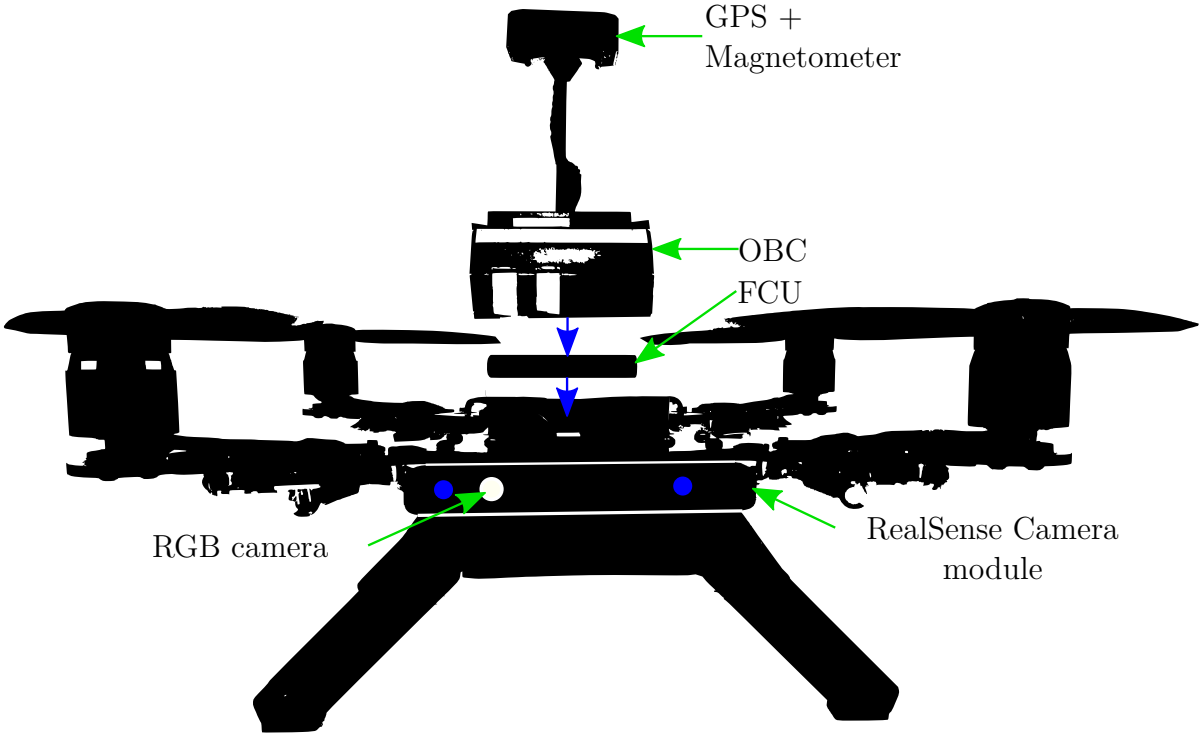

Figure 3.1: Hardware configuration of the Intel® Aero Ready-To-Fly drone

Figure 3.2 shows how the different hardware components are connected.

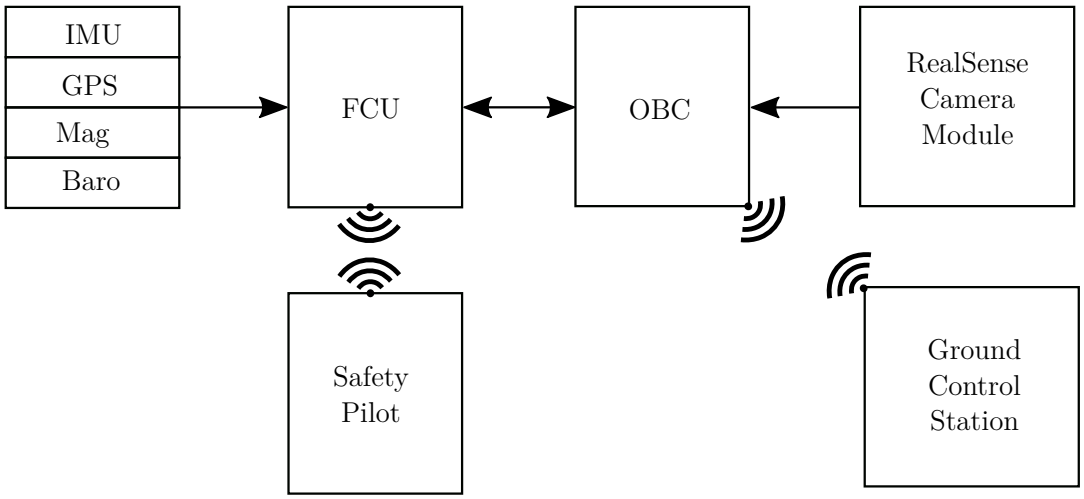

Figure 3.2: Communication flow between the hardware components of the UAV system

The FCU is connected to the OBC via a single HUART serial connection. The RealSense camera module is directly connected to the OBC, while the rest of the sensors (inertial measurement unit (IMU), GPS, magnetometer, etc.) are connected directly to the FCU. The IMU contains a three-axis accelerometer and a three-axis gyrometer that is directly attached to the FCU. The ground control station is a laptop running the QGroundControl open-source software, and its communicates with the OBC on the quadrotor via a Wi-Fi connection. All communications from off-board the vehicle are

Stellenbosch University https://scholar.sun.ac.za

over the Wi-Fi connection to the OBC, except for the wireless connection between the safety pilot's transmitter and the receiver connected directly to the FCU.

The Intel® RealSense camera module has two infrared cameras (in a stereo configuration), a depth sensor, and an RGB-camera. Due to the scale of the project, only the RGB-camera is used. The RGB-camera streams the images at 30 fps with a resolution of 640x480 to a ROS topic (see Section 2.7), along with the camera and distortion matrices. The camera module is fixed to the front of the vehicle where the RGB-camera's center is offset from the centre of gravity (CoG) of the Intel® Aero RTF drone.

3.2 Software Tool Chain

The quadrotor UAV is controlled by on-board flight control and state estimation software. The PX4 open-source flight control software was chosen to implement the control system and estimator. The Robotics Operating System (ROS) was chosen to perform the image processing and pose estimation because the on-board Intel® Realsense camera can stream the images as a ROS topic. ROS was also used to implement the waypoint scheduler as a simple navigator to test that the system is successfully integrated. A key component to verify the system, without the risk of damaging any hardware, is to simulate the platform and system before practical flight tests. Gazebo was used for these simulations. An overview of these three software packages follows and the different a summary of the tool chain is shown in Figure 3.3.

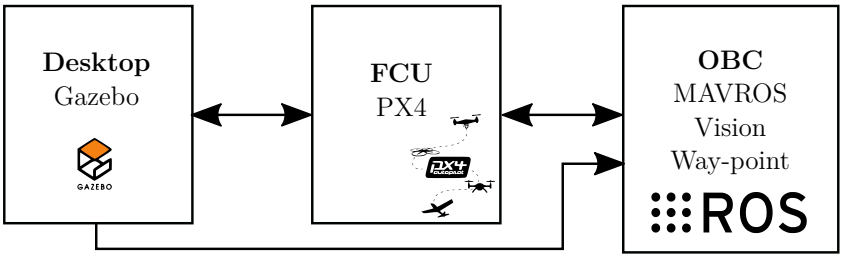

Figure 3.3: Software tool chain

3.2.1 PX4

Several flight control and estimator architectures are included in the PX4 package. The flight control architecture that was chosen for this project is a successive loop closure architecture with two-degrees-of-freedom (2 DoF) PID controllers. An overview and analysis of the flight control architecture will be given in Chapter 5.

The state estimation is performed using a delayed time horizon extended Kalman filter (EKF). The estimator works in two stages. The first stage is a standard strap-down INS EKF (Inertial Navigation System Extended Kalman Filter), where the sensor measurements are stored in a buffer and the estimator is delayed between 100 and 200 ms. The EKF works at a delayed time where it uses the stored sensor measurement from a buffer as it reaches the estimator's delayed time. This allows the use of sensors with differing rates. The problem is that the state estimates are delayed by 100 to 200 ms, which is too slow to fly. The second stage is to propagate the delayed states to the current time using a combination of the latest IMU measurements and an output prediction algorithm. The estimator is explained in Chapter 7.

Stellenbosch University https://scholar.sun.ac.za

In addition to an estimator and controllers, PX4 has safety checks and logging required for reliable flight. PX4 was loaded onto the FCU.

3.2.2 ROS

The ROS environment allows easy integration of different functionalities required on the OBC. As mentioned in Chapter 2, the master node used for this project is MAVROS, which converts ROS topics to Mavlink messages and vice versa. This also means that different ROS-nodes could subscribe and use the actual flight states. Localisation and navigation is implemented on the OBC.

Localisation

The localisation used in this project is marker-based. The marker-based localisation makes use of square fiducial markers placed at known locations and orientations. The ArUco virtual reality libraries are used for the localisation, which are in turn based on OpenCV.

Each marker that is used has a black border that encloses a 5x5 grid of black and white blocks. The ArUco detection process first detects the black square border of the markers to determine possible candidates. Then it decodes the 25-bit encoding inside the square border of each identified candidate. The encoding provides a marker ID which correlates to the marker map, where the marker's pose is stored. The pose of the camera can then be calculated by solving a perspective-n-point problem. An overview, including the camera model used, will be provided in Chapter 6.

Navigation

The waypoint navigation system for this project is implemented by a waypoint scheduler, where each waypoint is given as a position and heading angle. The scheduler is capable of sending the position and heading angle references directly to PX4's controllers through the MAVROS node. There is no reliance on the ground station for any navigation commands.

3.2.3 Gazebo

The last component required for the successful completion of this project was the simulation of all the previous components to enable testing and developing without risk to the hardware. Gazebo is an open-source software package used for simulations and has many protocols already implemented for communication with PX4 and ROS. Its largest advantage is the ability to add cameras to a model and to stream the raw images to a ROS topic. This enables for a simulation environment that closely resembles an actual flight. Gazebo allows the addition of noise, drift and biases on sensor models (including the camera models). This allows conceptual testing with noiseless sensor measurements and robustness testing with noisy sensor measurements.

Gazebo provides a simulation environment in which gravity, ground plane and wind conditions are defined. A model can be added, which can be either static or dynamic, allowing for the addition of vehicles and obstacles. A Gazebo model contains a physical component (also referred to as the collision component) and a visual component. Gazebo's physics engine acts on the physical component: the visual component is merely

Stellenbosch University https://scholar.sun.ac.za

the way in which it renders. The visual component does not influence the physics or behavior of the model, except for the additional computational power necessary to render it.

The physical component consists of at least one link but can contain any number of links. A link is a geometrical shape which has a mass, a moment of inertia and a defined magnitude. Different links are connected together by joints. A joint's movement can be restricted in any or none of the six-degrees-of-freedom. The combination of links and joints creates a model, with which the physics engine of Gazebo interacts. Therefore, no explicit mathematical equations are required, enabling easier and more complex modeling. A practical example is a case where a quadrotor is created in this manner. The base would be a simple (or complex) box, four beams for the motor arms (with static joints), a cylinder at the end of each arm for the motors, and a propeller on each motor (jointed to the motors with constraints only allowing rotation around one axis). Each link requires the mass and moments of inertia to be known. Each propeller would have a small script (also known as a plug-in) providing correlation between rotational speed and thrust. The control software connected to such a simulation would send rotational speed commands to the propellers and the rest of the physics would be managed by Gazebo's physics engine.

Since there are many variables that have to be tested to verify that Gazebo's physical simulator is comparable to the mathematical derivations, it was decided to bypass most of it.

Instead of multiple links, a single box with the mass of the vehicle and inertias of the vehicle was used. A plug-in was created to accept motor thrust commands and by using the dynamic equations of motion (which are provided in Chapter 4) the force and moment vectors acting on the body were calculated. This is a solution in which the motion of the vehicle is based on the mathematical model derived in Chapter 4 and can be verified, but still uses the messaging protocols of Gazebo to connect to the other open-source software packages.

3.3 SITL Simulation

The first type of simulation used to test the different components was software-in-the-loop (SITL) simulations. One of the control board architectures for which PX4 can be configured, is a Linux-based operating system. Therefore, PX4 can run on a stand-alone desktop computer in exactly the same manner as it would run on an STM32 chip. The only difference is that for SITL simulations, the controllers and estimator are set to simulation mode and the communication protocols used are slightly different.

A model of the Intel® Aero RTF drone was created and placed in a Gazebo environment with ArUco markers. The model contains a camera mounted on the front of the vehicle with all the offsets measured from the actual platform. The Gazebo camera streams a generated image of the camera's view as a ROS topic at 14 Hz and the rest of the generated sensors' measurements are sent straight to PX4. A representation of the software connections is shown in Figure 3.4.

Stellenbosch University https://scholar.sun.ac.za

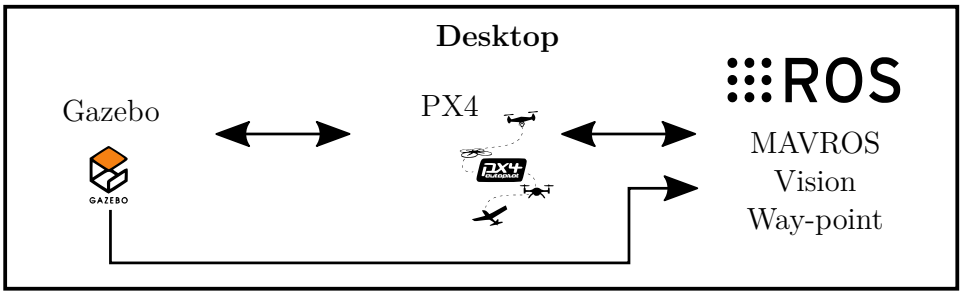

Figure 3.4: SITL simulation setup

The OBC has Ubuntu 16.04 LTS as an operating system and ROS installed on it. Therefore, there is no difference between ROS running on the stand-alone desktop computer or on the OBC. Both the waypoint scheduler and the vision-based pose estimation algorithm could be tested along with the rest of the software.

A recent update of Gazebo allows for lockstep between Gazebo and the PX4 software in both SITL and HITL simulations. "Lockstep" is where the controller waits for the sensor readings before executing and the physics simulator waits on the controller for motor commands. The simulation time is therefore independent of actual time. The simulations could be executed in less time than actual time would pass, or more time if the computational power was a problem. But this does not apply to all the sensors: the most notable sensor excluded from lockstep by Gazebo, was the on-board camera.

3.4 HITL setup

The second type of simulations used is HITL simulations. A HITL simulation generally involves uploading all the software on the hardware and connecting it to a simulation computer. The computer runs the physics simulator that generates sensors' measurements and receives the motor commands. This simulation tests the software in an embedded/on-board environment and ensures that each component works as designed. The ideal HITL simulation for this project would be the system shown in Figure 3.5.

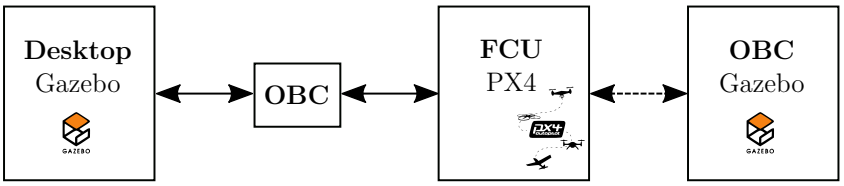

Figure 3.5: HITL simulation setup using the GPS-based state estimator

PX4 is loaded on the FCU. The OBC has MAVROS, the visual pose estimation node, and the waypoint scheduler loaded on it. Gazebo is then running on a stand-alone computer connected directly to the FCU and the OBC. Motor commands are received from the FCU and sensor measurements are sent to it and the images are streamed to the OBC. This setup will then test the full integration of the entire system, since the OBC and the FCU are not aware that it is not an actual flight. The FCU does not have any external ports to which a computer can connect. Therefore, all connections had to go through the OBC.

Stellenbosch University https://scholar.sun.ac.za

Consequently, the HITL simulations for this project were done in two steps. Firstly, the estimator and control system were tested using a simulated GPS. Secondly, the visual odometry system was tested with the estimator. The splitting of simulations was necessary because of the high computational load placed on the OBC when Gazebo and ROS are running on it. The other option was to use the OBC as a bridge between the stand-alone computer and the FCU. ROS could not be run on the OBC, because the bridge used blocked any access from the OBC to the FCU. The two different options are shown in Figure 3.6.

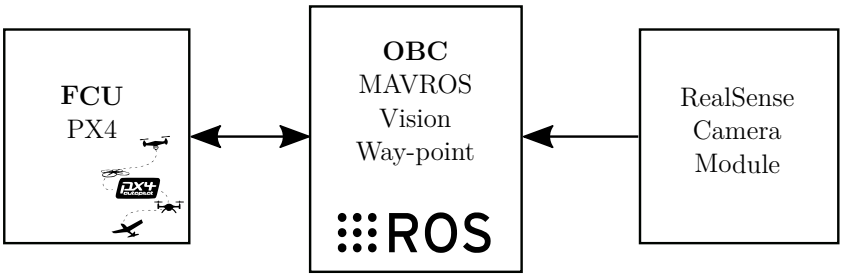

Figure 3.6: HITL simulation setup using the vision-based state estimator

The first simulation tests showed that the estimator and control system work when receiving sensor readings and that the logging does not slow the process down. The second simulation tests did not have simulated images streamed, but the actual cameras and IMU were used when the vehicle was picked up and moved manually in front of the markers. The visual pose estimation's accuracy will be discussed in Chapter 6. The estimator combines the IMU and visual estimates together in a timely manner for flight. Therefore, a flight test could safely be done, as the different systems were working with the estimator, overlapping in both simulations.

3.5 Flight test

PX4 was uploaded onto the FCU and MAVROS. The ArUco libraries and the waypoint scheduler was uploaded onto the OBC. Each software package used on the OBC has a separate script that activates it, so that the different components could be tested sequentially to incrementally build confidence in the system. The test procedures were done with GPS-based pose estimation and tested the control loops sequentially and thereafter the waypoint scheduler. The Visual Inertial Odometry (VIO) system was activated and the estimator with vision was tested. Finally, the full system with vision-based localisation was tested. The results of the flight test will be provided in Chapter 8.

3.6 Summary

The Intel® Aero Ready-To-Fly drone was chosen as the flying platform for this project. PX4 was uploaded onto the FCU and was responsible for the control system and state estimation. The companion computer of the Intel® Aero RTF drone was used as the OBC and had Ubuntu 16.04 LTS as an operating system. ROS, a MAVROS node, the

Stellenbosch University https://scholar.sun.ac.za

waypoint scheduler, and the visual pose estimation node was installed on the OBC. QGroundControl was the software used for the ground station and connected to the vehicle over a Wi-Fi connection. Gazebo was used for the simulation testing. Due to a physical wiring restriction, a full system HITL simulation could not be achieved and two separate simulations were done to test the system.

Stellenbosch University https://scholar.sun.ac.za

Chapter 4

Aircraft Dynamics

This chapter establishes a mathematical model for the quadrotor UAV flight mechanics that can be used for the flight control design and analysis, and as the basis for a simulation model. First, the various axis systems that are used by the PX4 flight control software, the Gazebo simulation software, and the camera system are defined. The standard notation for the aircraft variables are also introduced. Next, the differential equations that describe the six-degrees-of-freedom motion of a general aircraft are presented, followed by the force and moment models that are specific to a quadrotor UAV. The force and moment models include models for the actuators (rotor thrusts), the aerodynamics, and the gravitational force. The full quadrotor model is then summarised, and a linear model is derived to serve as the basis for the control system design and analysis. Finally, the vehicle parameters for the Intel® Aero RTF are determined.

4.1 Axes Systems

The mathematical models used to model the motion of a vehicle in the air require a frame of reference. As different open-source software packages were used, different axes systems are involved, namely: PX4's earth and body axes, Gazebo's earth and body axes, and the camera's earth and body axes. PX4's axis systems were considered as the reference axis for all the axes systems and the estimator and controller were therefore defined in terms of PX4's body and earth axes. Therefore, in this chapter "the body axes" refers to PX4's body axes and "the earth axes" refers to PX4's earth axes. The axes are shown in Figures 4.1, 4.2 and 4.3.

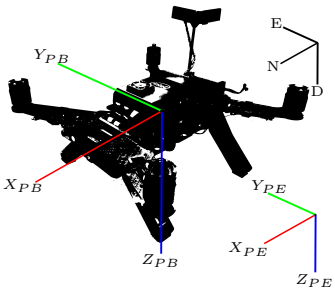

Figure 4.1: PX4 axes system

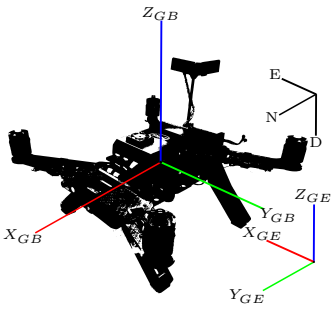

Figure 4.2: Gazebo axes system

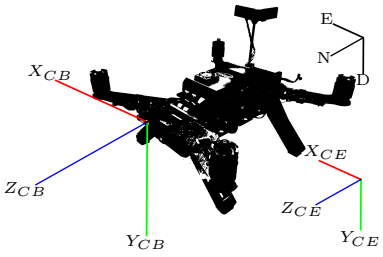

Figure 4.3: Camera axes system

Stellenbosch University https://scholar.sun.ac.za

4.1.1 PX4 Earth Axes

An earth axis system, also commonly referred to as an inertial axis system, is required to apply any of Newton's laws of motion. The PX4 earth axis system serves as the base to which all the other axes systems are measured. In this case, it is assumed that the earth is flat and not rotating, because a small UAV is used and is intended for short-range flights. The orientation follows the North East Down (NED) convention, where the origin of the axis system is placed at a convenient position, generally the take-off location. The x-axis coincides with the North direction, the y-axis with the East direction and the z-axis in the Down directions.

4.1.2 PX4 Body Axes

A body axis system is generally fixed to the aircraft body and its origin is chosen to be at the center of gravity (CoG). The x-axis points forward between the two front motors, the y-axis is positioned right between the two side motors and the z-axis faces downwards, relative to the vehicle. The rotation and translation of the body axis is expressed relative to the PX4 earth axis.

4.1.3 Gazebo Earth Axes

The Gazebo earth axes (also the simulation axes) is similar to the PX4 earth axes, as it is fixed to the world and non-moving. The difference is the orientation of the axis system. Instead of using a NED convention, an East North Up (ENU) convention is used. The origin is placed at the origin of PX4's earth axes. However, the y-axis corresponds to the x-axis of PX4, the x-axis corresponds to the y-axis of PX4, and the z-axis points in an upwards direction. It is therefore termed "Gazebo Earth Axes" as the earth axes is utilised by Gazebo and ROS.

4.1.4 Gazebo Body Axes

These body axes (also the simulation axes) coincide with the PX4 body axes, but also uses an ENU convention, where the x-axis points forward, the y-axis to the left, and the z-axis upward relative to the vehicle. The orientation of the Gazebo body axes is expressed relative to the Gazebo Earth axes.

4.1.5 Camera Earth Axes

The camera earth axes origin coincides with both of the other two earth axes. However, it uses an East Down North (EDN) convention, where the x-axis is in the East direction, the y-axis is in the Down direction, and the z-axis is in the North direction. This strange orientation is the result of how OpenCV defines its axis system.

4.1.6 Camera Body Axes

The camera body axes' origin is placed at an offset from the other two body axes. Therefore, it is in the position of the lens relative to the CoG (xyz = [0.12, 0.038, 0.0] metre in PX4 body axis). The orientation of the axes follows that of the camera's earth axis, where the x-axis is to the right, the y-axis is pointing downward, and the z-axis is

Stellenbosch University https://scholar.sun.ac.za

pointing forward. The pose estimate of the visual odometry system is expressed in this axis system and is given relative to the camera's earth axes.

Image Axis

There exists an image plane which is the projection of the 3D environment onto the camera. This plane is measured relative to the camera body axis and, while it is not indicated in Figure 4.3, it will be discussed and used in Chapter 6.

4.2 Notation

The following standard notation is used for the variables that are used in the quadrotor UAV model.

X,Y,Z Coordinates of the force vector in body axes (axial, lateral, and normal force)

L,M,N Coordinates of the moment vector in body axes (rolling, pitching, and yawing moment)

N,E,D: Coordinates of the position vector in earth axes

U,V,W Coordinates of the linear velocity vector in body axes (axial, lateral, and normal velocity)

ϕ**,**θ**,**ψ**:** The Euler 3-2-1 attitude vector of the body axis system with respect to the earth axis system

α_i**:** The quaternion describing the rotation of the body axis system with respect to the earth axis system, where i is from w to z

P,Q,R Coordinates of the angular velocity vector in body axes (roll, pitch, and yaw rate)

$\delta_{\mathbf{A}}$**,**$\delta_{\mathbf{E}}$**,**$\delta_{\mathbf{R}}$ Virtual aileron, elevator and rudder control

As discussed in the Chapter 2.1, a quadrotor does not have any control surfaces (ailerons, elevators and rudders) like aeroplanes. The mathematical models and control laws for these control surfaces have been extensively researched and are well understood. Therefore, a virtual control surface was defined, where the motors were grouped together so that the virtual actuator deflection on the quadrotor vehicle would have a similar effect as a physical actuator deflection on a fixed-wing aircraft. The exact conventions and mathematical model used is given in Section 4.5.1.

Stellenbosch University https://scholar.sun.ac.za

4.3 Aircraft Dynamics Overview

The block diagram in Figure 4.3 provides the full nonlinear aircraft dynamics model that will be described in the following sections.

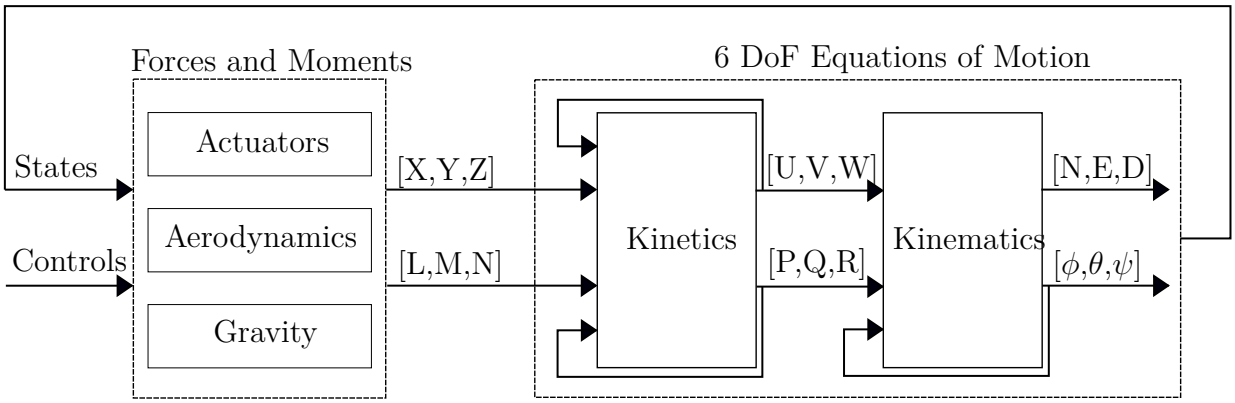

Figure 4.4: Block diagram overview of aircraft dynamics

The right-hand side contains the 6DOF equations of motion that describe the motion of a rigid body in 3D space, given a force and moment vector. The left-hand side models the forces and moments that act on the rigid body, given a commanded virtual actuator deflections (aileron, elevator, rudder, and thrust) and the current state of the rigid body.

4.4 Six-Degrees-of-Freedom Equations of Motion

Now that the aircraft variables are defined and an overview was provided, the equations of motion for the quadrotor can be derived. The quadrotor is subject to six-degrees-of-freedom when moving in a 3D-space - this being a translation along the NED axes and a rotation around each axis. The modelling of the quadrotor in 6DOF was done by using two fields of dynamics (a subset of mechanics), namely kinetics and kinematics, where Hebbeler [4] (pp. 3-6) defines kinetics and kinematics as follows:

Kinetics The analysis or study of motion caused by forces.

Kinematics The study of the geometric aspects of motion, such as acceleration, velocity and position.

In the following Sections it is assumed that the quadrotor is a rigid body, which implies that its CoG remains fixed relative to the body axis.

4.4.1 Kinetics

The derivation steps for the kinetic equations of motion used for this model are provided in the textbook by J.H. Blakelock[1]. Blakelock uses Newton's second Law of Motion, which states that all forces acting on an object must equate to zero, assuming zero acceleration. The aircraft is symmetrical about its XZ-plane and YZ-plane. Consequentially

Stellenbosch University https://scholar.sun.ac.za

it is assumed that the cross product of the moment of inertias (I_{xy}, I_{yz} and I_{xz}) all equate to zero. Therefore, the equations derived by Blakelock are simplified into the following:

$$X = m(\dot{U} - VR + WQ) \tag{4.4.1}$$
$$Y = m(\dot{V} + UR - WP) \tag{4.4.2}$$
$$Z = m(\dot{Q} - UQ + VP) \tag{4.4.3}$$
$$L = \dot{P}I_{xx} + QR(I_{zz} - I_{yy}) \tag{4.4.4}$$
$$M = \dot{Q}I_{yy} + PR(I_{xx} - I_{zz}) \tag{4.4.5}$$
$$N = \dot{R}I_{zz} + PQ(I_{yy} - I_{xx}) \tag{4.4.6}$$

where m is the quadrotor's mass, and I_{xx}, I_{yy} and I_{zz} are the principal moments of inertia about the respective body axes.

4.4.2 Kinematics

According to Hibbeler[4], at every time instant, the kinematics of a particle is specified by its position, velocity and acceleration. Both linear and angular motion are present in the case of a quadrotor, but for this project angular acceleration was not considered. Both Euler angles and quaternions were used to represent orientation. A brief overview of each follows and also of the position and orientation dynamics.

Euler Angles

The attitude of an aircraft can be described using one of three main parameterising techniques, namely Euler angles, Quaternions and Directional Cosine Matrix (DCM). Each of the techniques have advantages and disadvantages. Quaternions and the DCM parametrisation are mathematically more complex and are not as intuitive as Euler angles, while Euler angles have the problem of a singularity when one of the angles reaches a 90° rotation. This problem is generally solved by assuming that, during regular flight, the pitch angle will never reach the singularity angle. Throughout this thesis an attitude parametrisation of 3-2-1 is used and it refers to the order of rotation which is:

1. Yaw: the axis is firstly rotated through the heading angle (ψ) around the D-axis.
2. Pitch: the axis is secondly rotated through the pitching angle (θ) around the body's new E-axis.
3. Roll: the axis is lastly rotated through the rolling angle (ϕ) around the body's new N-axis.

Euler angles are therefore intuitive and easily understood.

Quaternions

The other attitude parameterisation used in this thesis is quaternions, specifically unit quaternions. Like Euler angles, quaternions are descriptions of the rotation from one

Stellenbosch University https://scholar.sun.ac.za

axis system to another, but lacks the limitation of the specific order of operation like Euler angles. Quaternions are commonly provided in the general form:

$$\bar{\alpha} = \begin{bmatrix} \alpha_w & \alpha_x \underline{\mathrm{i}} & \alpha_y \underline{\mathrm{j}} & \alpha_z \underline{\mathrm{k}} \end{bmatrix}. \tag{4.4.7}$$

where α_w, α_x, α_y and α_z are real numbers and $\underline{\mathrm{i}}$, $\underline{\mathrm{j}}$ and $\underline{\mathrm{k}}$ are the quaternion units.

A more visual definition of quaternions can be seen when converting it from an axis angle representation. Axis angle is a way of representing an orientation by the use of a three element vector and an angle of rotation around the vector shown in Figure 4.5.

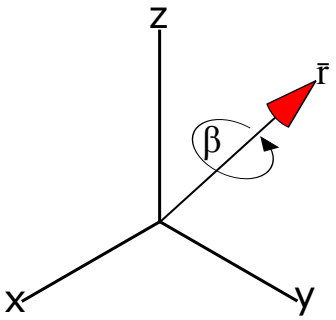

Figure 4.5: Axis angle representation

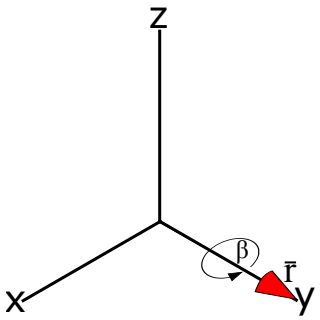

Figure 4.6: Special case of axis angle

Given that $\bar{r} = [r_x \ r_y \ r_z]^T$ the quaternion equivalent is:

$$\bar{\alpha} = \begin{bmatrix} \alpha_w \\ \alpha_x \\ \alpha_y \\ \alpha_z \end{bmatrix} = \begin{bmatrix} \cos(\beta/2) \\ r_x \sin(\beta/2) \\ r_y \sin(\beta/2) \\ r_z \sin(\beta/2) \end{bmatrix}. \tag{4.4.8}$$

where β is the angle around the $\bar{r}$ vector. However, if the vector is only in one dimension (i.e. coincides with one of the axes) then β is directly equal to the Euler angle representative around that axis as shown in Figure 4.6.

Attitude Dynamics

Even though Euler angles and quaternions serve the same purpose, this project made use of both because PX4's controllers use quaternions. Euler angles were used to design the controller, since (when linearised) Euler angles are double the quaternion equivalent (as shown in Section 4.6) and Euler angles are simpler. The equation describing the relationship between the angular velocity in the body axis to earth axis, as derived by Etkin and Reid [3] (pp. 100), is provided below:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix}, \text{where } |\theta| \neq \pi/2 \tag{4.4.9}$$

A singularity occurs when the pitch angle equals 90°. However, Euler angles are only used on the linearised model, where small angles are assumed (as Section 4.6 will explain) and are therefore not a concern, as quaternions are used for the non-linear model.

Stellenbosch University https://scholar.sun.ac.za

Position Dynamics

The last part of the kinematic model is the linear velocity measured in the earth axis. Etkin and Reid[3] (pp. 104) derived a relation between these variables expressed by the following equation:

$$\begin{bmatrix}\dot{N}\\ \dot{E}\\ \dot{D}\end{bmatrix} = \begin{bmatrix}C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi\\ S_\theta C_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi\\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta\end{bmatrix}\begin{bmatrix}U\\ V\\ W\end{bmatrix} \begin{matrix}C_\# = \cos(\#)\\ S_\# = \sin(\#)\end{matrix} \tag{4.4.10}$$

Should the reader require be interested in the derivation, Möller's master's thesis[5] contains a detailed approach.

4.5 Forces and Moments

Besides the dynamics of the quadrotor, which were discussed above, there are forces and moments acting upon the quadrotor. These forces and moments are caused by the actuators (i.e. motors and propellers) and environmental effects (i.e. gravity and aerodynamics). The ground and consequently ground effects also cause forces and moments, but for this model they were not taken into consideration. This section will expand on the importance and function of these forces and moments acting on the quadrotor.

4.5.1 Actuators

The first of these forces and moments acting on the body is that of the actuators (motor and propeller pair). As a quadrotor in a cross orientation (see Section 2.1) was used, all the definitions are for a cross configuration. The forces and moments acting on the quadrotor was due to the different actuators running at different speeds. The largest influence on the maneuver ability of a quadrotor is arguably the time delay (lag) of each actuator's to respond to a command. The effect of a time delay was modelled as a simple first order equation as seen below:

$$\dot{T} = (T_c - T)/\tau \tag{4.5.1}$$

where T_c is the commanded thrust, T is the actual thrust and τ is the time delay of a single actuator.

The motors were fixed in place and constant pitch propellers were used. Consequently, the forces caused by the actuators were only in the negative z-axis (of the body). However, varying speeds on each actuator caused moments around the three axes. The forces and moments due to actuators are given as:

$$Z^T = -\delta_T \tag{4.5.2}$$
$$L^T = d_{\text{arm}_y}\delta_A \tag{4.5.3}$$
$$M^T = d_{\text{arm}_x}\delta_E \tag{4.5.4}$$
$$N^T = \frac{r_D}{R_{LD}}\delta_R \tag{4.5.5}$$

Stellenbosch University https://scholar.sun.ac.za

where d_{arm_j} is the distance from the actuator centre point to CoG, r_D is the rotor drag coefficient, and R_{LD} is the lift to drag ratio.

As mentioned in Section 4.2, a quadrotor does not have control surfaces. However, a mixing matrix is used that maps the virtual actuator commands to physical rotor thrust commands. The mixing matrix is defined as:

$$\begin{bmatrix} \delta_T \\ \delta_A \\ \delta_E \\ \delta_R \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \tag{4.5.6}$$

where the $\frac{1}{\sqrt{2}}$ is due to the geometry, since the d_{arm_i} is measured from the actuator center to the CoG and the moment is only proportional to the distance in either the x-axis or the y-axis. The cross-configuration's x-axis and y-axis components are therefore $d_{arm_i} \cos(45°)$.

4.5.2 Aerodynamics

One of the environmental effects acting on the vehicle is aerodynamics. The aerodynamic model that was used was a simple application of the dynamic drag due to the frontal area. The equation describes the drag force experienced by an object moving through a liquid and is represented by:

$$F_D = -a_D \cdot v^2 \tag{4.5.7}$$

where

$$a_D = \frac{1}{2}\rho C_D A_i \tag{4.5.8}$$

where ρ is the air density, v is the linear velocity of the object, C_D is the drag coefficient, and A_i is the reference area.

The equations assumed a blunt form factor. It was assumed that the projected area in all directions was 1 m^2 and that the error due to this assumption was absorbed by the drag coefficient. The resultant force produced by aerodynamics drag was proportional to the velocity of the quadrotor through the air and can be expressed as:

$$\begin{bmatrix} X^A \\ Y^A \\ Z^A \end{bmatrix} = -a_D \begin{bmatrix} U|U| \\ V|V| \\ W|W| \end{bmatrix} \tag{4.5.9}$$

4.5.3 Gravity

The last environmental effect affecting the vehicle is gravity. The resultant force derived from gravity, at stable hover, will merely be the weight acting on the body in the downwards direction and was expressed as:

$$\begin{bmatrix} X^G \\ Y^G \\ Z^G \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} mg \tag{4.5.10}$$

Stellenbosch University https://scholar.sun.ac.za

where m is the mass of the vehicle and g is the gravitation acceleration constant for earth. When the quadrotor rotates, equation (4.5.10) is also rotated. This rotation can be described by the DCM given in equation (4.4.10) and the resultant equation is:

$$\begin{bmatrix} X^G \\ Y^G \\ Z^G \end{bmatrix} = \begin{bmatrix} -sin(\theta) \\ cos(\theta)sin(\phi) \\ cos(\theta)cos(\phi) \end{bmatrix} mg \tag{4.5.11}$$

There were no resultant moments due to gravity, because gravity only acts through the quadrotor's CoG.

4.6 Linearisation

The full nonlinear aircraft system allows for accurate simulations, but a linearised, time-invariant model is required for control system design and analysis using linear control theory. Therefore, this section focuses on linearising the nonlinear model around hover.

4.6.1 Linearising about Trim

The nonlinear dynamics equations of interest can be represented in the state space form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{4.6.1}$$

where:

$$\mathbf{x} = \begin{bmatrix} \delta_T & \delta_A & \delta_E & \delta_R & U & V & W & \Phi & \Theta & \Psi & P & Q & R \end{bmatrix}^T \tag{4.6.2}$$

$$\mathbf{u} = \begin{bmatrix} \delta_{T_R} & \delta_{A_R} & \delta_{E_R} & \delta_{R_R} \end{bmatrix}^T \tag{4.6.3}$$

and $\mathbf{u}$ is the virtual control surface reference commands. The states can be defined as the sum of the trim values and small deviations around trim. Therefore, small angle approximation was used. As the state deviations are assumed to be small, multiplication of two near-zero states result in an insignificantly small value and can be ignored. The trim condition for a quadrotor in this project was chosen as hover. Therefore, the states were re-defined as:

$$\mathbf{x} = \mathbf{x}_{Trim} + \mathbf{\Delta x} \tag{4.6.4}$$

$$\mathbf{u} = \mathbf{u}_{Trim} + \mathbf{\Delta u} \tag{4.6.5}$$

while:

$$\mathbf{x}_{Trim} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.6.6}$$

$$\mathbf{u}_{Trim} = \begin{bmatrix} T_{hover} & 0 & 0 & 0 \end{bmatrix}^T \tag{4.6.7}$$

$$\mathbf{\Delta x} = \begin{bmatrix} \delta_t & \delta_a & \delta_e & \delta_r & u & v & w & \phi & \theta & \psi & p & q & r \end{bmatrix}^T \tag{4.6.8}$$

$$\mathbf{\Delta u} = \begin{bmatrix} \delta_{t_R} & \delta_{a_R} & \delta_{e_R} & \delta_{r_R} \end{bmatrix}^T \tag{4.6.9}$$

Stellenbosch University https://scholar.sun.ac.za

where T_{hover} is the total thrust required at hover. The notable 6 DoF nonlinear equations with their linearised counter parts are summarised in table 4.1.

Standard Form	Linearised Form
$\dot{\delta_T} = \frac{\delta_{T_R}}{\tau} - \frac{\delta_T}{\tau}$	$\dot{\delta_t} = \frac{\delta_{t_R}}{\tau} - \frac{\delta_t}{\tau}$
$\dot{\delta_A} = \frac{\delta_{A_R}}{\tau} - \frac{\delta_A}{\tau}$	$\dot{\delta_a} = \frac{\delta_{a_R}}{\tau} - \frac{\delta_a}{\tau}$
$\dot{\delta_E} = \frac{\delta_{E_R}}{\tau} - \frac{\delta_E}{\tau}$	$\dot{\delta_e} = \frac{\delta_{e_R}}{\tau} - \frac{\delta_e}{\tau}$
$\dot{\delta_R} = \frac{\delta_{R_R}}{\tau} - \frac{\delta_R}{\tau}$	$\dot{\delta_r} = \frac{\delta_{r_R}}{\tau} - \frac{\delta_r}{\tau}$
$\dot{U} = X/m + VR - WQ$	$x = \dot{u}m$
$\dot{V} = Y/m - UR + WP$	$y = \dot{v}m$
$\dot{W} = Z/m + UQ - VP$	$z = \dot{w}m$
$\dot{\Phi} = P + Q\sin(\phi)\tan(\theta) + R\cos(\phi)\tan(\theta)$	$\dot{\phi} = p$
$\dot{\Theta} = Q\cos(\phi) - R\sin(\phi)$	$\dot{\theta} = q$
$\dot{\Psi} = Q\sin(\phi)\sec(\theta) + R\cos(\phi)\sec(\theta)$	$\dot{\psi} = r$
$\dot{P} = L/I_{xx} - QR(I_{zz} - I_{yy})/I_{xx}$	$l = \dot{p}I_{xx}$
$\dot{Q} = M/I_{yy} - RR(I_{xx} - I_{zz})/I_{yy}$	$m = \dot{q}I_{yy}$
$\dot{R} = N/I_{zz} - PQ(I_{xx} - I_{yy})/I_{zz}$	$n = \dot{r}I_{zz}$
$\alpha_w = \cos(\beta/2)$	$\alpha_{w,l} \approx 1$
$\alpha_x = r_1\sin(\beta/2)$	$\alpha_{x,l} \approx \phi/2$
$\alpha_y = r_2\sin(\beta/2)$	$\alpha_{y,l} \approx \theta/2$
$\alpha_z = r_3\sin(\beta/2)$	$\alpha_{z,l} \approx \psi/2$

Table 4.1: Linearised 6DOF equations Of motion

Similarly, the equations of the forces and moments were linearised around hover. The nonlinear and linear equations are summarised in table 4.2.

Standard Form	Linearised Form
$X = -F_D U^2 - mg\sin(\theta)$	$x = -mg\theta$
$Y = -F_D V^2 + mg\sin(\phi)$	$y = mg\phi$
$Z = -F_D W^2 + mg - \delta_T$	$z = mg - \delta_t$
$L = d_{\text{arm}_y}\delta_A$	$l = d_{\text{arm}_y}\delta_a$
$M = d_{\text{arm}_x}\delta_E$	$m = d_{\text{arm}_x}\delta_e$
$N = \delta_R r_D/R_{LD}$	$n = \delta_r r_D/R_{LD}$

Table 4.2: Linearised forces and moment

By combining the linear equations of both tables 4.1 and 4.2, the linearised state space can be calculated and simplified to:

$$\dot{\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{u} \tag{4.6.10}$$

or when expanded shown to be:

Stellenbosch University https://scholar.sun.ac.za

$$\boldsymbol{\Delta\dot{\mathbf{x}}} = \begin{bmatrix} -\frac{\delta_t}{\tau} \\ -\frac{\delta_a}{\tau} \\ -\frac{\delta_e}{\tau} \\ -\frac{\delta_r}{\tau} \\ -g\theta \\ g\phi \\ \frac{\delta_t}{m} \\ p \\ q \\ r \\ \frac{d_{\text{arm}_y}\delta_a}{I_{xx}} \\ \frac{d_{\text{arm}_x}\delta_e}{I_{yy}} \\ \frac{\delta_r r_D}{I_{zz} R_{LD}} \end{bmatrix} + \begin{bmatrix} \frac{\delta_{t_R}}{\tau} \\ \frac{\delta_{a_R}}{\tau} \\ \frac{\delta_{e_R}}{\tau} \\ \frac{\delta_{r_R}}{\tau} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.6.11}$$

This linearised model was used for the controller design that will be presented in Chapter 5.

4.7 Vehicle Parameters

The physical parameters of the vehicle is the final requirement for a full aircraft dynamic model. The parameters used to model the vehicle are shown in table 4.3. The mass was measured with a digital scale, the motor arms where measured with a ruler, the rotor drag was read off the propeller's data sheet, the lift to drag coefficient was estimated, and an overview of how the moments of inertia were calculated is given below.

Property	Symbol	Value	Unit
Mas	m	1.190	kg
Principle moment of inertia around x-axis	I_{xx}	0.014	$kg \cdot m^2$
Principle moment of inertia around y-axis	I_{yy}	0.022	$kg \cdot m^2$
Principle moment of inertia around z-axis	I_{zz}	0.032	$kg \cdot m^2$
Motor moment arm in x-axis	d_{arm_x}	0.180	m
Motor moment arm in y-axis	d_{arm_y}	0.180	m
Rotor drag coefficient	r_D	0.437	No Unit
Rotor lift to drag coefficient	R_{LD}	10.000	No Unit
Aerodynamics Drag coefficient	C_D	0.025	No Unit
Body reference Area	A_i	1.000	m^2

Table 4.3: Quadrotor's physical properties

4.7.1 Mass Moment of Inertia Experiment

The identification of the moments of inertia was performed using a simple experiment. Treurnicht [6] describes an experiment for calculating the moment of inertia, where the object is suspended from two ropes. The axis around which the inertia is to be measured is placed perpendicular to the horizon and parallel to the ropes. The object is rotated slightly and allowed to oscillate. A sketch of the experiment is shown in Figure 4.7.

Stellenbosch University https://scholar.sun.ac.za

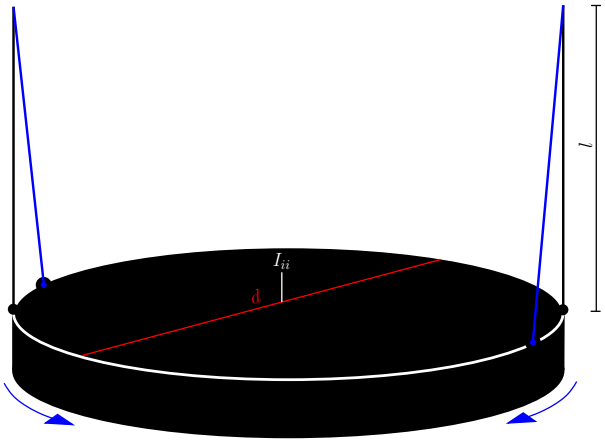

Figure 4.7: Solid disk representation of the experiment

The equation to determine the moment of inertia is as follows:

$$I_i = \frac{m \cdot g \cdot d^2 \cdot t_p^2}{16\pi^2 \cdot l} \tag{4.7.1}$$

where m is the mass, g is the gravitational acceleration on earth, d is the distance between the two parallel ropes, l is the length of the ropes, and t_p is the period of the oscillation.

The experiment was repeated 3 times per axis for all three axes. The result of each experiment is provided below and the final values are given in table 4.4.

Inertia	d[m]	l[m]	$t_{p,1}$[s]	$t_{p,2}$[s]	$t_{p,3}$[s]
I_{xx}	0.23	0.81	1.702	1.667	1.737
I_{yy}	0.25	0.77	1.883	1.917	1.933
I_{zz}	0.24	0.77	2.383	2.424	2.413

Table 4.4: Mass moment of inertia experiment results

Figure 4.8 shows how the vehicle was attached to determine the I_{zz} variable.

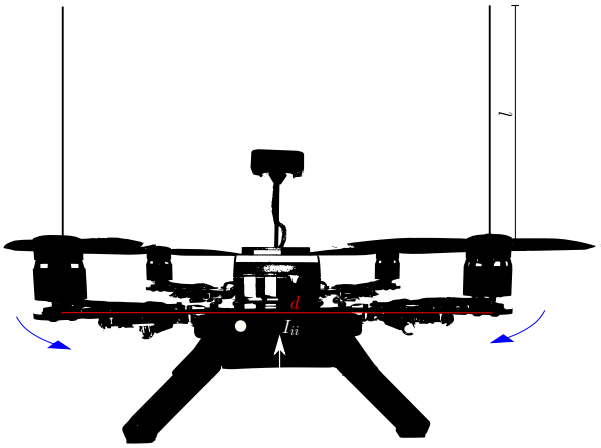

Figure 4.8: Experimental setup for system identification of the z-axis moment of inertia

Stellenbosch University https://scholar.sun.ac.za

4.8 Summary

This chapter established a mathematical model for the quadrotor UAV flight mechanics. The various axis systems that are used by PX4 flight control and state estimation software, the Gazebo simulation software, and the camera system were defined. An overview of the standard notation used for the aircraft variables were provided. Thereafter, the differential equations that the describe the six-degrees-of-freedom of the aircraft was presented and the force and moment model specific to the quadrotor UAV. The full quadrotor model was summarised, and a linear model was derived to serve as aid in the control system design and analysis. The final part of this chapter was the determination of the vehicle parameters for the Intel® Aero RTF drone, which included an experiment to estimate the moment of inertias.

Stellenbosch University https://scholar.sun.ac.za

Chapter 5

Control System Analysis and Design

PX4 is the control system used and what follows here is an overview of PX4's control architecture, the structure used to design the controllers and the tests that verified the design process.

PX4's control system can be uploaded on a wide range of vehicles because the controllers are normalised and are therefore not bound to a single airframe. The commands are scaled to the appropriate size at a vehicle level. To design controller gains specifically for the Aero, the quadcopter's physical properties had to be considered and a control structure similar to PX4's was created. The structure included the normalised controller laws which was denormalised and made use of the linearised vehicle dynamics determined in Chapter 4 to simplify the equations. The control structure consists of four control loops and each control loop was designed individually. Each loop will be given an overview and the detailed design process will be shown. The loops were all tested through SITL simulations to verify that the behaviour was satisfactory.

5.1 Overview of PX4 Architecture

The PX4 flight control architecture uses a successive loop closure approach, with four layers of control loops, from the fastest inner-loop controllers to the slowest outer-loop controllers. The control loops from innermost to outermost are the angular rate controllers, the attitude controllers, the velocity controllers, and the position controllers. A block diagram depicting the control architecture used by PX4 is shown in Figure 5.1:

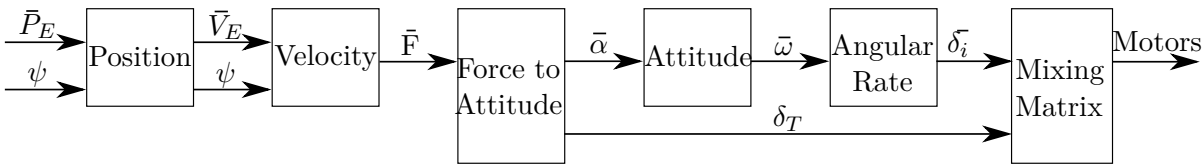

Figure 5.1: PX4 controller overview

where $\bar{P}_E$ is the positions in earth axis, ψ is the heading angle, $\bar{V}_E$ is the linear velocity in the earth axis, $\bar{F}$ is a force vector, ω is the angular rates in body axis and δ_i is the virtual control surface commands (δ_A, δ_E, δ_R). The position control loop receives the local position from a GPS or vision-based localisation system and controls the vehicle position to follow a commanded position reference by actuating the velocity controller references.

Stellenbosch University https://scholar.sun.ac.za

The velocity controllers controls the measured velocity to follow a commanded velocity reference by actuating a force vector reference. which is in turn converted to an attitude reference for the attitude controllers and a thrust command. The conversion is done by multiplying the acceleration by the mass of the vehicle, since $F = ma$. The force vector then points in the direction which the motors are required to provide thrust. Therefore, the thrust command is the magnitude of the vector and the attitude set-point is the orientation of the plane perpendicular to the vector. The heading angle is assumed as the rotation around the force. The attitude of the vehicle is obtained from a GPS-based or vision-based state estimator, which estimates the full vehicle state using a fusion of on-board sensor measurements, which include inertial sensor measurements.The attitude controllers control the vehicle attitude to follow a commanded attitude reference by actuating the angular rate references for the angular rate controllers. The final and fastest loop actuate the motor thrusts to control the angular rate references. PX4 abstracts the motor thrusts from the controllers by implementing a virtual control surface, mapping it to the motors. Therefore, the angular rate loop controls the virtual aileron, elevator and rudder. The conversion from acceleration to attitude and the abstraction of the direct control of the motor thrust introduces some uncertainty into the control system, which could cause unwanted behaviours and introduces unpredicted disturbances. It is assumed that PX4 needs a controller that is good at disturbance rejection and that reduces the influence of the references on the control signals. Therefore, PX4 chooses to control these loops with a two-degrees-of-freedom proportional integral and derivative (2 DoF PID) controller, instead of the normal PID controllers.

The degrees of freedom referred to in the name of this controller type is the number of closed-loop transfer functions that are capable of changing independently [9]. Araki and Taguchi [8] made comparisons between the two types of PID controller and noted that the 2DoF was better at reducing the influence of the reference (set-point) signal on the control signal and better at retaining the overshoot, while also increasing the disturbance rejection. These improvements are due to the weight that was added to the set-points of the proportional and derivative terms, which results in a change in the form of the control law:

$$u = K_P(b \cdot r - y) + \frac{K_I}{s}(r - y) + \frac{K_D \cdot s}{T_f \cdot s + 1}(c \cdot r - y) \tag{5.1.1}$$

where r is the reference, u is the control signal, y is the current state, K_P is the proportional gain, K_I is the integral gain, K_D is the derivative gain, T_f is the derivative filter time, b is the proportional set-point weight and c is the derivative set-point weight.

The equation above is the standard form for a 2DoF PID controller. However, PX4 uses a special case where $b = 1$ and $c = 0$. These weights result in an equally weighted proportional and integral term, while the derivative term's weight causes the derivative term to ignore the reference. Therefore, the 2DoF PID controller is used to increase the reliability and robustness of these controllers. The position and attitude loops actuates the references to these finely controlled loops and do not require more than proportional control for satisfactory responses.

In addition to the control loops, PX4 uses normalised forces and thrusts throughout the controller design, that allows any vehicle type to use the controllers. The vehicle type (or air-frame model) is then required to denormalise the thrust before it is passed on to the actuators. This modular control system has been implemented by various hobbyist and developers and have shown that the control system is functional. The design of

Stellenbosch University https://scholar.sun.ac.za

the controllers that was used for this project was based on this architecture and will be explained in the next Section.

5.2 Controller Design Overview

Successive control loop closure with interchanged P and PID control is a sensible solution to control a quadrotor UAV. PID control is a suitable choice for an on-board control system with limited computational power, as it is easy to implement, is generally low on resources, it can be easily tuned while being robust to mismatched tuning and it has very good disturbance rejection. Utilizing only PID controllers for the critical loops (and P control on the other loops) is a logical choice because the system provides satisfactory control while not taking up more computation than necessary. The linearised aircraft model derived in Section 4.6 was used for the controller design. The simulations did not make use of the 2DoF PID controllers because the linear model (with normal 1 DoF control) provided a close representation of the 2DoF response. Additionally, the gains were also tuned with the SITL simulations and the flight tests. The designed gains are implemented directly in PX4's architecture. Therefore, the controller gains are designed in a normalised form.

The controllers were designed sequentially from the fastest inner loops to the slowest outer loops, namely angular rate controllers, attitude controllers, velocity controllers, and position controllers. Each control loop layer consists of three individual controllers, one for each axis that is controlled. For example, the three-axis angular rate control is performed using three separate controllers for roll rate, pitch rate, and yaw rate, and the three-axis position control is performed using three separate controllers for North position, East position, and Down position, respectively. The controllers are also categorised into the following four groups into which the linearised model is decoupled: longitudinal, lateral, heave, and heading. In most cases, identical control architectures and design procedures are used for all three axes. Where there are differences (as is the case for the velocity controllers), the differences will be highlighted and discussed.

In the following Sections, the detailed design of the controllers for the East axis will be presented, namely the roll rate controller, roll angle controller, East velocity controller, and East position controller. The architectures and design procedures for the North and Down directions are similar, and will therefore not be presented explicitly.

5.2.1 Angular Rate Controller

The angular rate controllers are the fastest and most critical for stability in the system. The controller receives an angular rate reference and commands a virtual deflector and/or virtual thrust to match the reference. The controllers receive feedback of the quadrotors orientation from the gyrometers. PX4's controllers use normalised control gains to allow the control architecture to be used with any type of vehicle. Therefore, the output commands sent to the actuators (which range from -1 to 1) by the normalised controllers are scaled for the design model to denormalised it before supplying it to the denormalised plant.

The block diagram of the roll rate controller is given in Figure 5.2. The plant for this controller is the linearised roll rate dynamics derived in Chapter 4 and represents the response from virtual aileron command to roll rate. The angular rate controllers of PX4

Stellenbosch University https://scholar.sun.ac.za

uses a 2 DOF PID controller, however a 1 DOF PID controller was chosen to design the gains with. The assumption was made that when a linearised model was used that the difference between 1 DOF and 2 DOF will be negligible and it simplifies the equations. The controller gains were tested through SITL simulations and if the responses of the 2 DOF PID controllers were not in specifications adjustments were made.

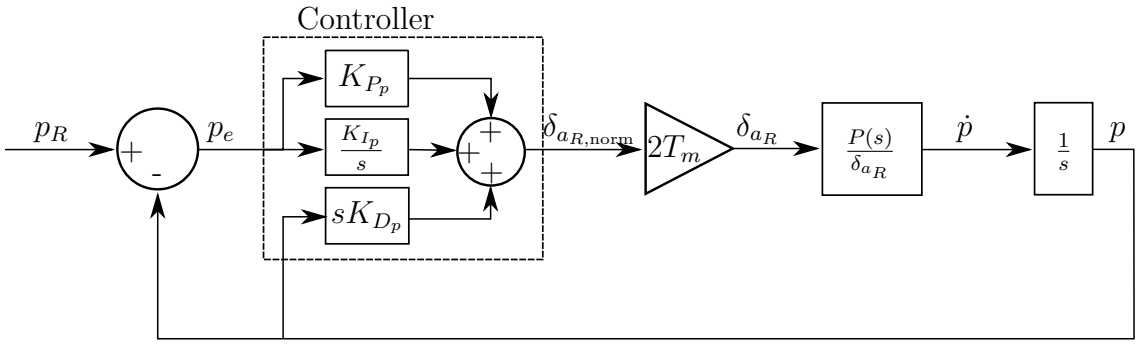

Figure 5.2: Roll rate control loop

The roll rate error p_e is calculated by subtracting the measured roll rate p from the roll rate reference p_R. The error is processed by the normalised controller to provide a normalised aileron command. The normalised aileron command is denormalised by the appropriate scaling factor before supplying it to the denormalised plant model. The time taken between aileron command to the actual aileron deflection is assumed to be a first-order response. The result in the linear model is the rate of change in the roll rate and requires integration to provide the next time step's roll rate.

The detailed design of the controllers were done by the combination of Bode plots and root locus design.

5.2.2 Roll Rate Gain Design

The angular rate controllers are primarily responsible for the stability and response time of the quadrotor. Therefore, the roll rate controller should (a) be as fast as possible (not faster than 33ms response time), (b) have a well damped response and (c) have steady-state error rejection.

The controller architectures and design procedures for the different axes are identical for all three angular rate components (roll, pitch and yaw rate). However, the controller gains may differ due to the different physical properties of the aircraft about each of the three axes.

The plant is a mathematical equation that provides a roll rate if an aileron command is given. The two equations that were used were given in Section 4.6 and their Laplace transforms are:

$$\delta_a(s) = \frac{\delta_{a_R}(s)}{(s + \frac{1}{\tau})\tau} \tag{5.2.1}$$

$$P(s) = \frac{d_{\text{arm}_y}}{I_{xx} \cdot s}\delta_a(s) \tag{5.2.2}$$

where τ is the motor's time constant, d_{arm_y} is the distance from the CoG to the motor in the Y-axis, and I_{xx} is the moment of inertia in the X-axis. When the above two

Stellenbosch University https://scholar.sun.ac.za

equations are combined then the plant's transfer function is derived:

$$\frac{P(s)}{\delta_{a_R}(s)} = \frac{d_{\text{arm}_y}}{\tau I_{xx} \cdot s(s + \frac{1}{\tau})} \tag{5.2.3}$$

The control law implemented by PX4 is in terms of the normalised control gains and is given in equation (5.2.4). This law will be denormalised to scale the commands like PX4s airframe modules does to ensure that the gains that are designed are directly implementable in PX4.

$$\delta_{a,\text{norm}}(t) = K_{P_p}(p_R - p) + K_{I_p}\int (p_R - p)dt + K_{D_p}\frac{d((-p)}{dt} \tag{5.2.4}$$

where p_R is the roll rate reference, $\delta_{a,\text{norm}}$ is the normalised aileron command, K_{P_p} is the proportional gain, K_{I_p} is the integral gain, and K_{D_p} is the derivative gain. The relationship between the normalised and denormalised aileron commands are required for denormalisation, which in turn requires the relationship between the normalised aileron command and the normalised rotor thrust. The PX4 flight control software uses the following mixing matrix to convert the four normalised rotor thrusts to the normalised total thrust, and the normalised virtual aileron, elevator, and rudder deflections:

$$\begin{bmatrix}\delta_t \\ \delta_a \\ \delta_e \\ \delta_r\end{bmatrix}_{norm} = \begin{bmatrix}1 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 1 \\ 1 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 1 \\ 1 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -1 \\ 1 & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -1\end{bmatrix}\begin{bmatrix}T_1 \\ T_2 \\ T_3 \\ T_4\end{bmatrix}_{norm} \tag{5.2.5}$$

The denormalised rotor thrust commands can be calculated by scaling the normalised rotor thrust commands with the maximum rotor thrust, as follows:

$$T_i = T_m \cdot T_{i,\text{norm}} \tag{5.2.6}$$

where T_m is the maximum denormalised thrust of an individual rotor and i is the motor index. By combining equations (5.2.5), (4.5.6) and (5.2.6) the relationship between the normalised and denormalised virtual control surface commands is expressed as:

$$\begin{bmatrix}\delta_t \\ \delta_a \\ \delta_e \\ \delta_r\end{bmatrix} = T_m \begin{bmatrix}4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4\end{bmatrix}\begin{bmatrix}\delta_t \\ \delta_a \\ \delta_e \\ \delta_r\end{bmatrix}_{norm} \tag{5.2.7}$$

The virtual aileron command can then be denormalised and shown to be:

$$\delta_{a_R} = 2T_m \cdot \delta_{a,\text{norm}} \tag{5.2.8}$$

where δ_{a_R} is the linearised virtual aileron reference. Note that the denormalised elevator command will have the same relationship with the normalised elevator command, but the denormalised rudder deflection will use a factor of 4 instead of 2. Up until now, all the equations have been provided in the time domain, but root locus analysis will be used for the gain design. Therefore, the equations must be transformed to the s-plane by Laplace transformations. The control law then becomes:

$$\delta_{a_R}(s) = 2T_m \cdot (K_{P_p}(p_R - p) + \frac{K_{I_p}(p_R - p)}{s} + K_{D_p} \cdot s(-p)) \tag{5.2.9}$$

Stellenbosch University https://scholar.sun.ac.za

Figures 5.3 and 5.4 show the root locus plot generated using the plant transfer function with the normalised aileron command as input, and the roll rate as output, as shown in Figure 5.2. The root locus plot also shows the placement of the closed-loop poles. Both figures are of the same root locus response, where Figure 5.3 is the full view and Figure 5.4 is a magnified view closer to the origin.

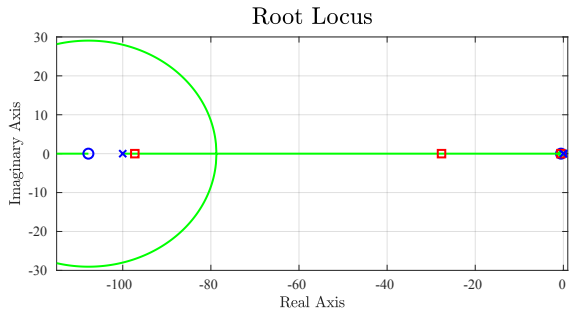

Figure 5.3: Root locus plot for the roll rate controller design

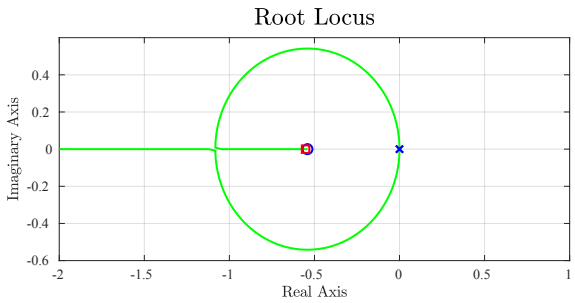

Figure 5.4: Root locus plot for the roll rate controller design (zoomed-in)

The plant has two open-loop poles, one at the origin ($s = 0$) and another at the inverse of the motor's time constant τ ($s + \frac{1}{\tau} = 0$). Consequently, the pole at the origin will be the dominant pole and the pole due to the motor delay (τ) will be the non-dominant pole of the plant at $s = -100$. However, the PID controller that was added, placed a zero close to the origin that caused the motor time constant pole to become the dominant pole and the pole at the origin the non-dominant pole.

The motor time constant could not be experimentally measured with the available equipment and it was therefore estimated based on the values used for other quadrotor vehicles in the research group. Since the other quadrotor vehicles have larger motors with larger propellers, it is assumed that the small quadrotor used in this project would have a smaller motor time constant. Because of the parameter uncertainty in the motor time constant, the controller gains for the angular rate controllers were designed conservatively.

The proportional gain size was chosen so that the bandwidth of the angular rate controller would be smaller than one third of the bandwidth of the plant. Generally, a closed-loop bandwidth less than half of the open-loop bandwidth is recommended to provide robustness to parameter uncertainty.

The integral term was then added to provide disturbance rejection for external disturbance torques such as asymmetrical thrust biases in the rotors, or external aerodynamic torques because the centre of pressure may be displaced relative to the centre of mass. The compensated system, therefore, has two free integrators in the open loop: one from the plant and one from the controller. Therefore, the control loop is type 2, and should be able to follow step and ramp angular rate references with zero steady-state error.

The integral term combined with the proportional term adds a pole at the origin and a zero close to the origin. The derivative term adds more damping to the system and combined with the proportional term places a zero just faster than the dominant plant pole.

The roll rate control system has three stable, real closed-loop poles at $s = -0.55$, -28, and -97, as shown in Figures 5.3 and 5.4. The pole at $s = -0.55$ has a zero close

Stellenbosch University https://scholar.sun.ac.za

to it, and is therefore non-dominant. The closed-loop pole at $s = -28$ is the slowest closed-loop pole that does not have a zero close to it and is the dominant pole of the closed system. Since all three closed-loop poles are real, the closed-loop response is not expected to exhibit overshoot. The zero due to the derivative term at $s = -110$ is not 4 times faster than the dominant close-loop pole as $s = -28$ and can cause overshoot.

Closed-loop step responses for the roll angle controller were obtained using the linearised, reduced-order model that was used for the control design, and also using the nonlinear, full-order PX4 software-in-the-loop (SITL) simulation. The simulated step response for both the linear model and the PX4 SITL model are shown in Figure 5.5.

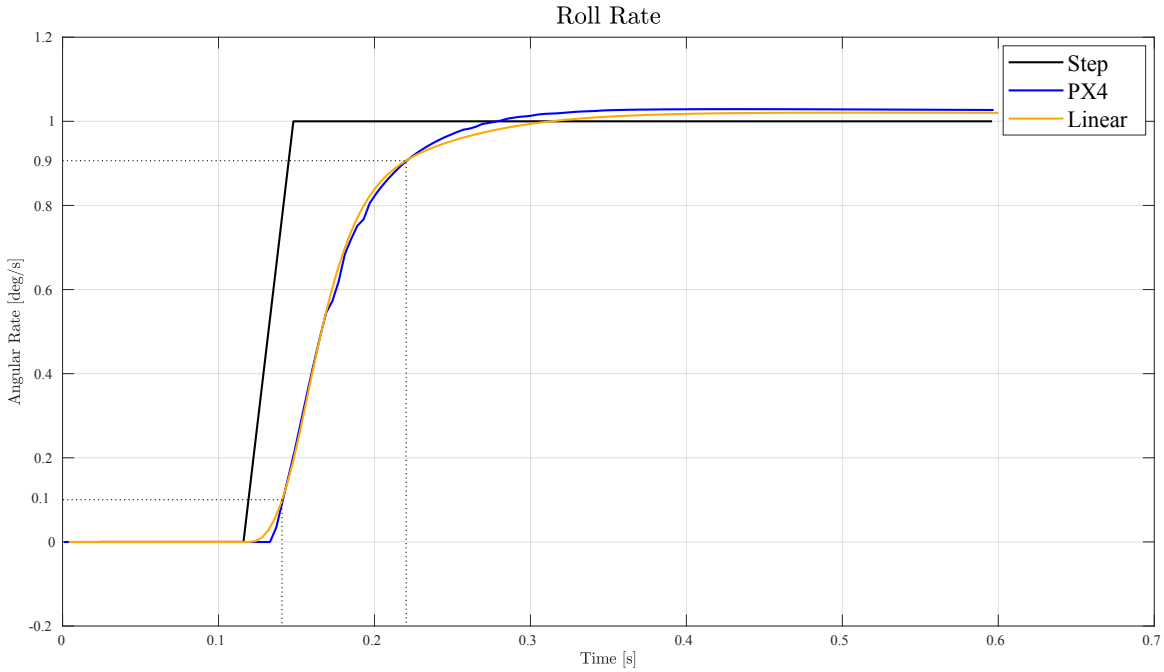

Figure 5.5: Linear vs. PX4 roll rate step response

The closed-loop step responses for both the linear model and the PX4 SITL simulation show overdamped transient responses with slight overshoot (linear model has 1.75% and PX4 has 1.98%). The response (not shown) has a long tail or decay to the commanded roll rate reference, which is generally an indicator that a dominant zero is influencing the system. The fast zero does not influence the system's stability. The rise time of both graphs are almost identical at 75 ms. The speed of the response is measured by the time constant τ_p, which is inversely proportional to the real part of the dominant pole. The expected time constant is 36 ms, since the bandwidth was 28.3 rad/s. The closed-loop Bode plots of all four controllers (roll rate, roll angle, East velocity, and East position) are shown in Figure 5.6.

Stellenbosch University https://scholar.sun.ac.za

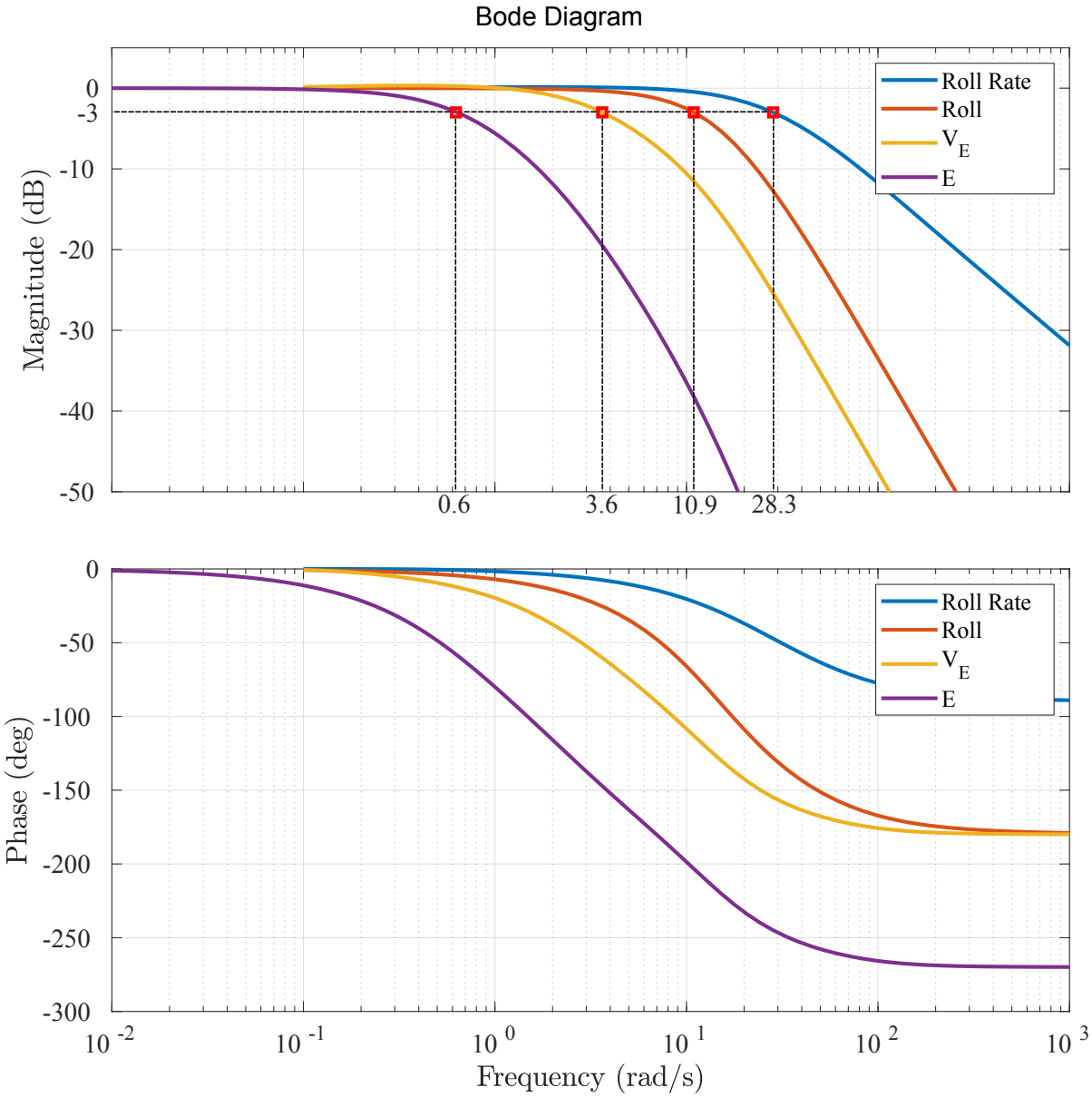

Figure 5.6: Bode plot of all four control loops

Note that the Bode plot shows all four controllers. The choices for the bandwidths of the other controllers will be discussed in the sections below.

The time constant of the PX4 SITL simulation is $44ms$ and $53ms$ for the linear model. Both of the responses are slower than the expected 36 ms from the dominant close-loop pole, but it is still within the bandwidth range that is acceptable. The SITL simulation is almost noiseless with a small amount of sensor noise and random walk added on the IMU measurements, which are required for the safety checks of the estimator. This accounts for the slight bends in the PX4 graphs. The graphs are close enough together so that the linear controller gains can be directly used without further adjustments to reach the design specifications.

Stellenbosch University https://scholar.sun.ac.za

5.2.3 Attitude Controllers

The attitude controllers actuate the references of the angular rate controllers (that were discussed in the previous section) to follow a commanded angular reference. The controller's purpose is to stop oscillations. The block diagram of the roll angle controller is shown in Figure 5.7.

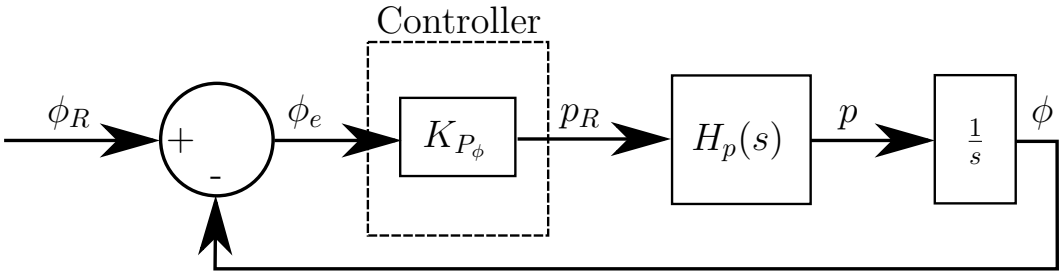

Figure 5.7: Roll Angle control loop

The roll angle error ϕ_e is calculated by subtracting the measured roll angle ϕ from the roll angle reference ϕ_R. The error is scaled with the proportional gain and sent to the roll rate controller as the roll rate reference. The roll rate controller controls the roll rate to follow the commanded roll rate, and the roll rate is integrated to result in a change in the roll angle.

The plant of this controller is the closed-loop transfer function of the angular rate controllers with a integrator in series. The next control loop that will be closed around the roll angle controller is the horizontal velocity controller which includes an integral term and can therefore provide zero steady-state tracking error for horizontal velocity references, and can provide disturbance rejection for horizontal disturbance forces. It is assumed that this is the reason why the PX4 flight control architecture uses only proportional control, rather than PI or PID control, for the roll angle controller. The output of this controller is the orientation, which is measured by the gyrometers and vision.

The design of the gains for the attitude controllers uses a combination of Bode plots and root locus designs.

5.2.4 Roll Angle Gain Design

The attitude controllers were designed to have a fast, but overdamped response so that the system would not exhibit oscillations. There also had to be a large enough time separation between the angular rate controller and the attitude controller to avoid any interference.

PX4's attitude controllers use quaternions to represent the orientation of the body axis system relative to the earth axis system. However, for small angles, the Euler angles (ϕ, θ, ψ) are approximately equal to twice their corresponding quaterion components (α_1, α_2, α_3). The three-axis attitude of the quadrotor is controlled using three separate, decoupled angle controllers: the roll angle controller, the pitch angle controller, and the yaw angle controller. controller architectures and design procedures for all three attitude controllers (roll, pitch, and yaw) are identical, and the only differences are due to the physical properties encapsulated in the angular rate controllers. Only the detailed

Stellenbosch University https://scholar.sun.ac.za

design of the roll angle controller will be presented in this Section. The architectures and design procedures for the pitch angle and yaw angle controllers are similar, and will not be presented explicitly.

The roll angle controller is designed using the linearised, reduced-order roll angle dynamics plant with the roll rate reference for the roll rate controller as the input signal and the roll angle as output signal. In the linear model, the rate of change of the roll angle $\dot{\phi}$ is approximately equal to the roll rate p. The closed-loop roll rate controller is expressed as:

$$H_p(s) = \frac{D_p(s) \cdot G_p(s)}{1 + D_p(s) \cdot G_p(s)} \tag{5.2.10}$$

where G_p is the roll rate dynamic plant and D_p is the denormalised PID controller that controls the roll rate. The plant for the roll angle controller is the closed-loop transfer function of the roll rate controller augmented with an integrator that integrates roll rate to roll angle.

The control law for the roll angle controller is formulated in terms of the quaternion attitude as follows:

$$p_R = 2K_{P_\alpha}(\text{signum}(\alpha_w) \cdot \alpha_{x,\text{error}}) \tag{5.2.11}$$

where $\text{signum}(\alpha_w)$ is the sign of the first element of the quaternion (either 1 or -1) and $\alpha_{x,\text{error}}(t)$ is the error in the quaternion component corresponding to the roll angle. When substituting the roll angle approximation into the equation above, the control law becomes:

$$p_R = 2K_{P_\alpha} \cdot \frac{\phi_{error}}{2} = K_{P_\alpha}(\phi_R - \phi) \tag{5.2.12}$$

where ϕ_R is the roll angle reference. None of the values here required scaling, because PX4 does not normalise any states involved in this controller.

The root locus of the combined roll rate and roll angle controllers are shown in Figures 5.8 and 5.9.

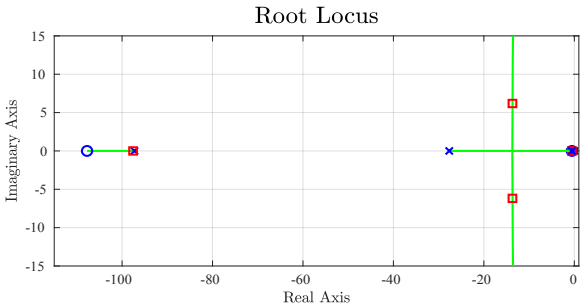

Figure 5.8: Root locus plot for the roll angle controller design

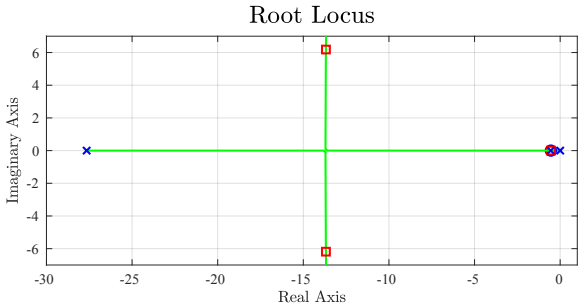

Figure 5.9: Root locus plot for the roll angle controller design (zoomed-in)

Since a proportional controller is used, no open-loop poles or zeroes are added by the controller. The open-loop pole at the origin is the most dominant pole of the system, but the zero added by the integrator cancels the pole at the origin. Therefore, the dominant open-loop pole was the pole at $s = -28.3$ and was replaced by a slower closed-loop pole due to the proportional gain. The dominant pole was replaced by a complex pole pair at $s = -13.68 \pm 6\text{j}$, which have a damping ratio of 0.91 and a natural frequency of $w_n = 14.9 rad/s$. This response will be underdamped because the damping ration is

Stellenbosch University https://scholar.sun.ac.za

below one (zeta < 1), but for a damping ratio larger than 0.9 the response does not show oscillations. Furthermore, the step response is used to show that oscillations were not present.

Since the roll angle controller only has one free integrator in the open loop, it is a type 1 system, and is expected to have zero steady-state error for step references, and a finite steady-state error for ramp references. The desired closed-loop bandwidth for the roll angle controller was chosen to be less than half of the bandwidth of the roll rate controller (which is 28.3 rad/s) to ensure adequate time scale separation between the controllers (see Figure 5.6). The desired closed-loop bandwidth for the roll angle controller was initially chosen as 14 rad/s, which is just below half the closed-loop bandwidth of 28.3 rad/s for the roll angle controller. However, this choice resulted in the roll angle controller exhibiting a very underdamped response. The proportional gain was lowered until no oscillations were present, which was when the closed-loop bandwidth of the roll angle controller equaled 10.9 rad/s. The gain that was found was implemented in the PX4 SITL simulation, and the linear response compared to the SITL response is shown in Figure 5.10.

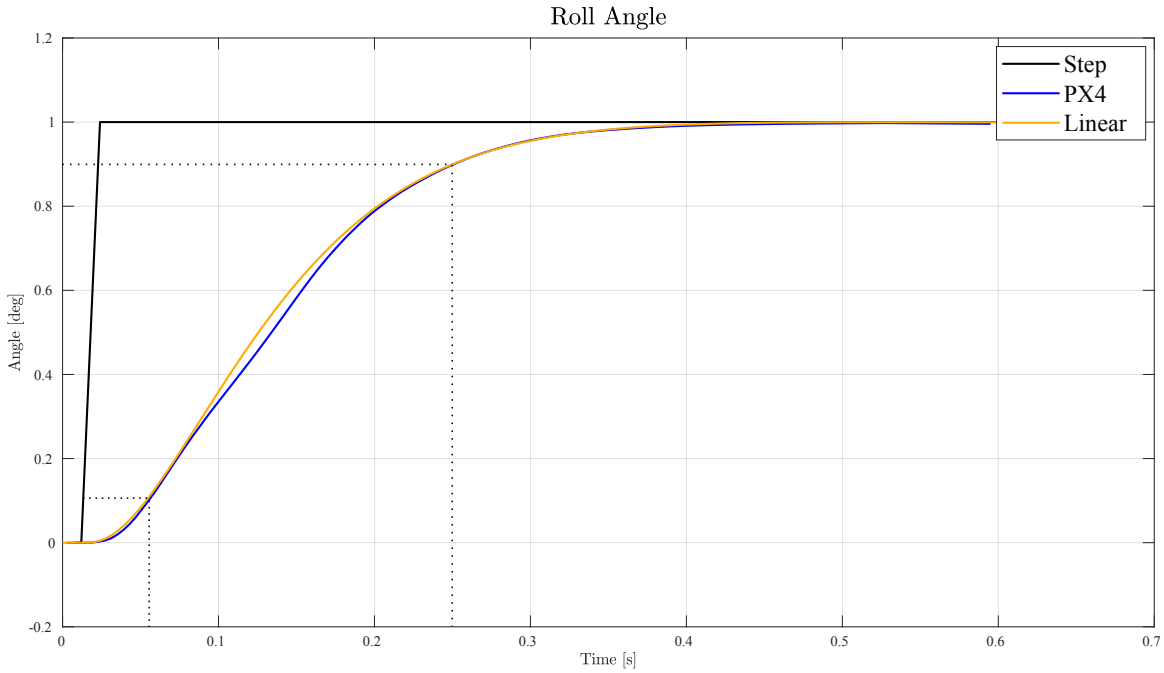

Figure 5.10: Linear vs. PX4 Roll angle step response

The rise time of both graphs are 200 ms and both have zero overshoot. The real part of the dominant closed-loop poles was placed at $s = -13.68$, which gives an expected time constant τ_ϕ of 74 ms. The time constants for the linear model was 150 ms and for the PX4 simulation was measured as 156 ms. Both of the response were just more than twice as slow as what was expected from the dominant close-loop pole. The response time was still satisfactory as it responded nearly three times slower than the roll rate controller allowing a large enough time separation between the two controllers. The PX4 controller response show disturbances, which is caused by the noise added to IMU sensors, since noise is required for the safety checks of PX4's EKF. The linear response

Stellenbosch University https://scholar.sun.ac.za

was close enough to the simulation response to considered it identical and no adjustments were required for the gains to be implemented in PX4.

5.2.5 Velocity Controller

The velocity controllers were the most complex to linearise, as a result of the acceleration to thrust to attitude conversion implemented by PX4. The controller architectures and design procedures for the horizontal velocity controllers are similar and, therefore, only the detailed design of the East velocity controller will be presented. The controller architecture for the vertical velocity controller differs from those for the horizontal velocity controllers, and will also be presented.

East Direction

The horizontal velocity controller is tasked with with following changing velocity commands and rejecting high-frequency force disturbances, such as wind. The block diagram of the controller is shown in Figure 5.11. The controller receives horizontal velocity references and actuates the attitude controller, where the command is converted from an acceleration reference to an attitude references.

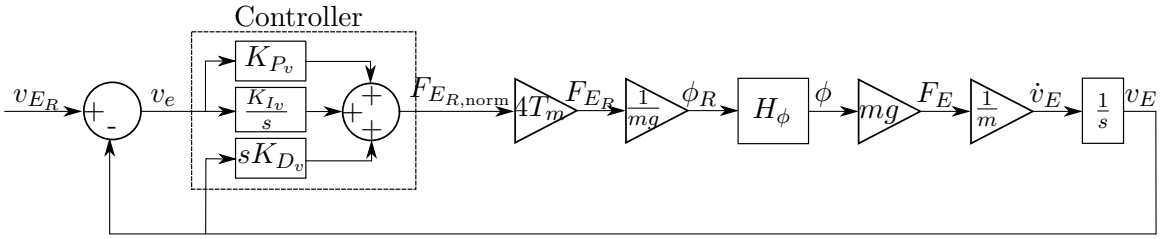

Figure 5.11: East velocity control loop

The East velocity error is calculated by subtracting the measured East velocity from the reference East velocity. The error is used in the PID controller and a normalised East force command is produced. The normalised East force command is denormalised by the scaling factor and divided by the weight to provide a roll angle reference. The roll angle reference is sent to the roll angle controller, which controls the roll angle to follow the reference. The roll angle is scaled by the weight to provide a force again and the force is divided by the mass to provide the resulting East acceleration. The East acceleration is integrated to produce a change in the East velocity.

The plant of this controller is the conversion process to change the acceleration command to an attitude command used by the attitude controller and changing the resultant attitude back to an acceleration, which is integrated to produce a velocity measurement. The controller, therefore, outputs an acceleration reference and receives a velocity measurement as feedback, which is generally measured by either a GPS or vision-based system.

A 2 DoF PID controller is used by PX4 as this controller should be able to resist velocity disturbance. The gains were designed through a combination of Bode plot and root locus designs.

Stellenbosch University https://scholar.sun.ac.za

5.2.6 East Velocity Gain Design

The East velocity controller has to be fast and stable to allow for smooth velocity responses that eases the detection process of the camera.

The plant for the horizontal velocity controllers are the most complex of the control group because it is not just the roll angle controller transfer function H_phi given in equation (5.2.13), but is the whole series of blocks from the normalised East force $F_{E_{R,\text{norm}}}$ to the East velocity v_E, as shown in Figure 5.11.:

$$H_\phi = \frac{D_\phi \cdot G_\phi}{1 + D_\phi \cdot G_\phi} \tag{5.2.13}$$

where D_ϕ is the proportional controller of the roll angle controller and G_ϕ is the roll angle controller's plant. The aspect that complicated the design of the velocity controller was how to convert the desired horizontal acceleration to an angle reference. The approach followed here was to first look at the free-body diagram of the vehicle when a small roll angle is present, as shown in Figure 5.12.

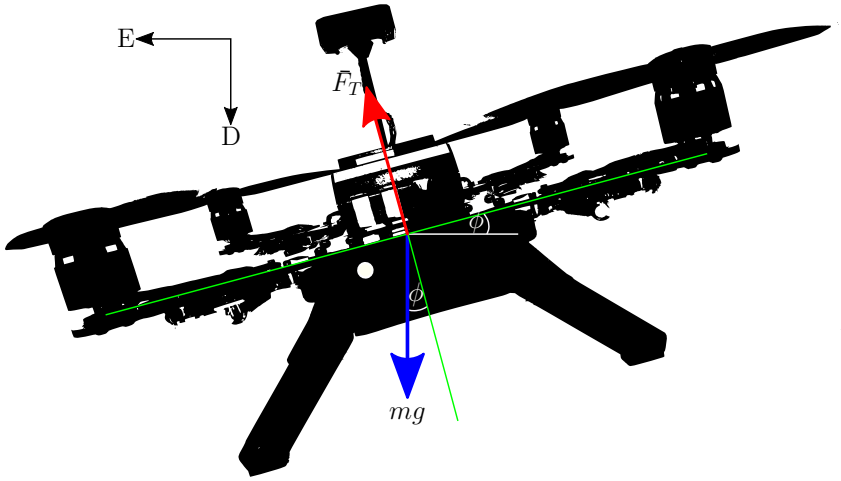

Figure 5.12: East velocity free-body diagram

F_T is the total maximum thrust produced by all four motor and propeller pairs. It is assumed that the vehicle starts at rest and that the free-body diagram depicts the vehicle that is about to move in the East direction, and is, therefore, experiencing a positive acceleration in the East direction. The vehicle will maintain a constant altitude, thus the acceleration in the Down direction will be zero. According to Newton's second law of motion, the acceleration of an object will be directly proportional to the net force acting on the object and inversely proportional to the mass. Therefore, the sum of the forces in the Down direction can be expressed as:

$$F_T \cos\phi - mg = 0 \tag{5.2.14}$$

Small angle approximation is assumed here, which simplifies the above equation to:

$$F_T \approx mg \tag{5.2.15}$$

The force in the East direction is the component of the thrust in the East direction and is expressed as:

$$F_E = F_T \sin\phi \approx mg\phi \tag{5.2.16}$$

Stellenbosch University https://scholar.sun.ac.za

where F_T from equation (5.2.15) was substituted into equation 5.2.16. Using Newton's law again, the acceleration in the East direction is related to the force in the East direction by:

$$F_E = m \cdot \dot{V_E} \tag{5.2.17}$$

where $\dot{V_E}$ is acceleration in the East direction. Two different definitions for the force in the East direction were calculated, one in terms of the roll angle and one in terms velocity.

The total thrust was normalised for PX4, consequently the controller gains are normalised and will have to be denormalised to be useful. The normalised control law for the East velocity controller is given by:

$$F_{E,\text{norm}}(t) = K_{P_v}(v_{E_R} - v_E) + K_{I_v}\int(v_{E_R} - v_E)dt + K_{D_v}\frac{d(-v_E)}{dt} \tag{5.2.18}$$

where $F_{E,\text{norm}}$ is the normalised force in the East direction, K_{P_v} is the proportional gain, K_{I_v} is the integral gain, K_{D_v} is the derivative gain and v_R is the velocity reference.

To scale this control law, it had to be multiplied by 4 times the maximum thrust produced by one motor propeller pair. This to add the weight each motor pair adds to the thrust equation. This is symbolically expressed as:

$$F_E = 4T_m \cdot F_{E,\text{norm}} \tag{5.2.19}$$

where T_m is the maximum thrust a single actuator can provide. The control law implemented in the linear model was used in the Laplace transform and is shown as:

$$F_E(s) = 4T_m[K_{P_v}(v_{E_R} - v_E) + \frac{K_{I_v}(v_{E_R} - v_E)}{s} + s \cdot K_{D_v}(-v_E)] \tag{5.2.20}$$

The root locus of the Earth velocity controller is shown in Figures 5.13 and 5.14.

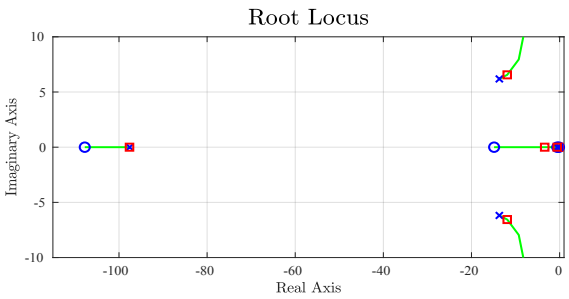

Figure 5.13: Root locus plot for the East velocity controller design

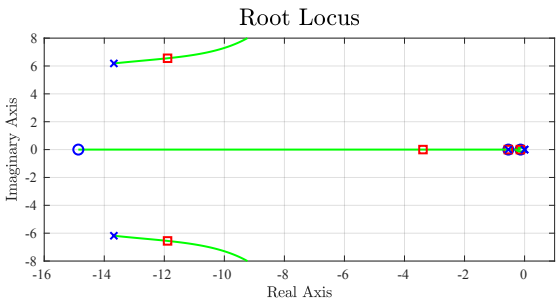

Figure 5.14: Root locus plot for the East velocity controller design (zoomed-in)

The closed-loop system of the horizontal velocity controller has two real poles near the origin: one is an integrator at the origin and the other is at $s = -0.53$. It also has two complex poles at $s = -13.68 \pm 6.1\text{j}$ and a zero at $s = -0.57$. The zero and pole near $s = -0.55$ do not influence the system because of pole-zero cancellation, which causes the two imaginary poles to be the dominant poles of the system.

The first consideration was to ensure enough time separation between the attitude controller and the velocity controller. In the Bode plot in Figure 5.6, the roll angle bandwidth was 10.9 rad/s. The velocity controller was chosen to have a third of the

Stellenbosch University https://scholar.sun.ac.za

bandwidth (3.6 rad/s), thereby ensuring a large enough time separation between the controllers. A third of the bandwidth was chosen in preference to half of the bandwidth to make the horizontal motion less aggressive to ensure that the vision-based localisation system did not lose the markers.

The integral term combined with the proportional term added an integrator at the origin and a zero near the origin at $s = -0.13$. The added integrator caused the system to again have two free integrators. Therefore, it was a type 2 system, which could follow a ramp input with a zero steady-state error and reject disturbances due to modelling errors.

The derivative term was added last for better damping and combined with the proportional term added a zero at $s = -15$. The system has three closed-loop poles that need to be placed, one real pole and one complex pole pair. The real pole is the slowest and the dominant pole. The complex pole pair are close enough to still influence the dynamic response. The result is an overdamped system that can reject disturbances, but has a slight overshoot because of the complex pole pair.

The gains were implemented in a SITL simulation with PX4 and the step response of the linear model and the PX4 SITL simulation is shown are Figure 5.15.

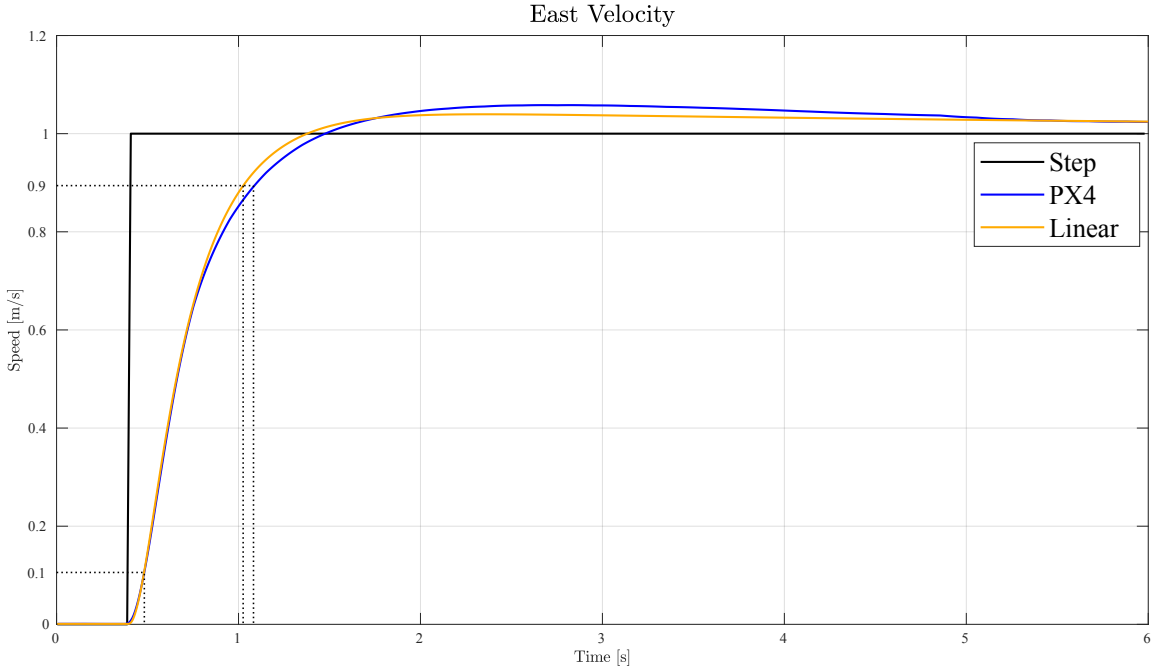

Figure 5.15: Linear vs. PX4 East velocity step response

The PX4 response has a rise time of 600 ms and overshoot of 5.8%. The linear response has a rise time of 545 ms and and overshoot of 3.8%. The rise time of the linear model is roughly 10% faster than that of PX4 and has a lower overshoot.

The most dominant closed-loop pole at $s = -3.7$ is overdamped and no overshoot would be expected, but the complex pole pair at $s = -13.68 \pm 6.1\text{j}$ is close enough to influence the response, as seen with the overshoot. The response also shows a long "tail" or "decay" of the response, which is generally associated with a dominant zero. Therefore, the zero at $s = -14.9$ is not fast enough to be ignored and does still influence the response.

Stellenbosch University https://scholar.sun.ac.za

With the dominant real pole at $s = -3.7$ a time constant of 270 ms is expected, but again both the response show much slower time responses. The linear model has a time constant of 495 ms and the PX4 simulation response has a time constant of 519 ms. The response is more than double as slow and also shows the influence that the complex pole pair has on the system. The response time is still within bound, as both respond well slower than half the roll angle controller.

The linear model is not exactly identical to the PX4 simulations, but it is within bounds to use the controller gains without further adjustments. The two responses were close, considering that the linear model made use of PID control for the velocity controller and Euler angle representation was used for the attitude controller (where PX4 uses a 2 DoF PID controller for the velocity controller and a quaternion representation for the attitude controller).

Down Direction

Unlike the horizontal velocity controllers, the Down (vertical) velocity controller does not have faster inner loops, and is itself the innermost controller for the Down direction. This controller controls the vertical velocity and, therefore, follows a slightly different design. The controller architecture for the vertical velocity controller is shown in Figure 5.16.

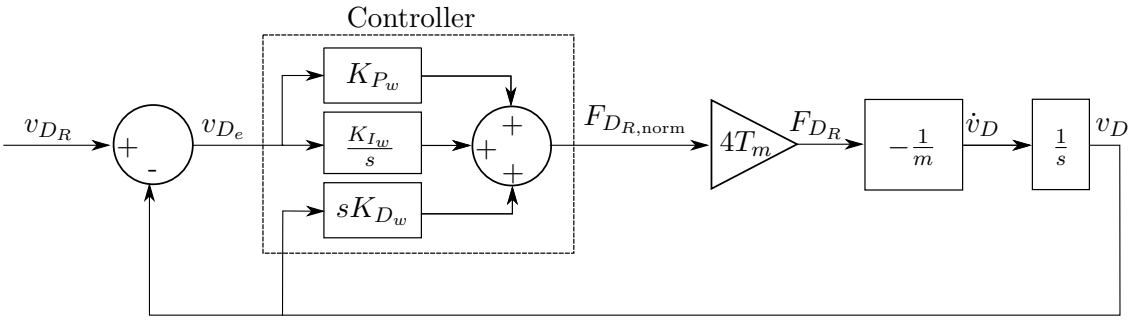

Figure 5.16: vertical velocity control loop

The vertical velocity error v_{D_e} is obtained by subtracting the measured vertical velocity v_D from the vertical velocity reference vD_R. The PID controller operates on the vertical velocity error to produce the normalised vertical force command $F_{D_{R,\text{norm}}}$. The force is denormalised with a scaling factor. The vertical acceleration is the force divided by the mass and, is integrated to obtain the vertical velocity.

The plant of this controller is the denormalisation of the controller thrust reference, converting it to acceleration and integrating the acceleration to provide a velocity.

The controller actuates the motors general thrust to produce lift and the barometer or vision-based system provides the measurements used to determine the altitude. The controller uses the 2 DoF PID controllers like the angular rate and horizontal velocity controllers. The same methods used to design the gains for the horizontal gain were used and will not be provided again. However, the plant model will be provided.

The free-body diagram for the vehicle with zero pitch and roll, is given in Figure 5.17.

Stellenbosch University https://scholar.sun.ac.za

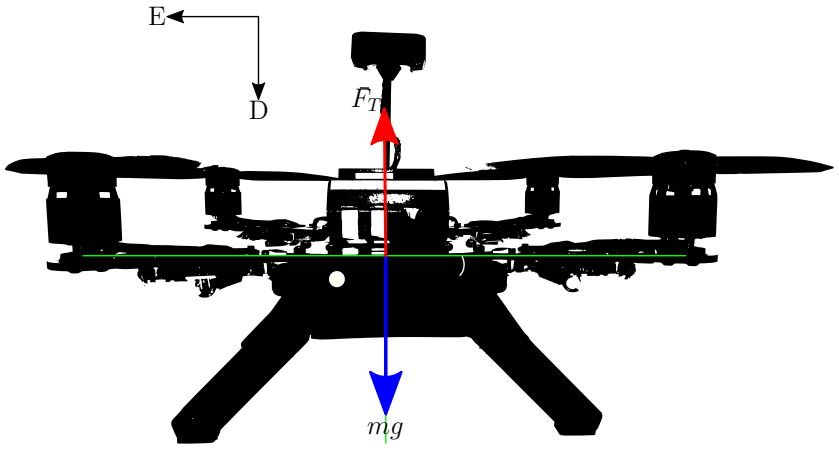

Figure 5.17: Down velocity free-body diagram

The free-body diagram shows all the forces acting on the body in the Down direction. Using Newton's second law of motion and adding the forces in the Down direction, the equation for the downward acceleration is:

$$-F_D = m\ddot{D} \approx m\dot{v}_D \tag{5.2.21}$$

The plant model for the vertical controller is simpler than the plant models for the horizontal controllers.

5.2.7 Position Controller

The position controller is the simplest of the four controllers because it receives a position reference in inertial axis and commands a linear velocity. This block diagram for the East position controller is shown in Figure 5.18. The controller enables the following of position references and ensures that the movement is stable for the cameras to not loose marker.

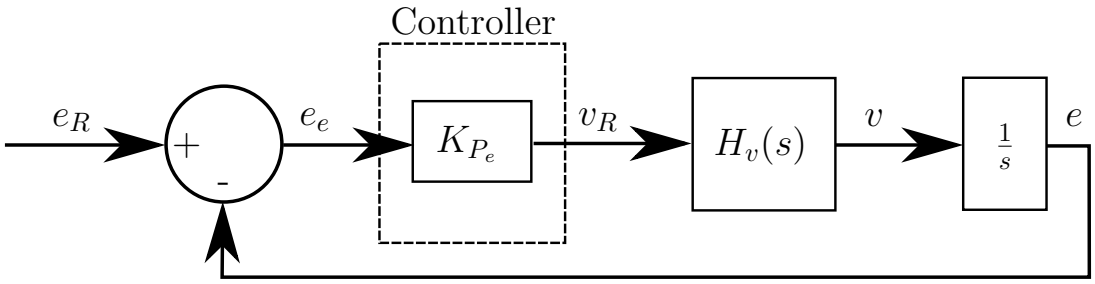

Figure 5.18: East position control loop

The error in position is calculated by subtracting the current position from the reference position. The error is then scaled by a proportional controller and supplied to the closed-loop transfer function of the velocity controller. The velocity controller produces a velocity which is integrated to determine the current position used for the feedback. The current position is measured by either the GPS or vision-based system and is given in the earth axis. The design of this controller was a combination of Bode plot and root locus design.

Stellenbosch University https://scholar.sun.ac.za

5.2.8 East Position Gain Design

The vision-based localisation system is dependent on identifying markers in each frame. The images are usually distorted if the camera moves too fast. The distortion could be so severe that no markers are detected. When this occurs, it is referred to as "motion blur". The exact speed at which motion blur would occur was unknown. Therefore, the position controllers had to be designed to not be aggressive and to have an overdamped response. The design presented below is that of the East position controller. There were no differences between the different axes.

The positional controllers are not normalised in PX4 and therefore does not require to be denormalised. The plant of this system was the closed-loop transfer function of the velocity controller combined with a integrator in line and was calculated as follows:

$$H_v(s) = \frac{D_v \cdot G_v}{s + s \cdot D_v G_v} \tag{5.2.22}$$

where D_v is the scaled PID controllers of the East velocity and G_v is the plant of the East velocity controller.

The position controllers utilised proportional control and the East direction's control law is expressed as:

$$v_{E_R} = K_{P_e}(E_R - E) \tag{5.2.23}$$

where K_{P_e} is the proportional gain, E_R is the East position reference, and E is the East position.

The Bode plot shown in Figure 5.6 shows thee closed-loop bandwidth of the East direction velocity controller to be 3.6 rad/s. To allow for the slow movement of the quadrotor to the position controller closed-loop bandwidth was chosen to be a sixth of the close-loop bandwidth of the horizontal velocity controller. The gain was chosen to provide a bandwidth of 0.6 rad/s. The East position controller root locus is shown in Figures 5.13 and 5.14.

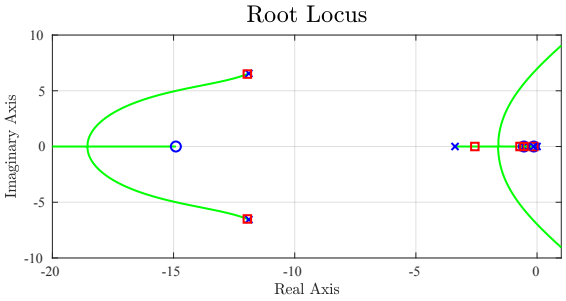

Figure 5.19: Root locus used for the East position controller design

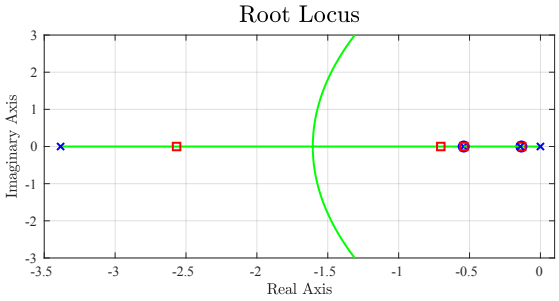

Figure 5.20: Root locus used for the East position controller design (zoomed-in)

Since a proportional controller is used, no open-loop poles or zeroes are added by the East position controller. From the placement of the close-loop poles, the dominant pole becomes the one at $s = -2.6$. The dominant pole lies on the real axis and consequently the system is overdamped. The gains were again implemented in a SITL simulation and compared to the linear model. The resultant step response are shown in Figure 5.21.

Stellenbosch University https://scholar.sun.ac.za

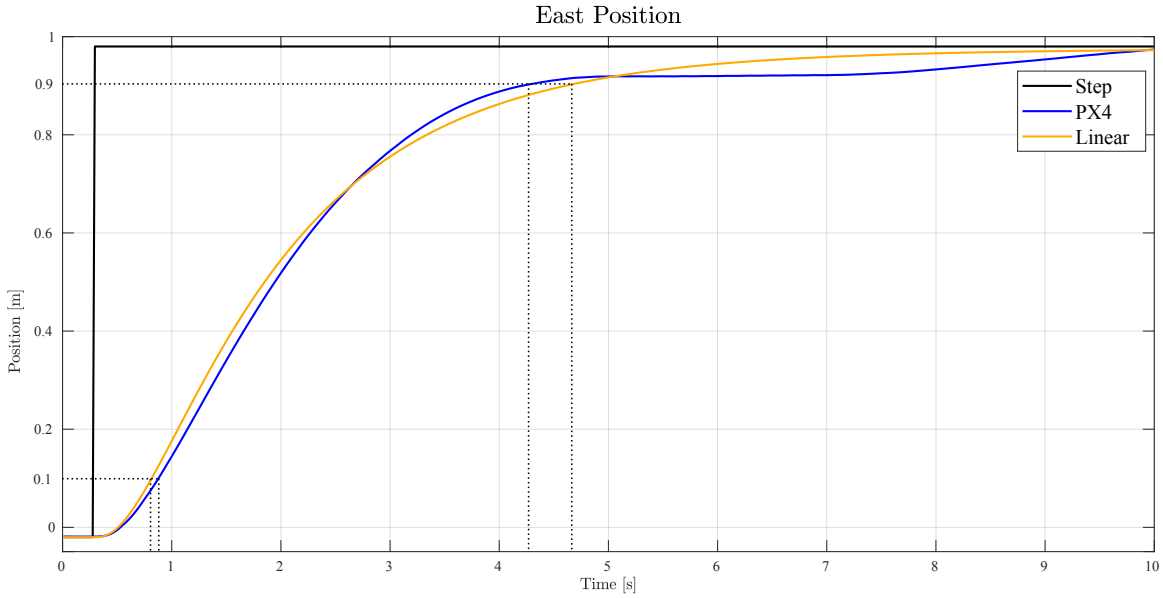

Figure 5.21: Linear vs. PX4 East position step response

Neither of the graphs show overshoot. The PX4 response has a rise time of 3.15 s and the linear model has a rise time of 3.60 s. When looking at the graphs, the PX4 system does not have the same shape as the linear model, which could be cross-coupling effects between the vertical and horizontal controllers as a full 3 axes simulation was done in PX4. A response time of 385 ms is expected from the most dominant pole at $s = -2.6$. However, the response times of the linear model and PX4 simulation was $2,380$ ms and $2,490$ ms respectively, which is much slower than what was expected. This suggest that the slower pole does influence the response and its influence is not entirely cancelled by the zero. Both the responses show a overdamped system and there is a large enough time separation, both in rise time and time constant, between the position controller and velocity controller (nearly 5 times slower) to not cause any interference. The linear model's response was close enough to the PX4 simulation response to use the gains without any alteration.

The other directions for each of the control loops were designed in exactly the same manner as described above, where different physical characteristics were used when appropriate. The exact gains used for the flight is provided in table B.1 in Appendix B, as well as a set of gains that was recommended by PX4 for the Intel® Aero RTF drone air-frame.

5.3 Practical Controller Verification

This section presents the practically measured step responses for the position controllers performed during actual flight tests. The details of the flight tests will be discussed in Chapter 8, but the practical step response results are presented here in order to compare them to the simulated step responses of the linear model, and to the expected dynamics responses based on the controller designs. Only the practical step responses of the position controllers were measured and are presented here. The linear positional controller will be compared to the actual flight data in the following Section. This

Stellenbosch University https://scholar.sun.ac.za

comparison is possible, because the waypoints that were followed caused the vehicle to step in the East direction (first a 1 m step and then a 10 m step). The first two flight tests were performed using the GPS-based state estimation, to verify that the controllers operate as expected before introducing the vision-based state estimation. The first flight test was performed using the GPS-based state estimation and the controller gains designed in this chapter; the second flight test was performed using the GPS-based state estimation and the more aggressive controller gains supplied with the PX4 flight control software.

The third and fourth flight tests were performed using the vision-based state estimation, to verify that the controllers operate as expected when using the vision-based localisation instead of GPS. The third flight test was performed using the vision-based state estimation and the controller gains designed in this chapter; the fourth flight test was performed using the vision-based state estimation and the more aggressive controller gains supplied with the PX4 flight control software. Due to the difficulty of safely allowing the vehicle to follow step commands for the faster loops, only the step responses for the position controllers were practically measured in flight tests and will be presented in this section. It should be noted that there was a slight wind from the South West present on the day of these flight tests. The first flight was that of the controllers designed in the previous section and the resultant log is shown in Figure 5.22.

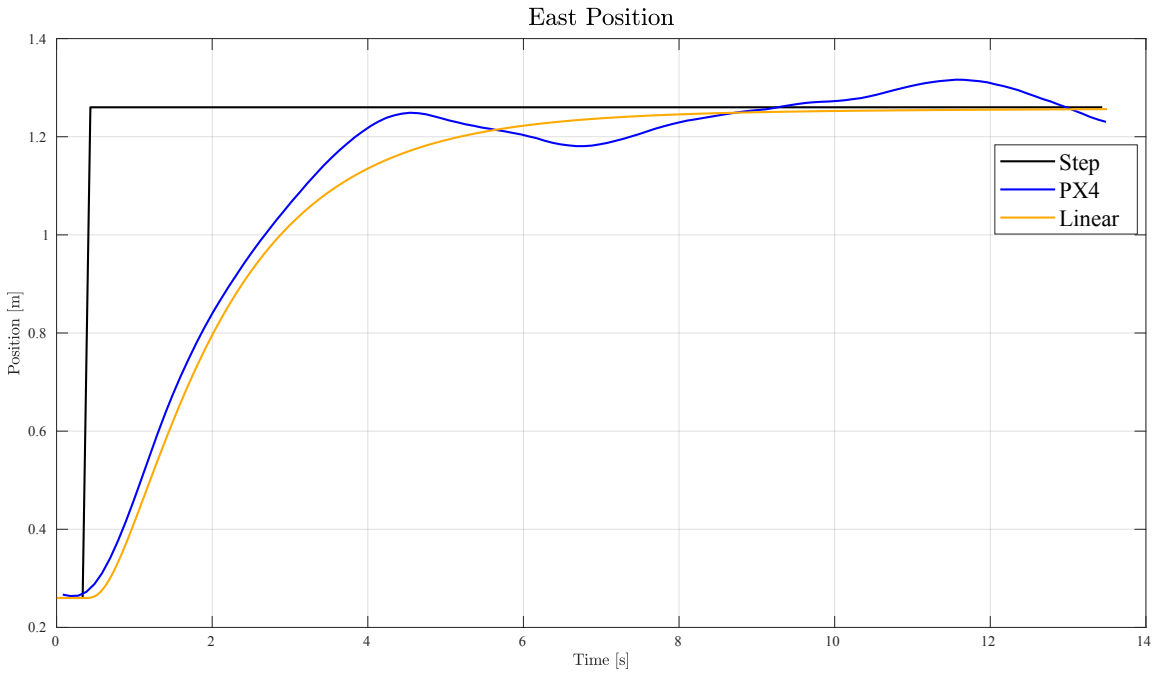

Figure 5.22: Flight test with designed controller gains and GPS-based state estimation

The graphs show that some disturbances were present as the vehicle reached a hover state. These disturbances could easily be accounted for by the presence of wind on the day of the flight test, noise of the GPS, the motor vibrations causing harmonic responses, or any combination of these. The average transient response seems to be overdamped, where the apparent oscillations are due to external disturbances. The average steady-state position follows the commanded position reference. The time constant of both the practical flight test and the linear model is just over 2 seconds with negligible difference

Stellenbosch University https://scholar.sun.ac.za

between them. Therefore, the response time of the two system are considered similar. When comparing only the rise time until the vehicle stepped 70% (at 1 m) of the step distance, the difference was within 30 ms of each other. The designed controllers were functional and had the slow movement and the low attitude angle change that was desired. PX4's more aggressive controllers were uploaded onto the vehicle and the same test was conducted again using the GPS-based state estimator. The result of the test is plotted against a linear response (with the same gains) in Figure 5.23.

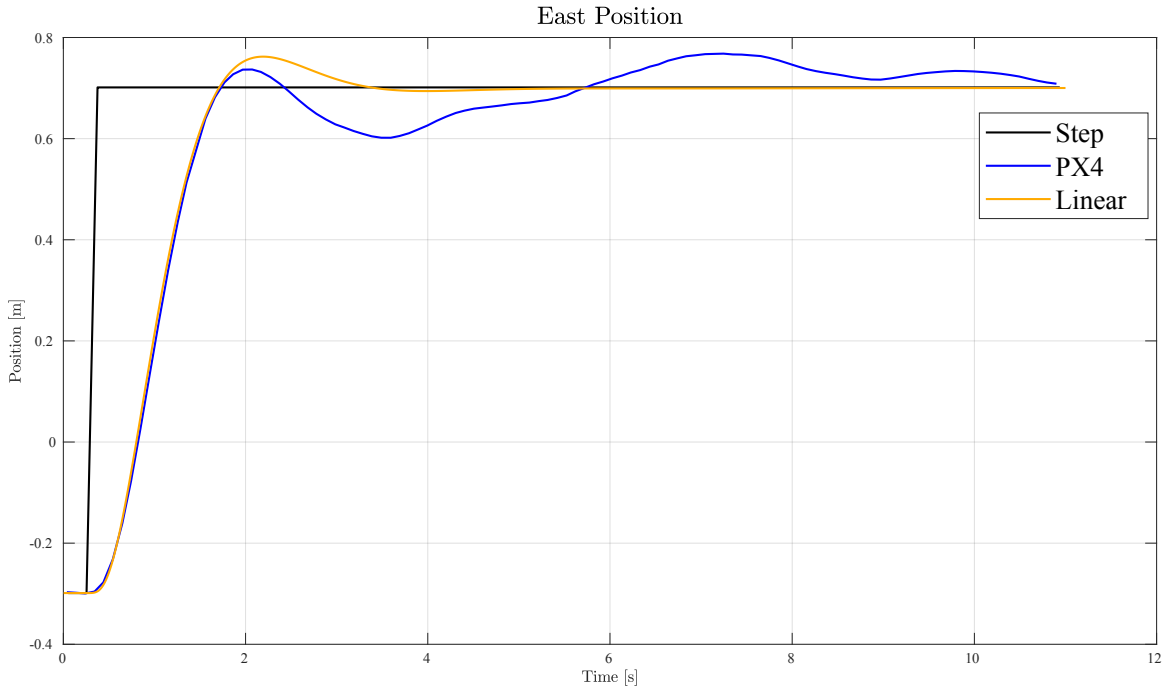

Figure 5.23: Flight test with more aggressive controller gains and GPS-based state estimation

As with the previous test, disturbances were again present, but since the GPS system, wind and the motors were all still in the system, the cause could be the result of a variety of factors. However, the shapes of the disturbances are different, which supports the notion that these disturbances were caused by environmental influences and not by a fault in the controllers. The average steady-state position again shows that the practical system follows the commander position reference, and that the oscillation motion is due to external disturbances and not due to damping. There is a very good agreement between the two time constant of the linear response and the practically measured response. The average transient response also shows an underdamped response, which was expected with the more aggressive controllers. The rise time of the linear response was 950 ms and the actual flight had a rise time of 1010 ms. The rise times were close enough to be considered identical. The response was unexpected. The expectation was that it would show the interference of the position controller on the velocity controller, since the time scale separation (according to our model) was too small. This suggests that the time constant of the motors was smaller than estimated because the smaller the time constant, the faster the response of the velocity controller and the greater the time scale separation between the two controllers. This test provided enough confidence

Stellenbosch University https://scholar.sun.ac.za

to test both sets of controllers for the vision-based system flights, because both sets of controller gains maintained a controlled flight.

The controller gains designed in this chapter were again uploaded onto the vehicle and the vision-based system was activated. There were limitations on where markers could be placed and, as a result, only a step of 1 m could be performed. The results of this test are in Figure 5.24.

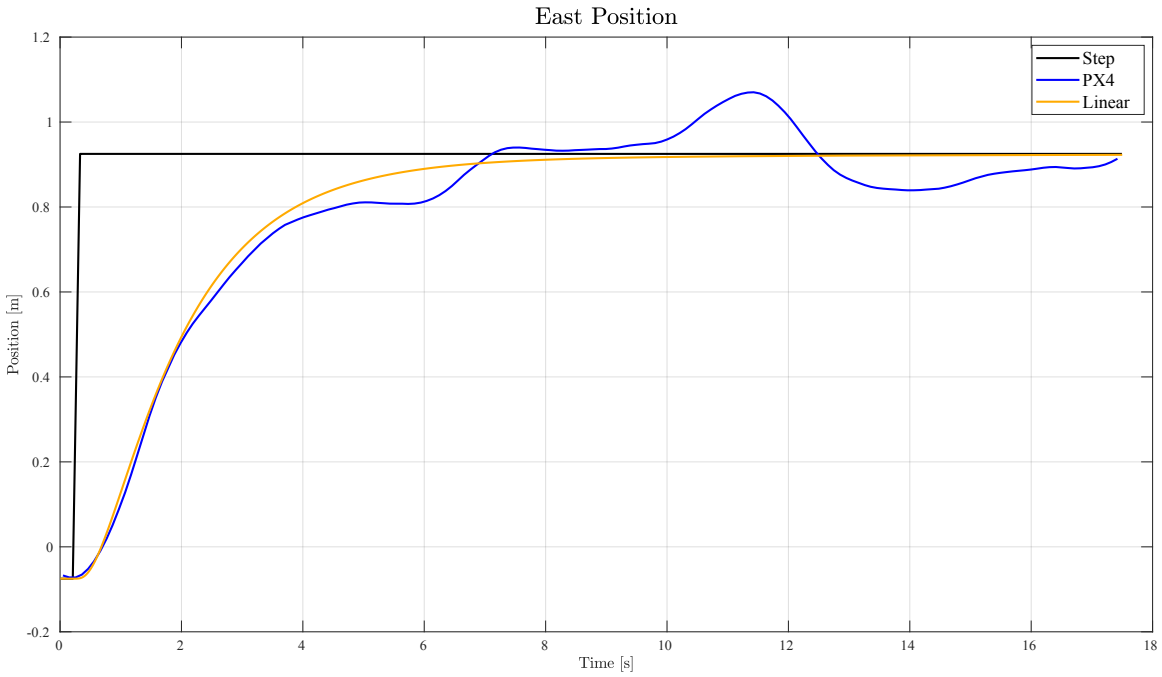

Figure 5.24: Flight test with designed controller gains and vision-based state estimation

The graphs show an interesting disturbance shape when steady-state was reached. The influence of the GPS was excluded and replaced by the vision-based system. The test was done close to the markers, where only one marker at a time was visible for the most part of the flight. As will be shown in Chapter 6, the more markers visible, the lower the noise of the localisation process. Despite the strange disturbances in the flight test, the linear model appears to show the same type of response. The average transient response show an overdamped response with no overshoot, where the oscillations are due to external disturbance. The practical measured response shows a slightly slower response time of 2.2 seconds compared to the linear response of 2.1 seconds, but the difference is small enough to be considered negligible.

The vehicle did move slow enough so that motion blur was not a problem and controlled flight was achieved. The success of the slower controller resulted in the decision that the faster controller should also be tested to determine whether motion blur would occur. The results of the more aggressive controller are in Figure 5.25.

Stellenbosch University https://scholar.sun.ac.za

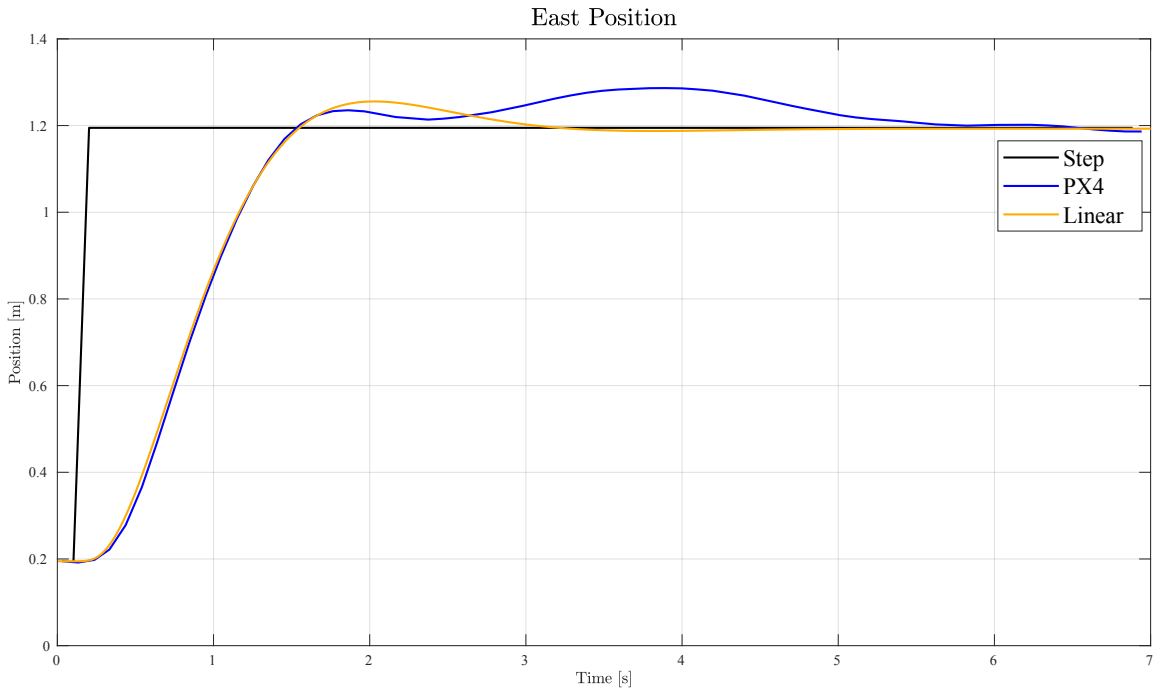

Figure 5.25: Flight test with more aggressive controller gains and vision-based state estimation

The vehicle was placed further away from the markers than in the previous test. Therefore, it had a less noisy localisation system. Yet, there are still un-modelled oscillations in the response that further verifies the external disturbances. The responses again show strong time constant agreement, with near identical rise times. The average transient steady-state is at the commanded position reference and show an underdamped response. Again the lack of interference between the position controller and velocity controller is unexpected and suggest that the time constant of the motor, that was assumed, is smaller than what is assumed here. The more aggressive controllers did not cause the camera to loose markers and controlled flight with vision was achieved.

Considering that the linear model (using a 1 DoF PID controller with no safety checks or noise) was compared to an actual flight test result (where the values were measured with sensors, filtered through an estimator and where 2 DoF PID controllers were used), these graphs show that the linear model could be used to design controller gains.

5.4 Summary and Conclusions

This chapter gave an overview of PX4's control architecture as it was used with the hardware. The design process of a similar control structure was shown, where some assumptions and simplifications were made. The process followed the design of the roll rate, roll angle, East velocity and East position control loops. Each controller was compared with a step response of the PX4 control structure using SITL simulations. The linear controller used for the design was then compared with the results of the practical flight tests. Safety concerns resulted in the testing of only the position controller with step commands and only the East direction controller was analysed.

Stellenbosch University https://scholar.sun.ac.za

The results showed that the linear model gave a fair representation of the practical system and that even though the results were not exactly similar, it was satisfactory for the design requirements. Two different sets of controller gains were tested: the gain designs discussed in this chapter and the gains provided by PX4 specifically for the Intel® Aero RTF drone air-frame. The controllers that were designed for this project were chosen to be slow to avoid motion blur on the cameras. The faster gains of PX4 were demonstrated to be slow enough to avoid the occurrence of motion blur. According to the model, there should have been control loop interference between the position and velocity controller for the faster controllers. But the flight data does not indicate that there was such interference, which leads to the assumption that the time constant of the motors was too large and that a smaller time constant could have been used. It is recommended that faster control gains are used for future work.

Stellenbosch University https://scholar.sun.ac.za

Chapter 6

Visual-Based Pose Estimation

The aim of the visual odometry system is to provide an estimate of the vehicle's pose (position and attitude) in the environment and relative to the inspection target. First, the camera model that will be used will be defined, and the distortion compensation and camera calibration will be described. Thereafter, an overview of ArUco's marker detection and pose estimation algorithm will be given. Lastly, the camera noise will be calculated.

6.1 Camera Model

A basic pinhole camera model is used to describe the projection of a point in a 3D space onto a 2D image plane [7] pp.153. This model is highly dependent on the accuracy and type of camera, but it is considered the industry standard for computer vision applications. A representation of the model can be seen in Figure 6.1, where the camera axis system (denoted with the subscript CB) is located at the origin of the camera and the image plane (denoted with the subscript I) is parallel to the $X_{CB} - Y_{CB}$ plane.

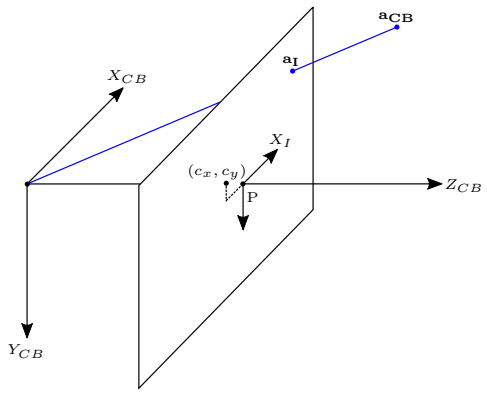

Figure 6.1: Basic pinhole camera model

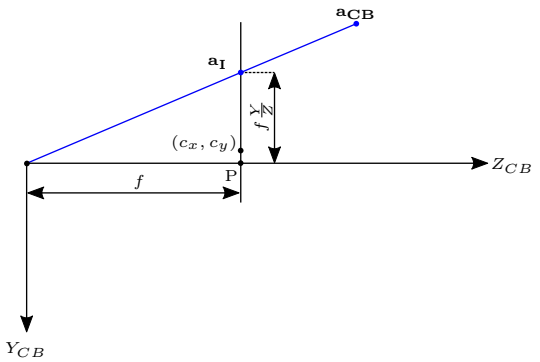

Figure 6.2: Side view of basic pinhole camera model

The point at which the Z_{CB}-axis (principle axis) and the image plane intersect should ideally be in the middle of the image plane (*camera centre*) and is called the principle point. The offset of the camera centre relative to the principle point is defined as c_x and c_y for the two axes in the image plane. The distance between the camera axis system origin and the principle point is called the "focal length" and denoted by f. When

Stellenbosch University https://scholar.sun.ac.za

considering a point in the 3D space $\mathbf{a_{CB}}$, it would be captured on the image plane at $\mathbf{a_I}$, where a straight line between $\mathbf{a_{CB}}$ and the camera axis origin intersects with the image plane as shown in Figure 6.2. Given that:

$$\mathbf{a_{CB}} = \begin{bmatrix} x_{CB} & y_{CB} & z_{CB} \end{bmatrix}^T, \tag{6.1.1}$$

and using similar triangles, the projection onto the image plane is represented by:

$$\mathbf{a_I} = \begin{bmatrix} f\frac{x_{CB}}{z_{CB}} + c_x & f\frac{y_{CB}}{z_{CB}} + c_y & f \end{bmatrix}^T \tag{6.1.2}$$

A convenient way of representing points for computer vision, is by using homogeneous vectors. A homogeneous vector describes a 2D point using three elements: $\mathbf{a_I}$ is represented as $\mathbf{A_I}$ where $a_{I,x} = \frac{A_{I,x}}{A_{I,z}}$ and $a_{I,y} = \frac{A_{I,y}}{A_{I,z}}$. The relation given in equation (6.1.2) can also be rewritten in homogeneous vector form as:

$$\mathbf{A_I} = \begin{bmatrix} X_I \\ Y_I \\ Z_I \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_I \\ y_I \\ z_I \\ 1 \end{bmatrix}, \tag{6.1.3}$$

and more concisely written as:

$$\mathbf{A_I} = \mathbf{P} \begin{bmatrix} x_I \\ y_I \\ z_I \\ 1 \end{bmatrix} \tag{6.1.4}$$

where:

$$\mathbf{P} = \mathbf{K}[I|0] \tag{6.1.5}$$

$$\therefore \mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.1.6}$$

where $\mathbf{P}$ is called the camera projection matrix and $\mathbf{K}$ is the camera calibration matrix. The rotation and translation of the camera axis system to body axis system is discussed in Section 4.1. So far, it has been assumed that square pixels are used, but that is not always the case. A scaling is required to rectify the model to include rectangular pixels and a skewing parameter is necessary to accommodate parallelograms. The camera calibration matrix is, therefore, given as:

$$\mathbf{K} = \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & s & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} fm_x & s & c_x m_x \\ 0 & fm_y & c_y m_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6.1.7}$$

where m_x is the scaling in the x-direction, m_y is the scaling factor in the y-direction and s is the skew factor. The model is considered to have 11 degrees-of-freedom, as 6 are from the translation and rotation in the three axes and 5 are from the calibration matrix above.

Stellenbosch University https://scholar.sun.ac.za

6.2 Camera Calibration

The camera model described in the previous section provides a relation between 3D points and 2D image representations when $\mathbf{P}$ is known. Several 3D points and the corresponding 2D representations are required to calculate $\mathbf{P}$ ([7] pp.178). Given a set of n points in 3D space ($\mathbf{A_{I,n}}$), and the corresponding 2D points ($\mathbf{a_{I,n}}$) in homogeneous coordinates and using equation (6.1.4), the correspondence can be shown as:

$$\mathbf{a_{I,n}} = \mathbf{P} \cdot \mathbf{A_{I,n}} = \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix} \cdot \mathbf{A_{I,n}} \tag{6.2.1}$$

where $\mathbf{P}$ is written in terms of its rows and n = 1, ..., m. Due to how homogeneous coordinates work, $\mathbf{a_{I,n}}$ does not exactly equal $\mathbf{PA_{I,n}}$, as there can be a magnitude difference. Yet, when written as the cross product $\mathbf{a_{I,n}} \times \mathbf{PA_{I,n}}$, it will equate to zero and will provide a linear solution. Equation (6.2.1) in the vector cross product notation is:

$$\begin{bmatrix} 0^T & -A_{I,n}^T & y_{I,n} \cdot A_{I,n}^T \\ A_{I,n}^T & 0^T & -x_{I,n} \cdot A_{I,n}^T \\ -y_{I,n} \cdot A_{I,n}^T & x_{I,n} \cdot A_{I,n}^T & 0^T \end{bmatrix} \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix} = 0. \tag{6.2.2}$$

However, the third row of the cross product matrix is dependent on the first two rows, where the first two are linearly independent. The solution can be simplified to:

$$\begin{bmatrix} 0^T & -A_{I,n}^T & y_{I,n} \cdot A_{I,n}^T \\ A_{I,n}^T & 0^T & -x_{I,n} \cdot A_{I,n}^T \end{bmatrix} \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix} = 0. \tag{6.2.3}$$

Consequently, every point of correspondence provides two equations and 11 equations are required to solve an 11 DoF system. Thus, 5 and a half points are required to calculate $\mathbf{P}$. The camera used for this project came pre-calibrated from Intel® and the calibration process is proprietary information. Yet, the camera matrix and distortion matrix could be streamed directly as a ROS topic to the ArUco marker detection algorithm.

6.3 Distortion

So far it was assumed that the transition between 3D space and 2D plane is linear, but any physical camera that has a lens will distort the incoming light. The distortion is generally small at the camera center and increases radially towards the periphery. The distortion is the result of the domed shape of a lens. Because the camera model and calibration process assumes undistorted images, this distortion requires correction. An example of what radial distortion does to an image can be seen below, where Figure 6.3 is the undistorted image and Figure 6.4 is the same image with radial distortion.

Stellenbosch University https://scholar.sun.ac.za

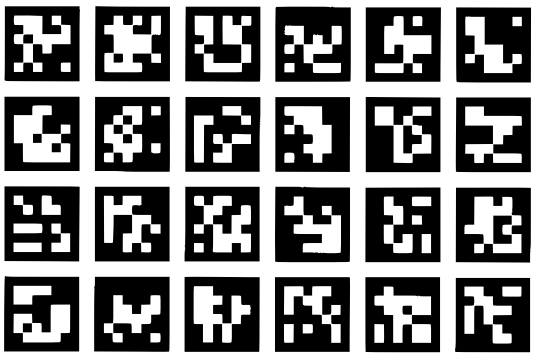

Figure 6.3: Undistorted images

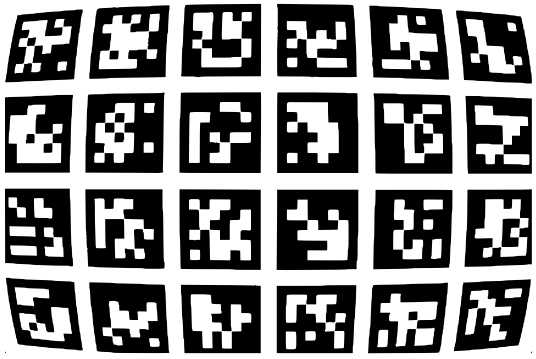

Figure 6.4: Radial distortion

The degree of distortion is proportional to the distance from the center point of the distortion $[x_{0_D}\ y_{0_D}]^T$, which is generally close to the principle point. If the distorted point is given as $[x_D\ y_D]^T$, the rectified point is $[x_I\ y_I]$ and the distance from the centre of distortion is:

$$r = \sqrt{(x_{0_D} - x_D)^2 + (y_{0_D} - y_D)^2}, \tag{6.3.1}$$

Then the distortion can be described as:

$$\begin{bmatrix} x_I \\ y_I \end{bmatrix} = \begin{bmatrix} x_{0_D} \\ y_{0_D} \end{bmatrix} + L(r) \begin{bmatrix} x_D - x_{0_D} \\ y_D - y_{0_D} \end{bmatrix} \tag{6.3.2}$$

where $L(r)$ is a distortion factor which is function of r.

As the exact function is not known, it is approximated by the Taylor expansion:

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + ... \tag{6.3.3}$$

The parameters used for radial distortion correction ($\mathbf{D} = \{x_{0_D}, y_{0_D}, \kappa_1, \kappa_2, \kappa_3, ...\}$) are considered as additional calibration intrinsics. This project used the assumption that a 3rd order representation of $L(r)$ is sufficient to describe the distortion. The OpenCV function that was used to calculate the $\mathbf{P}$ also has an optimization algorithm that solves for the $\mathbf{D}$ parameters. Since the camera matrix and distortion matrix have been calculated and a camera model has been derived, the pose estimation can now be considered.

6.4 Pose Estimation

The pose estimation of the camera is now achievable, because the distortion has been compensated for and the camera is calibrated. The method used in this thesis uses techniques that find the translation and rotation between the 2D image plane coordinates of a detected object and the reference 3D coordinates of the object. The relationship between these can be expressed as:

$$\mathbf{a_I} = \mathbf{K} \begin{bmatrix} R & T_{vec} \end{bmatrix} \mathbf{A_I} \tag{6.4.1}$$

or expanded as:

$$\begin{bmatrix} a_x \\ a_y \\ 1 \end{bmatrix} = \begin{bmatrix} fm_x & s & c_x m_x \\ 0 & fm_y & c_y m_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix} \begin{bmatrix} X_I \\ Y_I \\ Z_I \\ 1 \end{bmatrix} \tag{6.4.2}$$

Stellenbosch University https://scholar.sun.ac.za

where R is a 3x3 rotation matrix and T_{vec} is a 3x1 translation vector. This equation is an extension of equation (6.1.4), where instead of $\mathbf{P}$ equal to $\mathbf{K}[I|0]$, it instead equals $\mathbf{K}[R|T_{vec}]$. The former is true if the image plane perfectly aligned with the axis in which the environment is defined, while the latter incorporates a change in the position and orientation of the camera axis.

There are many different algorithms and solutions to solve this problem. What makes this a difficult problem to solve is to have the exact position of both the 2D image plane coordinates that correspond to the 3D coordinates. In most cases the projection on the 2D plane is measured, but the 3d coordinates are not so simple to measured. The algorithm chosen for this project was OpenCV's "solvePNP" function.

6.5 ArUco Overview

The previous sections have provided the basis for the image processing. As mentioned in Chapter 2, the ArUco library was used and, therefore, this section will provide an overview of the marker based detection and pose estimation.

6.5.1 Markers

The markers are one of the most crucial components to consider, since marker-based localisation relies solely on markers for the pose estimation. Different dictionaries of markers are provided in the ArUco libraries, where the main difference is merely in the number of bits used in the encoding. An example of a marker is shown in Figure 6.5 and the grid is shown in Figure 6.6.

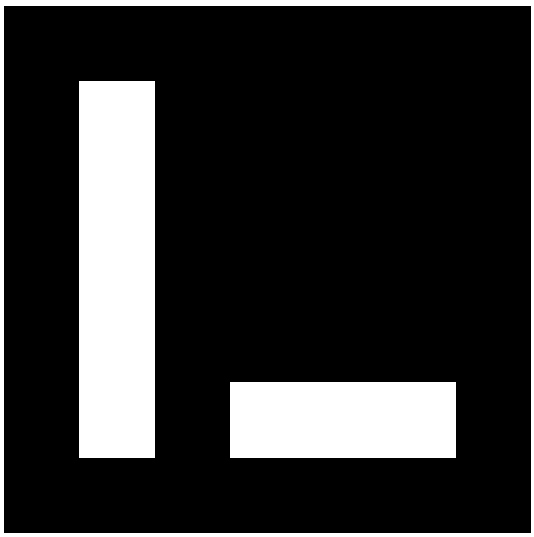

Figure 6.5: ArUco marker

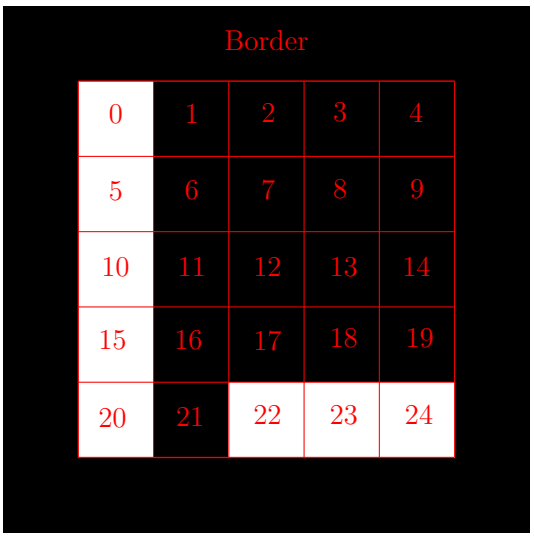

Figure 6.6: ArUco marker grid

The encoding refers to the 5x5 grid of blocks that are either black or white. This example uses a 25 bit encoding, where each marker has a unique code that is listed in a dictionary. The encoding does not directly relate to the decimal position in the list, as the created pattern does not follow the standard binary numbering, which is a result of asymmetry in the pattern. The asymmetrical pattern eliminates ambiguity

Stellenbosch University https://scholar.sun.ac.za

in the detection. Therefore, the dictionary of each marker group is essential for the identification of markers.

6.5.2 Marker Detection

These markers are placed in the environment and mapped in the camera inertial axis system. The camera takes an image of the environment. The process of marker detection is done in two main steps: the identification of marker candidates and the decoding to determine the marker ID. Using adaptive thresholding, the image is segmented and then contours are extracted. All non-convex and square shapes are discarded and the remaining candidates are sent through another filter, which provides a list of all the marker candidates in the image. Figure 6.7 shows a marker candidate after the thresholding process.

The list of candidates are then analysed further to extract the binary encoding to identify the marker. A perspective transformation is used to extract the canonical form of each candidate because a rotation will exist between the image plane and the marker. The inner grid of white and black blocks are extracted using Otsu's thresholding technique[68]. The marker dictionary defines the size of the grid (i.e. 5x5 or 25 bit) and, combined with the mean values of pixels, the encoding is extracted. Should the encoding not be defined in the dictionary, that candidate is ignored. If the encoding is defined, the four corners of the square border are saved in the order top left, top right, bottom right and bottom left; as shown in Figure 6.8.

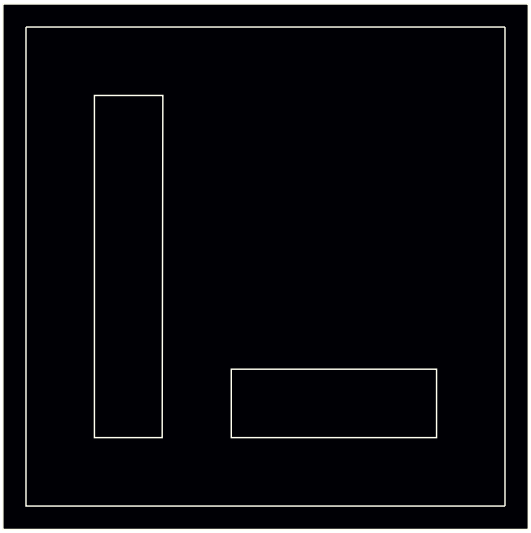

Figure 6.7: Thresholded Marker

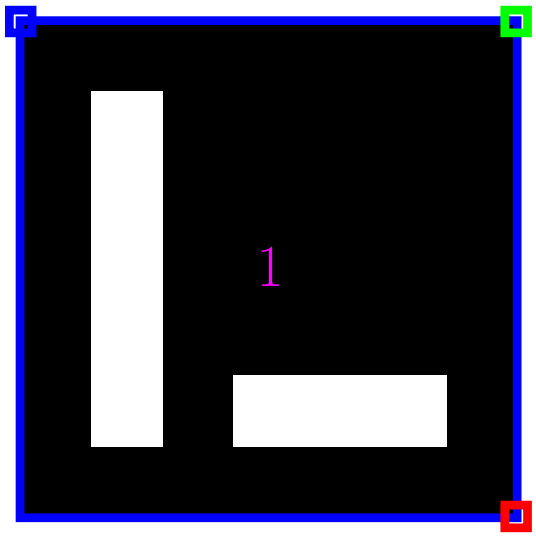

Figure 6.8: Detected Marker

Figures 6.9 and 6.10 show pictures from the on-board camera after processing. Note how the bottom marker in figure 6.10 is not detected, since the bottom edge is out of view.

Stellenbosch University https://scholar.sun.ac.za

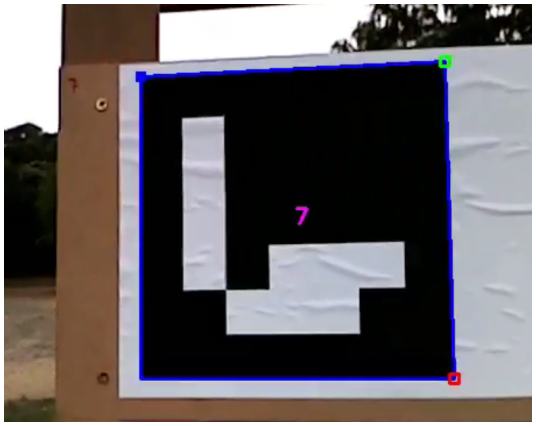

Figure 6.9: Physical Flight Marker Detected

Figure 6.10: Physical Flight Multiple Markers Detected

6.5.3 Pose Estimation

The pose estimation is done using OpenCV's solvePnP function, the algorithms of which were discussed in Section 6.4. A map in which the position of the four corners of each marker is provided (measured in the camera earth axis) is also required. The ArUco marker detection process (see Section 6.4) extracts the four corner positions (in pixels) of each marker that is projected onto the image plane. The 3D points, along with the corresponding 2D points are transformed by equation (6.4.1), resulting in 3x1 rotation (r_{vec}) and 3x1 translation vector (t_{vec}) that is measured relative to the camera earth axis origin. Because OpenCV provides the orientation in the most compact form of axis angle representation, it requires transformation to a quaternion representation. r_{vec} is the denormalised vector of an axis angle representation and the angle (β) rotated around the vector is equal to the magnitude of the vector. This can be mathematically shown as:

$$\beta = \sqrt{r_x^2 + r_y^2 + r_z^2} \tag{6.5.1}$$

with:

$$r_{vec} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \tag{6.5.2}$$

and when normalised, it equals:

$$\bar{r} = \frac{r_{vec}}{\beta}, \tag{6.5.3}$$

which is then transformed to quaternion representation by equation 4.4.8. t_{vec} was calculated after the rotation was done and, therefore, needs to be corrected before being used. r_{vec} can be transformed into a 3x3 rotation matrix (R) through the Rodrigues transformation. A 4x4 pose matrix (M) can then be created to contain the rotation and translation. It is expressed as:

$$M = \begin{bmatrix} R & t_{vec} \\ 0 & 0 \end{bmatrix}. \tag{6.5.4}$$

Stellenbosch University https://scholar.sun.ac.za

The translation ($\mathbf{P_C}$), with respect to the camera earth axis, is then equal to the inverse of M:

$$\mathbf{P_C} = \begin{bmatrix} P_c x \\ P_c y \\ P_c z \end{bmatrix} = \begin{bmatrix} M_{14}^{-1} \\ M_{24}^{-1} \\ M_{34}^{-1} \end{bmatrix}. \tag{6.5.5}$$

Both the translation and rotation vectors are then rotated to be represented relative to Gazebo's earth axis. Gazebo's earth axis is the axis system used by ROS and was described in Section 4.1.

6.6 Experimental Results

An important aspect for reliable pose estimation is knowing the measurement noise. This noise is the error in the estimation of the camera's pose relative to the world. To establish the noise characteristics of the camera and pose estimation algorithm, a series of experiments were done.

These experiments were done to establish the accuracy of the pose estimation (position accuracy and orientation accuracy) and to show the difference between the estimation error when only one marker is detected in comparison to when multiple markers are detected. The camera used was the RGB-camera that was part of the Intel® RealSense camera module.

The first tests performed were laboratory experiments to evaluate the accuracy of the position estimates and attitude estimates. The experimentally determined accuracy was used to specify the pose estimate measurement noise values for the vision-based Kalman filter.

6.6.1 Position

The experiment used to parameterise the position error was done by placing the marker(s) against a wall and marking a horizontal surface in front of the wall with the grid shown in Figure 6.11. The grid's origin (marked [0.0;0.0]) was placed 1 metre away from the marker in the negative North direction and the center of the marker was inline with the center of the camera in both the East and Down axes. The camera was placed at each of the nodes, starting in the top right corner and was then moved as shown by the blue arrows in Figure 6.11. At each node the camera was left stationary for 30 seconds and 15 seconds was then allowed for the transition between nodes.

Stellenbosch University https://scholar.sun.ac.za

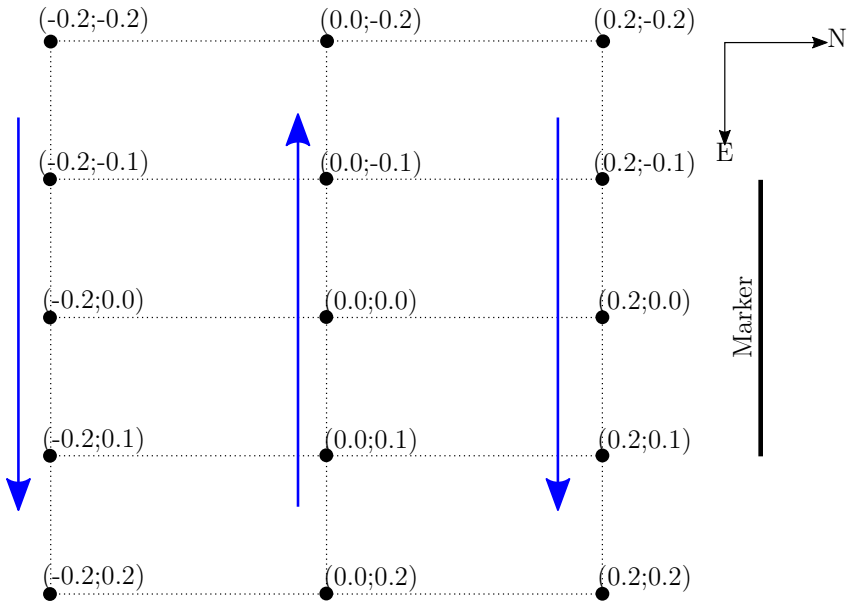

Figure 6.11: Position estimate noise experiment

The results of the position estimate from the marker detection is plotted in Figures 6.12 to 6.15. Figures 6.12 and 6.13 show the estimated positions compared to the true positions in the Earth axis system. In 6.12 only one marker was detected and in 6.13 at least two markers were detected. When multiple markers were detected the East position estimate was less noisy and, conversely, when only a single marker is detected, the East position estimate is more noisy (shown by the thicker line thickness). The vision estimates are denoted as EV, which stands for external vision. This term is used by PX4 to refer to any camera system's measurements.

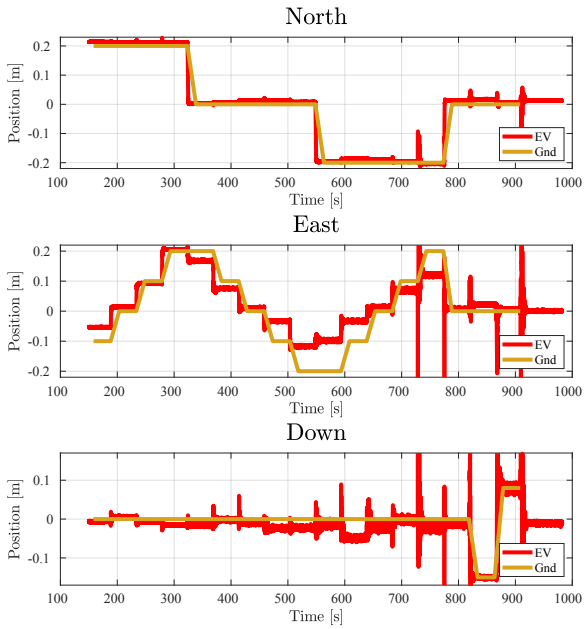

Figure 6.12: Position estimate using a single marker

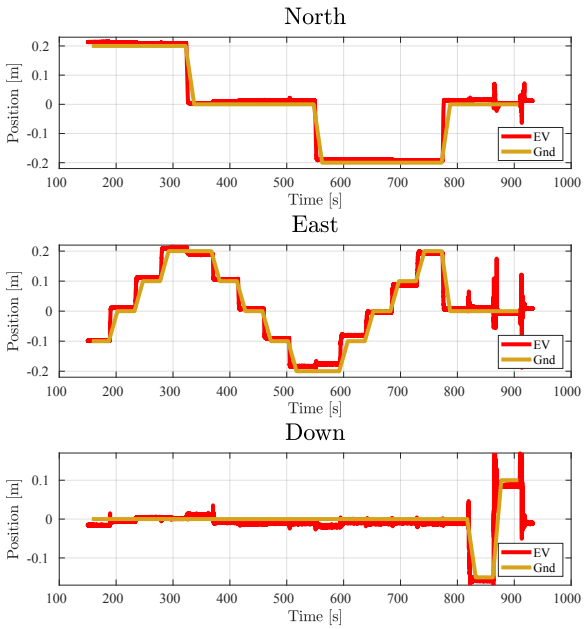

Figure 6.13: Position estimate using multiple markers

Stellenbosch University https://scholar.sun.ac.za

Figures 6.14 and 6.15 show the attitude estimate errors during the position experiments. The results show that when only a single marker is detected, then there are coupled bias errors in the East position and yaw angle, and the attitude estimates are more noisy.

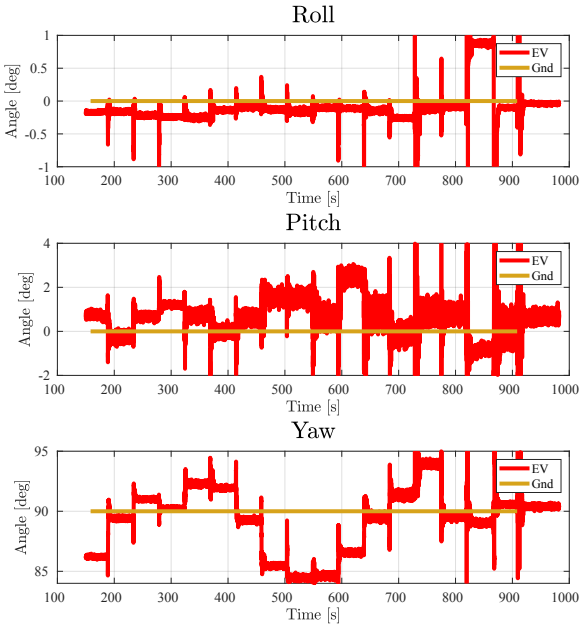

Figure 6.14: Attitude errors using a single marker (during position experiments)

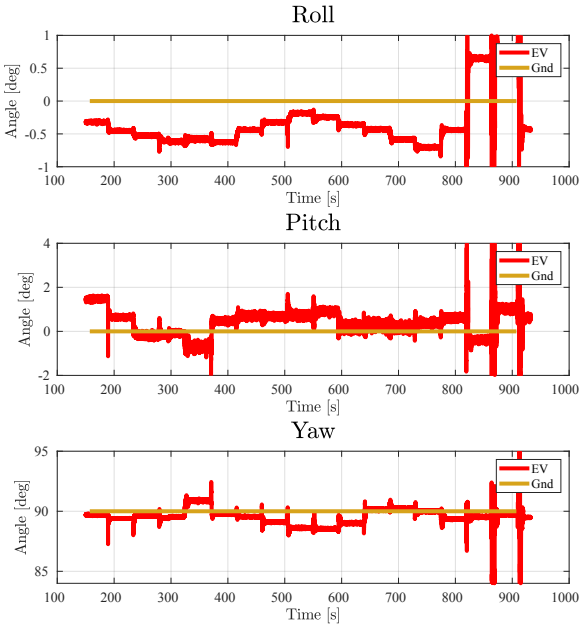

Figure 6.15: Attitude errors using a multiple markers (during position experiments)

6.6.2 Attitude

The attitude pose estimation accuracy was practically measured using a similar experimental setup as the one used for the position estimate experiments. The camera was placed in line with the center of the marker in the East and Down directions and 0.9 m in the negative North direction. The yaw angles were drawn on the flat surface on which the camera was placed with the angles marked from $-15°$ to $15°$ with $5°$ intervals. Two wooden blocks were used, one for a $10°$ and the other for a $5°$ angle, to consistently place the camera at the correct roll and pitch angles. The camera was rotated assuming a Euler 3-2-1 sequence: rotate through yaw angle first, then through pitch angle, and finally through roll angle. At each angle the camera was left stationary for 30 seconds, after which there was a 15 second transition period. The results of the attitude estimate experiments are shown in Figures 6.16 to 6.19.

Figures 6.16 and 6.17 show the estimated attitude (of the body axis system) relative to the PX4 Earth axis system. The attitude estimate is less noisy when multiple markers are detected. Yet when a single marker is detected, an accuracy of $2°$ is still achieved. However, the pitch angle estimate exhibits a large bias error when only a single marker is detected.

Stellenbosch University https://scholar.sun.ac.za

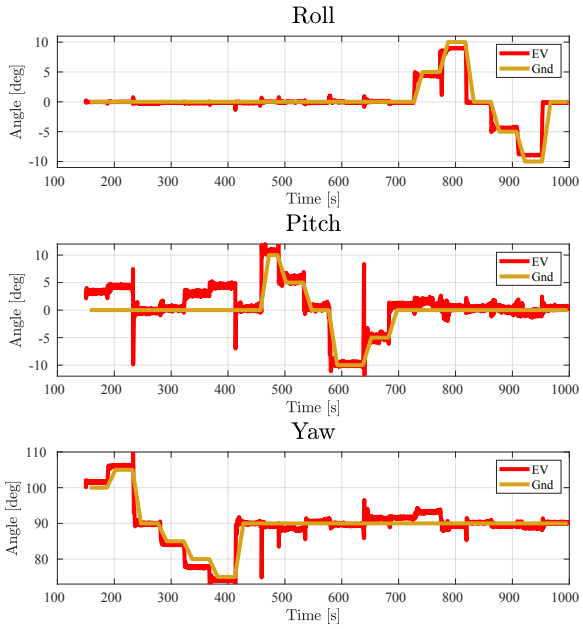

Figure 6.16: Attitude estimate when using only a single marker

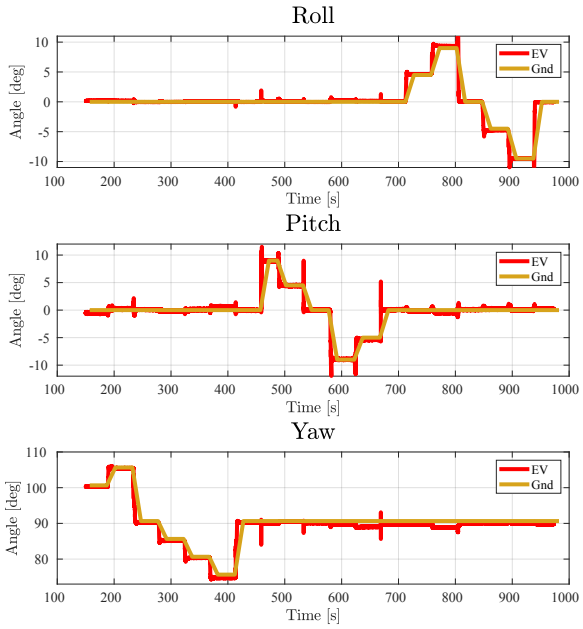

Figure 6.17: Attitude estimate when using multiple markers

Figures 6.18 and 6.19 show the position estimate errors during the attitude experiments. The position errors appear to be coupled with the orientation, since there are clear changes in position as the orientation changes. However, the way in which the experiment was done (on a flat surface with blocks lifting the camera on one side) caused the camera to move. Therefore, these figures were used to characterise the position estimate error noise values instead of treating them as bias errors.

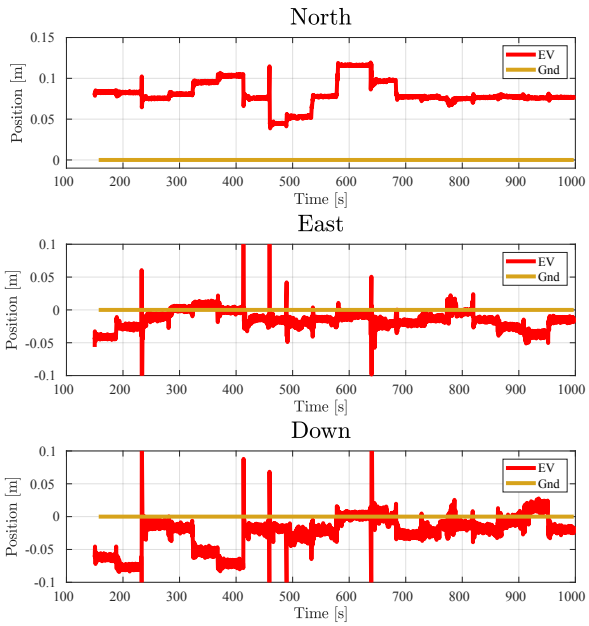

Figure 6.18: Position estimate error during attitude experiments when using only a singe marker marker

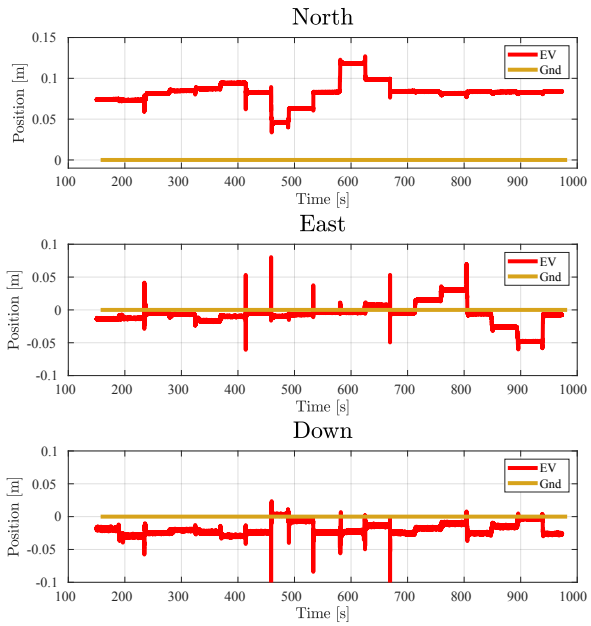

Figure 6.19: Position estimate error during attitude experiments when using multiple markers

Stellenbosch University https://scholar.sun.ac.za

6.6.3 Final Values

The position and orientation experiments tested the pose estimation system against ground truth and showed a maximum position estimate error of 100 mm (taking bias errors as noise), and a maximum attitude angle error of 5 degrees. However, in general the position and attitude estimate errors were lower than these maximum values. The position estimate error was, therefore, assumed to have a standard deviation of 50 mm, and the attitude estimate errors was assumed to have a standard deviation of 2.5 degrees. This would be one standard deviation, where three standard deviations would include 99.7% of measurements. The 3 standard deviation mark is in both cases higher than the maximum measured at 150 mm and 7.5 degree, but this compensated for additional noise due to motion blur. These values were then used as the measurements noise values for the vision-based state estimator that will be discussed in the next chapter.

As will be seen in the next chapter, the state estimator also needs to know the time delay of the pose estimate relative to the true pose. The image processing takes at most 10 ms from the point that the images are received by the ArUco library to the point that the pose estimate is published. The time stamp on the images when the pose estimation process was completed was a 100 ms after the image was captured. This suggests that the time stamping of when the original photo was taken has some delays and that more delay is added in the transmission of the image from the cameras to the software. Therefore, the time delay associated with the pose estimation used in PX4 from the VIO system was 100 ms.

6.7 Summary

This chapter described the pose estimation system that is used to determine the position and attitude of the quadrotor vehicle using computer vision and external ArUco markers. The basic pinhole camera model was shown, and the camera calibration and distortion compensation was discussed. An overview of ArUco's marker detection and pose estimation process were given and the markers used was explained. Laboratory experiments were performed to measure the accuracy of the position and attitude estimates provided by the pose estimation algorithm. The results of these experiments were also used to characterise the pose estimate noise values that will be used by the vision-based state estimator in the next chapter.

Stellenbosch University https://scholar.sun.ac.za

Chapter 7

State Estimator

This chapter describes the state estimator that is used to perform vision-based flight control. The PX4 flight control software was modified to replace the existing GPS-based state estimator with our own vision-based state estimator, and it was then integrated with the pose estimation system that was presented in the previous chapter. The first part of this chapter will give an overview of the existing PX4 delayed time horizon state estimation architecture, followed by a more detailed look at each step of the state estimation process.

The second part of the chapter will describe how the state estimator was modified to use the position and attitude measurements from the pose estimation system instead of using the GPS measurements. Finally, the results of hardware-in-the-loop (HITL) simulation tests and flight tests that were performed to verify the vision-based state estimator will be presented.

7.1 Overview of PX4 State Estimator

A block diagram showing the delayed time horizon estimator that was used, is shown in Figure 7.1. Each block of the diagram will be discussed in the sections to follow.

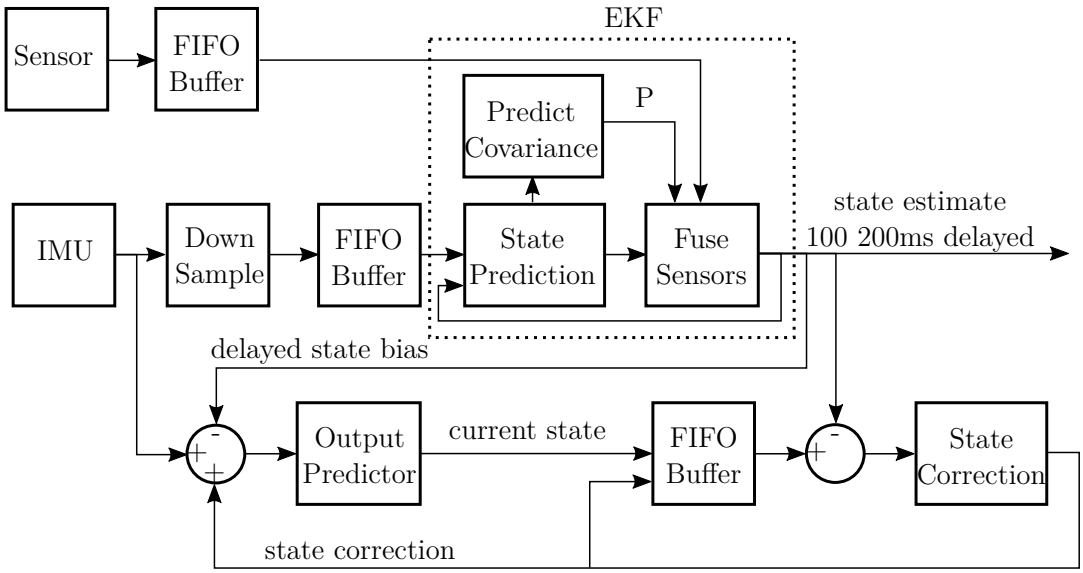

Figure 7.1: Delayed time horizon estimator

Stellenbosch University https://scholar.sun.ac.za

The estimator used by PX4 is a strapdown inertial navigation system (INS) extended Kalman filter (EKF) at a delayed time horizon, with an output state predictor to provide a current state estimate. A strapdown INS is like an EKF, where the motion model is determined by measuring the kinematic states. The kinematic states are measured by a 3-axis accelerometer and 3-axis gyroscope, which (when combined) is commonly called the IMU. The inertial measurement unit (IMU) measures the acceleration and rate of rotation which are integrated to provide the velocity, position and attitude. A strapdown INS is an inexpensive, small system in comparison to gimballed navigation [67]. The specific implementation that was used is based on the work of Bortz [61], where the changes suggested by Savage and Miller [62][63][64][65] have been implemented.

The delayed horizon state estimation architecture is based on the work done by Khosravian et al. [66], which is an estimator design that incorporates sensors of differing rates. State estimation is a difficult task because PX4 allows any combination of sensors (of varying rates). Allowing the EKF to operate at a delayed time and to propagate the delayed states forward with an output predictor to provide an estimate of the current states solves this problem. The EKF therefore uses the raw measurements at the exact time the measurements were taken and need not alter the data for time delays. The measurements are stored in a first-in-first-out (FIFO) buffer and are timestamped at the time the measurement was taken. As the EKF updates, it checks to see if its own (delayed) time correlates with the time stamp of the measurement. If the time matches, then the measurement is used. Otherwise, it is left in the buffer for the next update. The estimate will generally be delayed by $100 - 200$ ms, which are considered too long for robust flight. The current state is predicted by using the latest IMU measurement and subtracting the EKF's delayed state biases. The measurement is then corrected by a prediction algorithm to compensate for the time delay.

7.2 Extended Kalman Filter (EKF)

The extended Kalman filter estimates a state vector that contains 24 states, namely:

$$\mathbf{x} = \begin{bmatrix} \text{Quaternions}(\alpha_w, \alpha_x, \alpha_y, \alpha_z) \\ \text{Velocity}(V_n, V_e, V_d) \\ \text{Position}(N, E, D) \\ \text{Gyro delta angle bias}(X, Y, Z) \\ \text{Accelerometer bias}(X, Y, Z) \\ \text{Earth magnetic field vector}(N, E, D) \\ \text{Magnetometer bias errors}(X, Y, Z) \\ \text{Wind velocity}(N, E) \end{bmatrix} \tag{7.2.1}$$

where NED denotes coordinates in the PX4 earth axis and XYZ denotes coordinates in the PX4 body axis. The state space model for the system is represented by the following discrete-time state transition and output functions:

$$\mathbf{x_k} = g(u_k, \mathbf{x_{k-1}}) + \varepsilon_k \tag{7.2.2}$$

$$\mathbf{z_k} = h(\mathbf{x_k}) + \delta_k \tag{7.2.3}$$

where the subscript k indicates the discrete time step index, $\mathbf{x}$ is the state vector, $\mathbf{z}$ is the output vector or measurement vector, g is the nonlinear discrete-time state transition

Stellenbosch University https://scholar.sun.ac.za

function that relates the next state to the current state and the current input, h is the nonlinear output function that relates the outputs / sensor measurements to the states, ε is the process noise, and δ is the measurement noise.

The IMU measurements are coordinated in the body axis system and the position and velocity states (and measurements) are coordinated in the Earth axis system. Therefore, a rotation matrix is required to transform between body axes and Earth axes. The direction cosine matrix (DCM) (B_B^E) for the rotation from body to earth axes is:

$$\begin{bmatrix} N \\ E \\ D \end{bmatrix} = B_B^E \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{7.2.4}$$

where:

$$B_B^E = \begin{bmatrix} \alpha_w^2 + \alpha_x^2 - \alpha_y^2 - \alpha_z^2 & 2(\alpha_x \cdot \alpha_y - \alpha_w \cdot \alpha_z) & 2(\alpha_x \cdot \alpha_z + \alpha_w \cdot \alpha_y) \\ 2(\alpha_x \cdot \alpha_y + \alpha_w \cdot \alpha_z) & \alpha_w^2 - \alpha_x^2 + \alpha_y^2 - \alpha_z^2 & 2(\alpha_y \cdot \alpha_z - \alpha_w \cdot \alpha_x) \\ 2(\alpha_x \cdot \alpha_z - \alpha_w \cdot \alpha_y) & 2(\alpha_y \cdot \alpha_z + \alpha_w \cdot \alpha_x) & \alpha_w^2 - \alpha_x^2 - \alpha_y^2 + \alpha_z^2 \end{bmatrix} \tag{7.2.5}$$

and the rotation from earth to body axes is just the transpose of B_B^E, which is expressed as B_E^B.

These rotations are necessary to keep all the parameters in the same frame of reference and are necessary to express the attitude relative to the earth axis system. As all the variables are now coordinated in the same axis system, the states can be estimated (a process which occurs in four steps in the EKF). The first step is the prediction of the next states, which will be discussed in the next Sections.

7.2.1 State Prediction

The state prediction discussed here is that of position, velocity and attitude. These make up the first 10 states listed in equation (7.2.1). The first four states represent the attitude quaternion, which is propagated by integrating the gyrometers. The gyrometer provides delta angular movements around the three axes, which is symbolically expressed as:

$$\Delta_{angle} = \int_{t_k}^{t_{k+1}} \omega \; dt - \text{Bias}_{\text{gyro}} \tag{7.2.6}$$

where ω is the angular rotation measured with the gyrometers, $\text{Bias}_{\text{gyro}}$ is the estimated bias of the gyrometers, t_k is the current time step and t_{k+1} is the next time step. Δ_{angle} represents the incremental angle changes measured around the 3 axes. However, the incremental attitude change must be expressed in quaternions and the conversion is as follows:

$$\Delta_\alpha = \begin{bmatrix} \Delta\alpha_w \\ \Delta\alpha_x \\ \Delta\alpha_y \\ \Delta\alpha_z \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{\Delta_{angle,X}}{2} \\ \frac{\Delta_{angle,Y}}{2} \\ \frac{\Delta_{angle,Z}}{2} \end{bmatrix} \tag{7.2.7}$$

Equations (7.2.1) and (7.2.1) are simplifications of the mathematics involved and only holds for small angle approximations. The quaternion attitude is updated by rotating current attitude by the Δ_α, which in quaternion notation is expressed as:

$$\alpha_{k+1} = \alpha_k \otimes \Delta_\alpha \tag{7.2.8}$$

Stellenbosch University https://scholar.sun.ac.za

The velocity states are propagated by integrating the measured accelerations. The accelerometers measure an acceleration coordinated in the body axis system, which represents the rate of change of the velocity. The acceleration measurement generally has a bias offset that requires correction. This correction is expressed in the following equation:

$$\Delta_{vel} = \Delta_{IMU} - \text{Bias}_{\text{acc}} \tag{7.2.9}$$

where Δ_{IMU} is the acceleration measured by the accelerometers and Bias_{acc} is the estimated bias of the accelerometers. Since the linear velocity is required in the earth axis system and the accelerometers measure acceleration in the body axis system, the incremental velocity change Δ_{vel} must be rotated to the Earth axis system before being integrated. The accelerometer measurements include the acceleration due to gravity, for which correction is required. Therefore, the velocity is propagated using the following equation:

$$\begin{bmatrix} V_N \\ V_E \\ V_D \end{bmatrix}_{k+1} = \begin{bmatrix} V_N \\ V_E \\ V_D \end{bmatrix}_{k} + B_B^E \cdot \Delta_{vel} dt + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \cdot dt \tag{7.2.10}$$

where dt is the time interval between two measurements and g is the gravitational acceleration constant for the earth. The position is the last state to be updated and is predicted by adding the trapezoidal integration of the velocity, which is expressed by the following equation:

$$\begin{bmatrix} N \\ E \\ D \end{bmatrix}_{k+1} = \begin{bmatrix} N \\ E \\ D \end{bmatrix}_{k} + 0.5 \cdot \left(\begin{bmatrix} V_N \\ V_E \\ V_D \end{bmatrix}_{k} - \begin{bmatrix} V_N \\ V_E \\ V_D \end{bmatrix}_{k+1} \right) \cdot dt \tag{7.2.11}$$

As all the states have now been predicted, the corresponding covariance needs to be calculated. The covariance matrix describes the variance between two sequential states and the changes for each new prediction.

7.2.2 Covariance Prediction

The general form for the state transition between two time steps can be found in equation ((7.2.2)), where the Guassian noise was added to the non-linear function. PX4 uses a model where the noise is added to the measured INS values, which can be expressed as:

$$\mathbf{x}_k = g(\mathbf{x}_{k-1} + \varepsilon_k, u) + \varepsilon_{static} \tag{7.2.12}$$

where ε_k is the noise of the IMU and ε_{static} is a Guassian spread of static noise that is not related to the IMU. The covariance matrix can then be expressed as:

$$\Sigma_k = G_k \Sigma_{k-1} G_k^T + G_k \Sigma_\varepsilon G_k^T + Q_k \tag{7.2.13}$$

where Σ is the covariance matrix, G is the Jacobian of the non-linear function g that governs the transition, Σ_ε is the process noise of the IMU, and Q is the non-IMU process noise.

The covariance matrix is used in order to calculate the Kalman gain, which in turn is a measurement of how aggressively sensor readings need to be incorporated. Sensor fusion, namely the use of sensor measurements for correct prediction, is the next important point of discussion.

Stellenbosch University https://scholar.sun.ac.za

7.2.3 Sensor Fusion

Once the new states have been predicted and the covariance updated, the next step is to determine if new (non-IMU) sensor measurements are available. If there are no sensor measures available then this step is skipped. Otherwise, the innovation variance needs to be calculated. The innovation variance is:

$$S_k = H_k \Sigma_k H_k^T + R_k \tag{7.2.14}$$

where H_k is the Jacobian of the measurement function and R_k is the measurement noise. As briefly noted earlier, the covariance matrix is used along with the innovation variance to determine the Kalman gain, which is a matrix used to correct the prediction with the measurement. The Kalman gain is expressed as:

$$K_k = \Sigma_k H_k S_k^{-1} \tag{7.2.15}$$

and the states are then updated as follows:

$$\mathbf{x}'_k = \mathbf{x}_k - K_k(\mathbf{x}_k - \bar{z_k}) \tag{7.2.16}$$

where $\bar{z_k}$ is the measurement taken at the current time and $\mathbf{x}'_k$ is the newly updated states. The covariance matrix is then corrected with the latest Kalman gain. These four steps form the EKF. The EKF executes at a delayed time and these delayed estimates require must be propagated forward in time in order to be used by the flight controller. This forward propagation is called output prediction, which will be discussed next.

7.3 Output Prediction

The method used by PX4 to predict current states based on the delayed states are primarily reliant on the latest IMU measurements. The delayed state biases (from the EKF) are subtracted from the IMU measurements to determine the current states. These values require correction because of the uncertainty of the delayed time. PX4 uses a time constant (τ_{PX4}) that correlates how closely the predicted output should follow the delayed EKF estimates. The current time step's states are therefore predicted as follows:

$$\mathbf{x}_k = \Delta_{k,IMU} + \mathbf{x}_{correct} - Bias_{k-d} \tag{7.3.1}$$

where:

$$\mathbf{x}_{correct} = \mathbf{x}_{gain} \cdot \mathbf{x}_{error} + \frac{\mathbf{x}_{err\ integral} \cdot \mathbf{x}_{gain}^2}{10} \tag{7.3.2}$$

$$\mathbf{x}_{gain} = \frac{dt}{\tau_{PX4}} \tag{7.3.3}$$

$$\mathbf{x}_{error} = \mathbf{x}_k - \mathbf{x}_{k-d} \tag{7.3.4}$$

$$\mathbf{x}_{err\ integral} = \mathbf{x}_{err\ integral} + \mathbf{x}_{error} \tag{7.3.5}$$

and dt is the size of one time step of the EKF. The state $\mathbf{x}_k$ is the state used by the controllers and is logged. Note that these states would ideally be the position, velocity and attitude. But PX4 does not provide state correction for the attitude, as the necessary computational time was considered too large. Therefore, no state correction was done on the latest gyrometer measurements after the delayed state bias was subtracted.

Stellenbosch University https://scholar.sun.ac.za

7.4 Vision-Based State Estimator

The estimator architecture of PX4, as described above, allows any combination of sensors. Therefore, changing the system from GPS-based to vision-based does not require any change in the estimation architecture. The only requirements were: ensuring that the information was published to the correct topics and that the delay time was correctly specified. Regardless of the sensors that were used, an IMU was always required for a strapdown INS system.

The estimator, as provided by PX4, does not use the attitude of the vision to correct the estimated states, but to extract the heading angle, which is given in equation: (7.4.1).

$$\psi = \arctan \frac{2[\alpha_x \alpha_y + \alpha_w \alpha_z]}{\alpha_w^2 + \alpha_x^2 - \alpha_y^2 - \alpha_z^2} \tag{7.4.1}$$

The heading estimate was updated by using the attitude innovation as:

$$innov = \psi_{predicted} - \psi \tag{7.4.2}$$

This is because the attitude is not corrected for the output prediction step. The standard estimator was changed. Instead of extracting only the heading, the full attitude from external vision is used to correct the attitude estimate. The system that was implemented did not pre-calculate the heading, but rather used the full quaternion of the visual system for correcting the predicted attitude. The new equation used to describe the innovation was done element wise and expressed as:

$$innov = \alpha_{predicted} - \alpha_{vision} \tag{7.4.3}$$

This allowed for a more accurate estimation of the attitude.

7.5 Estimator Test

In Chapter 6, the visual odometry system was tested against ground truths and the errors of the system were characterised. Therefore, the visual system's pose estimate was considered the truth of the position system, where the noise was incorporated into the estimators. Consequently, the estimator was tested to verify that it could function satisfactorily. The tests were done through hardware-in-the-loop simulations and flight testing. For the HITL test, the vehicle was placed in front of a set of markers in a GPS-denied environment (inside a building) and the visual odometry system was activated. The vehicle was then manually moved around and both the pose estimated from the visual odometry system and the estimator's belief was logged. The results are shown in Figures 7.2 and 7.3.

Stellenbosch University https://scholar.sun.ac.za

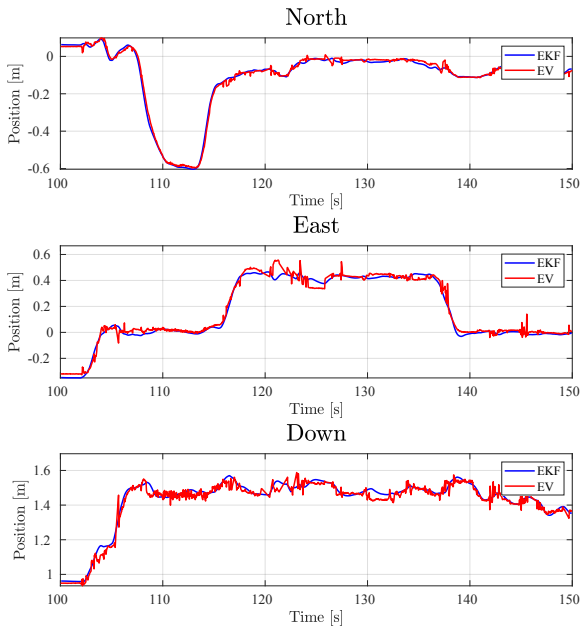

Figure 7.2: Estimated position provided by the vision-based state estimator in HITL simulation

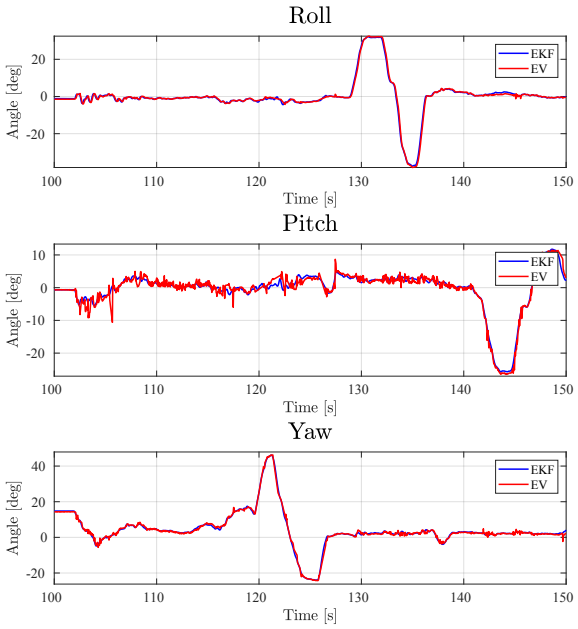

Figure 7.3: Estimated attitude provided by the vision-based state estimator in HITL simulation

The results show that the estimated position and attitude provided by the vision-based state estimator generally follows the measured position and attitude provided by the pose estimation system. However, the measured pose estimate is more noisy and exhibits "jumps" when no markers are detected in the image frame. The position and attitude estimated by the vision-based state estimator is less noisy and does not exhibit the jumps. This is because the vision-based state estimator ignores the outlier measurements supplied by the pose estimation system, and uses the inertial measurements to propagates the states when the pose estimate measurements are not available. The results shown in the two figures above provided enough evidence of satisfactory functioning of the visual estimator for an actual flight test. The flight test was conducted by placing the quadrotor UAV in front of a jig with markers attached to it (the jig will be explained in Chapter 8). The vision-based state estimator was activated and the safety pilot flew the quadrotor UAV in front of the markers. The results of the flight test are shown in Figures 7.4 and 7.5.

Stellenbosch University https://scholar.sun.ac.za

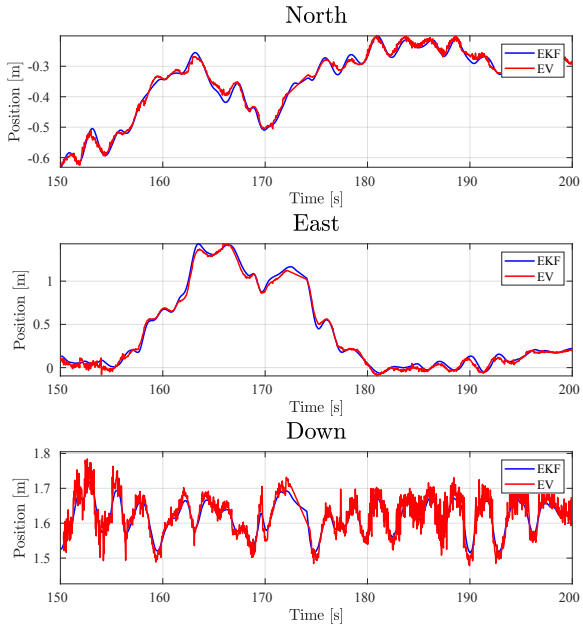

Figure 7.4: Estimated position provided by the vision-based state estimator in actual flight

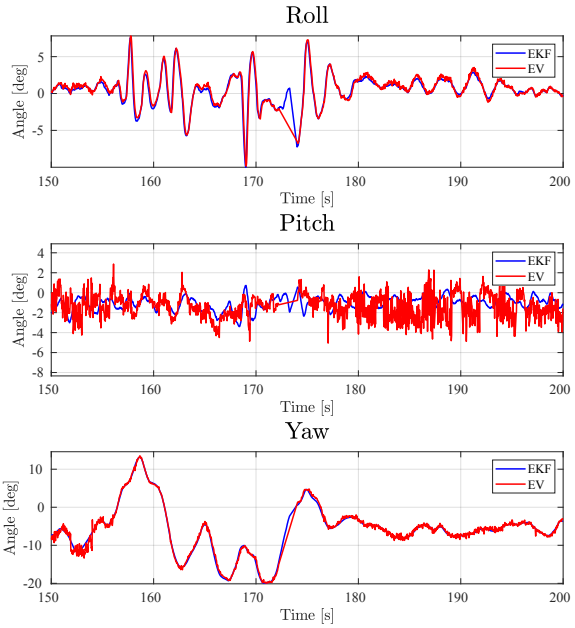

Figure 7.5: Estimated attitude provided by the vision-based state estimator in actual flight

The results show that the estimated position and attitude provided by the vision-based state estimator generally follows the measured position and attitude provided by the pose estimation system. Again the measured pose estimate is more noisy and exhibits the "jumps" when no markers are detected (like between 173s and 175s). The estimator again the results show the estimator ignoring the outliers. The estimator propagated forward based off of the inertial sensors when no markers were detected showing that the marker detection and pose estimation is very robust during flight.

The two tests showed that the estimator functioned as expected when receiving vision estimates. The estimation system was robust enough to deal with flight conditions, filtered out noise on the measurements and could propagate the states forward when no visual estimates were present.

7.6 Summary

This chapter provided an overview of the delayed time horizon EKF used by PX4. The EKF was a strapdown INS system that was executed with a delay. The sensor measurements were buffered to allow the EKF to use unaltered values. The four steps involved in estimating future states were shown, which predicted the next time step's states. Then the prediction of the corresponding covariance matrix, the fusion of the available sensors to correct the predicted states and, finally, the correction of the covariance matrix with the corrected states was discussed. The delayed state estimation was then predicted forward to the current time by an output prediction algorithm that combined the latest IMU measurements with the delayed states and corrected for the error. The estimation system was designed to work with different sensors, updating at different rates. Therefore, the visual system's estimates were easily added. The EKF did not use the full

Stellenbosch University https://scholar.sun.ac.za

quaternion to update the attitude and this was changed. Finally, the estimator was verified by an HITL simulations and a flight test. In both the HITL simulation and flight test the the estimator followed the measured position and the marker detection and pose estimation was shown to be very robust during flight. The estimator ignored outliers and propagated the states forward using only inertial measurements.

Stellenbosch University https://scholar.sun.ac.za

Chapter 8

Results

The Chapters have thus far explained the design and testing of different systems. Each system was tested individually in software. Some of the systems (like the controllers and the visual odometry system) were tested with HITL simulations, but this only proved that it should work in theory. This chapter will show how these different systems were integrated and tested in flight tests.

As discussed in Chapter 3, the HITL simulations were done in two steps. The same procedure was followed for the flight tests. The first test aimed to verify that the control system and the estimator worked as expected. The different control loops were tested with GPS providing localisation information and with the vision-based system switched off in the first test. The test was conducted by switching between different modes, where each mode gave the safety pilot control of the set-points of the loop that was being tested, i.e. in "acro-mode" the transmitter commands were directly given as angular rate references. Once the controllers and estimator were verified to function as expected, the waypoint scheduler was tested. The vehicle was flown to an altitude of $10m$ by the safety pilot and switched over to the waypoint scheduler, which then executed a series of points for the vehicle to follow.

Satisfied that the waypoint scheduler was working, the vehicle was switched from GPS-based localisation to vision-based localisation for the second set of tests. The vehicle was placed in front of a jig with markers attached to it. The safety pilot controlled the vehicle for the first flight and tested that the system functioned correctly. The success of this test provided enough confidence to proceed to switch the vision-based system on to follow waypoints. The vehicle followed the waypoints successfully and the safety pilot landed the vehicle. Each of the flight tests, with the corresponding log files, will be discussed in more detail below.

8.1 Control Loop and Estimator Test

During the test, the safety pilot controlled the vehicle with a wireless transmitter. The control system was tested, starting with the fastest loops and only proceeding with the slower loop once the faster loop was verified as functional. The angular rate loop's mode was called "acro" and in this mode the safety pilot flew at a low altitude to minimise damage, should something go wrong. The flight was successful. Since a flight mode was now tested and found to be reliable, the vehicle was flown to roughly 5 m and switched over to the attitude control loop that uses the "stability" mode. The safety pilot flew the vehicle around and the control loop functioned as expected. The vehicle was

Stellenbosch University https://scholar.sun.ac.za

then switched over to the velocity controller (that is somewhat mislabelled as "position mode"). This mode used GPS for the vehicle's velocity and again, the controller was verified as functional. The vehicle was landed and the waypoints were uploaded into it. The position controller could only be tested by the waypoint scheduler, as the transmitter had no mode that allowed for position references. The vehicle was again controlled by the safety pilot up to a height of 10 m and the waypoint scheduler was switched on. Due to an error with the logger, only the waypoint scheduler's logs were saved and the results are shown in Figure 8.1.

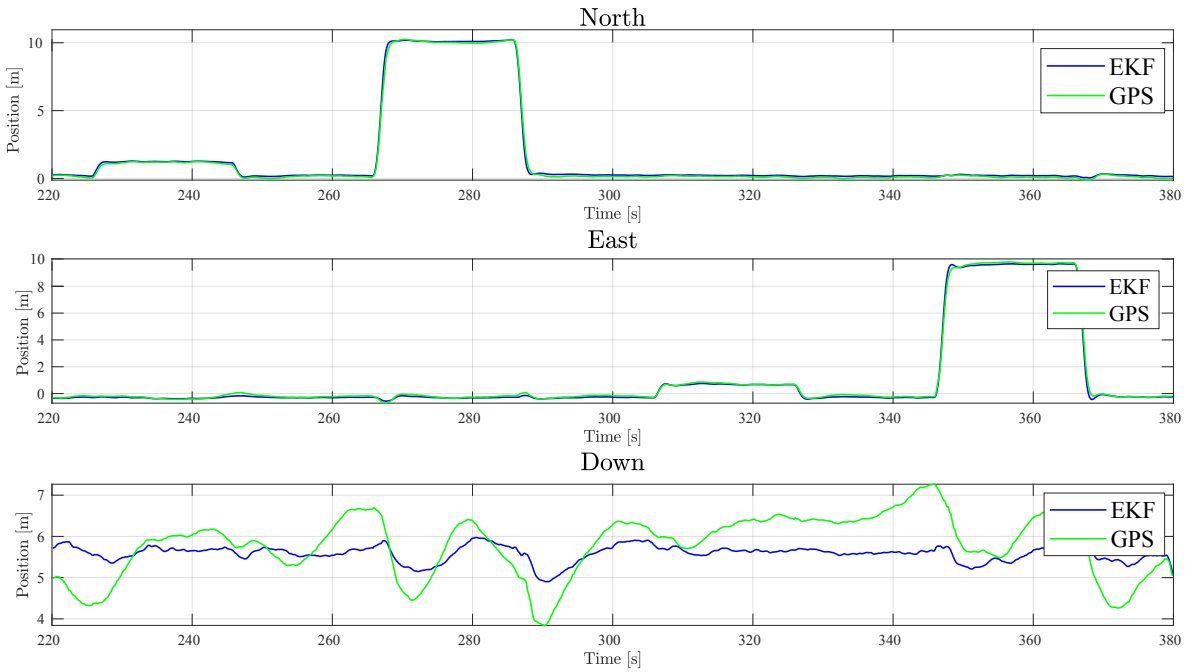

Figure 8.1: Estimated position provided by the GPS-based state estimator with more aggressive gains

The results shown above relate to the controller gains of PX4 and not the gains that were designed in Chapter 5. The controller verification was also discussed in Chapter 5 and this is only to show that the waypoint scheduler and estimator functioned as expected. It is noteworthy that for the Down direction, the GPS does not match the estimated altitude. This was because the estimator used the barometer as its primary source of altitude and used the GPS for latitude and longitudinal positioning. The waypoints that were commanded were the following: a 1 m step North, wait 20 s, then step back to the origin, wait 20 s, step 10 s North, wait 20 s, step back to the origin. The same steps were repeated the in the East direction. On the day of the flight test, a slight wind form the South West was present. Small changes in the East direction were observed when North steps were performed and the slightly less but still visible changes in the North direction when East steps were performed. This was the result of the method used by PX4 to prioritise the limited motor thrust commands. There was not enough thrust authority left to control the lateral directions when big steps in the longitudinal direction were commanded, and vice versa. It did correct fairly quickly for the slight deviations and the result was a stable flight following the waypoints.

Stellenbosch University https://scholar.sun.ac.za

8.2 Vision-based

Following the success of these first tests, the vision-based localisation system test could commence. The first flight was done with the safety pilot controlling the vehicle again. The vehicle was placed in front of an upside down 'L'-shaped structure that contained 14 markers, as seen in Figures 8.2 and 8.3.

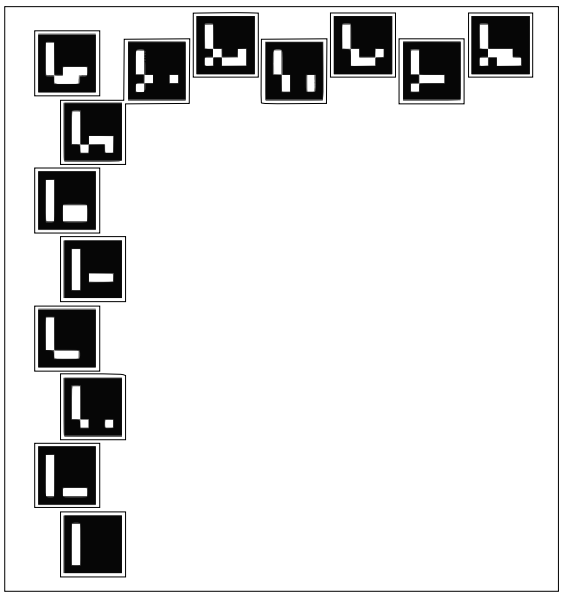

Figure 8.2: Marker Jig Layout

Figure 8.3: Actual Marker Jig

The markers were intentionally placed in a zig-zag pattern to lower the inaccuracies of co-planar markers. The vehicle was flown in line with the top row of markers and switched over to position mode. This mode uses the pose estimate of the vision system to localise the vehicle. It hovered stationary, showing that the system was working. The safety pilot then flew the vehicle in this mode at the same altitude. This ensured that the vehicle could fly with visual localisation. Figures 8.4 and 8.5 show the results of this flight.

Stellenbosch University https://scholar.sun.ac.za

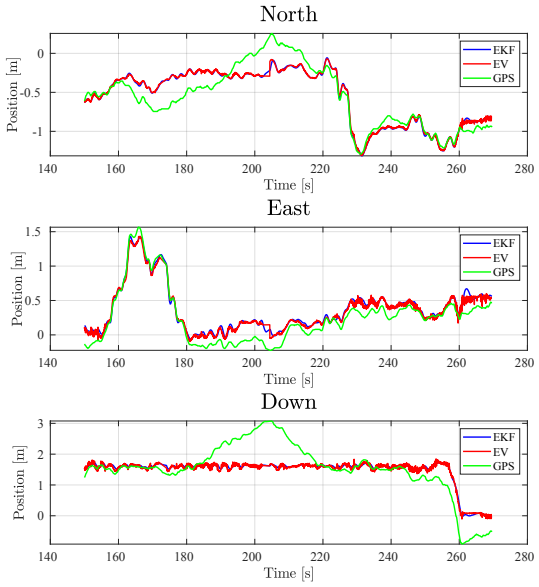

Figure 8.4: Estimated position provided by the vision-based state estimator in manual flight

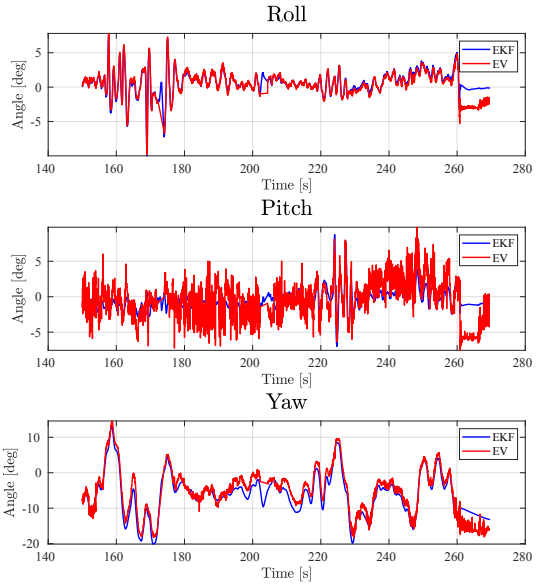

Figure 8.5: Estimated attitude provided by the vision-based state estimator in manual flight

The flight proved that the time delay that was set on the pose estimates from the vision, as well as the noise parameters, was correct. The flight was noisy, but this could be due to the influence of a slight wind on the day of the flight test. Despite the influence that the wind had on the aircraft, the result of this flight test was a stable flight.

The next flight test was performed with less aggressive controllers. The safety pilot controlled the vehicle and let it hover in front of the marker in the top, left corner of the jig and then switched the vehicle over to "off-board mode" for autonomous control. The navigation was a waypoint scheduler that sent position and yaw commands to the controllers. The waypoints that were used, were measured relative to the position that the autonomous mode was switched on. The position and heading angle of the vehicle (when the "off-board mode" was switched on) is considered the origin (0, 0, 0, 0) of the waypoint scheduler. The position waypoints were given in metres and the heading angle in degrees. The waypoints were:

1. (0, 0, 0, 0)
2. (0, 1.2, 0, 0)
3. (0, 0, 0, 0)
4. (-0.5, 0, 0, 0)
5. (0, 0, 0, 0)
6. (0, 0, 0, 15)
7. (0, 0, 0, 0)

The waypoints are provided as North, East and Down in metres, where the heading angle is given in degrees (as mentioned above). Each waypoint was held for 20 seconds

Stellenbosch University https://scholar.sun.ac.za

before proceeding to the next point. The results from this flight is provided in Figures 8.6 and 8.7.

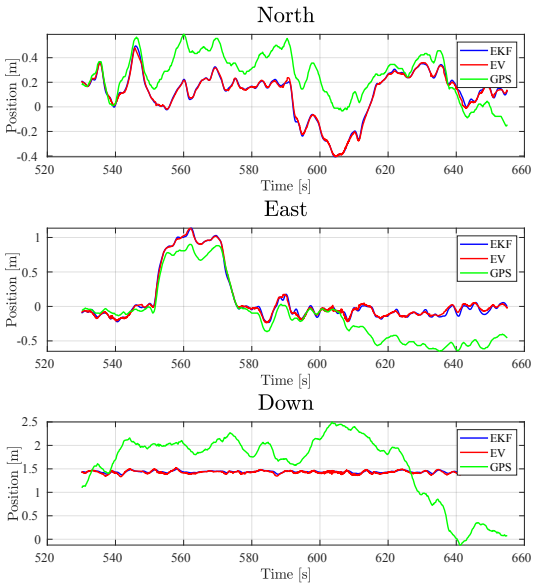

Figure 8.6: Estimated position provided by the vision-based state estimator with designed gains

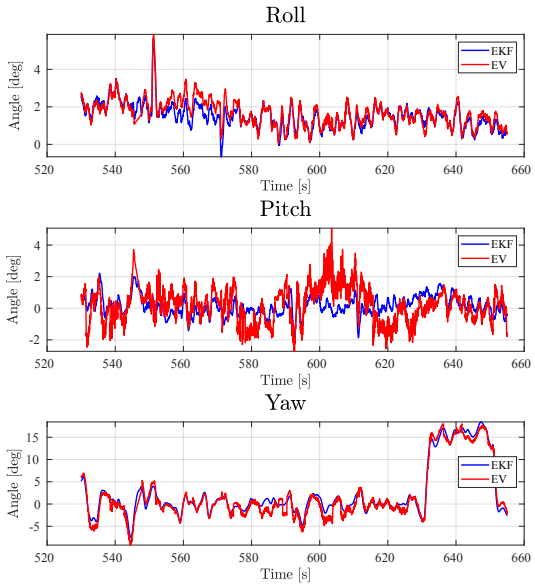

Figure 8.7: Estimated attitude provided by the vision-based state estimator with designed gains

Again, the GPS drift was clearly visible. The flight was done with the vehicle 800 mm away from the markers. After reviewing the on-board video, it was clear that most of the time only one marker was visible in the image frame at a time. However, the flight showed that the autonomous navigation with vision-based localisation was successfully implemented.

The last test flight was done exactly like the previous one, with the safety pilot controlling the vehicle in place and then switching it over. The same waypoints were used. The only difference was that the more aggressive controllers were used. The flight results are shown in Figures 8.8 and 8.9.

Stellenbosch University https://scholar.sun.ac.za

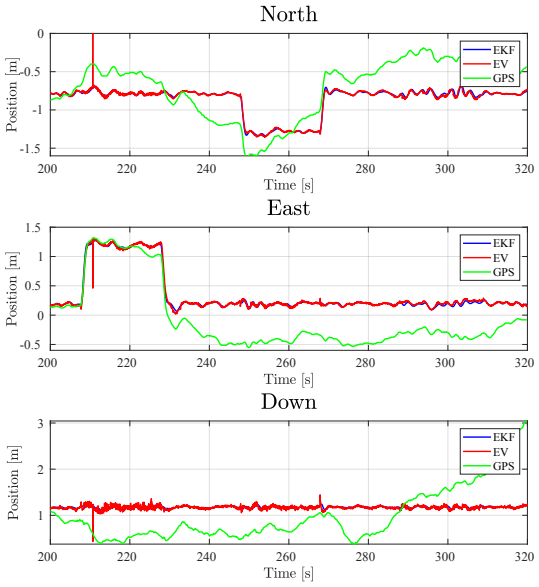

Figure 8.8: Estimated position provided by the vision-based state estimator with more aggressive gains

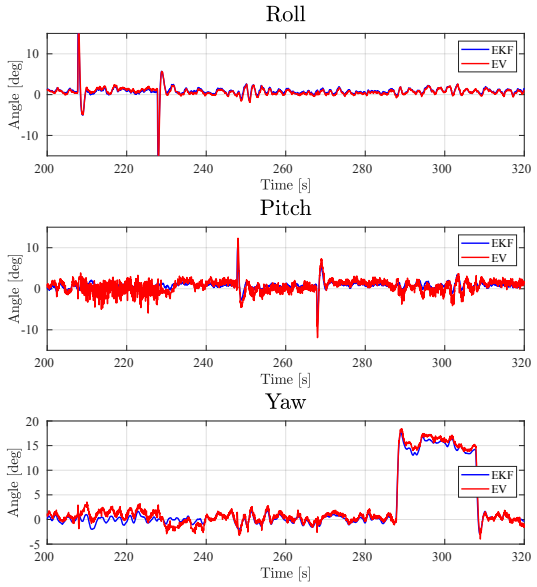

Figure 8.9: Estimated attitude provided by the vision-based state estimator with more aggressive gain

As can be seen, the more aggressive controllers were slow enough to not cause motion blur. The vehicle was also placed 2 m away from the markers, which caused multiple markers to be constantly visible.

To conclude, the less aggressive controller appears to be underdamped and slow. This behaviour could result from the close proximity to the markers because it had only one marker in frame at a time (the effect of which was shown in Section 6.6). Both the flights showed that the system was implemented successfully. The system proved to be remarkably robust to the loss of markers in its visual field, as it did not fail in the periods where no markers were visible. Stable flight was maintained throughout the tests.

8.3 Summary

The results were obtained by a series of flight tests. The tests followed consecutively, where every successive test was only performed when the previous test was successful. The test started with the fastest control loops and gradually tested the slower control loops, where GPS was used for the localisation. The waypoint scheduler was used for the positional control loops and, following the satisfactory behaviour of the controllers and estimator, the vision system was added and the GPS excluded. The vision-based system was first tested by manual control before handing the controls over to the waypoint scheduler and control system. All the flights were performed successfully.

Stellenbosch University https://scholar.sun.ac.za

Chapter 9

Conclusions and Recommendations

This project set out to design, implement and verify a vision based flight control system that controls the position of a quadrotor relative to an inspection target in a GPS-denied environment.A secondary goal was set to develop a UAV research platform using off-the-shelf UAV hardware and open-source software to replace the Electronic System Laboratory's (ESL's) in-house developed avionics, ground control station, and hardware-in-the-loop simulation environment. These goals were divided into ten objectives to ensure the goals were reached. These objectives were:

1. To select and procure a suitable off-the-shelf quadrotor UAV: an airframe, a flight control unit, an on-board computer, and a suite of sensors including an inertial measurement unit (IMU), GPS sensor, magnetometer, barometric sensor, and camera module.

2. To select suitable open-source flight control software to serve as the basis for the vision-based flight control system.

3. To create an integrated UAV system consisting of the UAV, the ground control station, and the hardware-in-the-loop simulation environment.

4. To establish a mathematical model of the quadrotor UAV flight mechanics that can be used for flight control design and analysis, and for simulations.

5. To reverse engineer the flight control system and the state estimators implemented by the open-source flight control software.

6. To re-design the flight controllers gains based on the flight dynamics of the chosen commercial off-the-shelf quadrotor UAV, if necessary.

7. To design and implement a vision-based localisation algorithm that executes in real time on-board the quadrotor UAV.

8. To design and implement a vision-based state estimator that estimates the position, velocity, and attitude of the quadrotor UAV without using GPS sensor measurements.

9. To design and implement a vision-based flight control and waypoint navigation system for the quadrotor UAV.

Stellenbosch University https://scholar.sun.ac.za

10. To verify the correct operation of the vision-based localisation, state estimation, flight control, and waypoint navigation using laboratory experiments, simulation tests, and practical flight tests.

This chapter will present how these objectives and goals were completed. The first section of this chapter contains a summary of the work done in this thesis. This will be followed by a discussion on the hardware and software that was used, where-after a discussion on the flight control system will be presented. A discussion of the vision-based localisation system is then presented, upon which the tests performed to verify the successful integration of the different systems will also be given. Lastly, recommendations will be made for future projects.

9.1 Summary of Work Done

This thesis begins with a review of the literature on the basic dynamics of a multi-rotor was, including the sensors that was required, the different methodologies for vision-based pose estimation and the open-source software packages that were considered. Thereafter follows a system overview of the specific hardware that was chosen and a discussion on the different ways in which the software packages and hardware were connected. This is followed by a demonstration of the mathematical models used to describe the dynamics of a quadrotor and the dynamics were linearised around hover. The linear model was used to design a control system that was based on the open-source flight control software of PX4's architecture. The control system was tested in a flight test and the simulation results were compared to that of the flight test. The vision-based localisation approach was designed, where the camera model used was first defined. Thereafter, an overview of ArUco's marker detection and pose estimation algorithm was given. Finally, the noise characteristics of the camera that was used was determined. The estimator architecture that incorporates the vision-based pose estimation and the IMU was described, as well as an overview of all of the changes that were made to them. The full system was implemented onto the vehicle and flight tests were performed without the use of GPS. The flight tests proved that the system was successfully integrated and worked as expected.

9.2 Off-the-shelf Hardware and Open-source Software

The first three objectives required the obtaining of open-source software, off-the-shelf hardware and the integration of different components in one UAV system. The Intel® Aero Ready-To-Fly drone that was procured had an on-board flight control unit (FCU), a separate on-board computer (OBC), and a range of sensors that included an inertial measurement unit (IMU), a GPS-receiver, a magnetometer, a barometric sensor, and a camera module.

A benefit of the Intel® Aero RTF drone is that it has an OBC on-board that allows the open source flight control software (PX4) to be loaded onto the FCU and ArUco's open-source vision-based pose estimation software to be loaded on the OBC. This allowed both the flight control and vision-based pose estimation software adequate computational power. The vision-based pose estimation software was written as a Robot Operating

Stellenbosch University https://scholar.sun.ac.za

System (ROS) node. Successful integration was achieved because the PX4 software supports ROS nodes.

Both PX4 and ROS has support for the open-source ground control software QGroundcontrol and the simulation software Gazebo. Gazebo enabled hardware-in-the-loop simulation. However, a complete system hardware-in-the-loop simulation was not achieved as a result of physical wiring restrictions on the hardware. Such difficulties with simulations were resolved via laboratory experiments that tested the system's hardware in a safe manner. These experiments, in addition to simulations, verified system integration before flight tests were conducted.

In conclusion, off-the-shelf hardware was procured, a suitable open-source flight control software was selected and an integrated UAV system was created. Therefore, the first three objectives listed were achieved.

9.3 Flight Control System

The next three objectives focused on the flight control system. In Chapter 4, differential equations of a quadrotor UAV flight mechanics were established and combined with the force and moment model, which provides a mathematical model of the quadrotor's flight mechanics. The model was linearised to provide the basis of the control system design and the Intel® Aero RTF drone's physical properties were determined.

PX4's control system was reverse engineered and an overview can be found in Chapter 5. The architecture used by PX4 was successive loop closure. The loops were (in order of fastest to slowest) (1) angular rate, (2) attitude, (3) velocity and (4) position; where the angular rate and velocity loops used a two-degree-of-freedom PID (2 DoF PID) controller and the attitude and position loop used proportional control. The architecture of PX4 was followed and the flight controller gains were re-designed for the Intel® Aero RTF drone. The gains were implemented in PX4's flight control system and was verified by simulations and flight tests.

A functional flight control system was therefore available, satisfying the fourth, fifth and sixth objectives.

9.4 Vision-based Localisation

Objectives 7 to 9 were based on the vision-based localisation algorithm. ArUco's marker detection and pose estimation algorithms were used with ArUco's square markers. The markers had a black square border around a 5x5 grid of either black or white blocks, which encoded the markers ID. The marker detection algorithm consisted of two steps. The first step was the detection of all possible markers by finding the square borders in an image via a combination of adaptive thresholding and contouring. The second step consisted of verifying each detected border and identify the ID by decoding the 5x5 grid within the border. Each detected marker was then matched to the corresponding points on a map of the environment. Each detected marker's four corner points, the 2D projection and the 3D points from the map were used to solve the PnP problem. This provided a pose estimate of the camera relative to the inertial axis. The estimate was then sent to the estimator, where it was combined with the IMU measurements to estimate the pose of the quadrotor UAV. The marker detection and pose estimation was tested in simulation and by practical laboratory tests that used the actual camera

Stellenbosch University https://scholar.sun.ac.za

and on-board computer. The position and attitude accuracy of the pose estimation system was experimentally determined, and were used to set the sensor measurement noise covariances for the vision-based state estimator. The first actual test consisted of moving the vehicle manually in front of the actual markers and the estimation of both the vision system and EKF were analysed to ensure that the system functioned correctly. The camera was accurate within 50 mm in position and 2.5° in orientation. The EKF followed the vision estimate closely and could predict the movement of the vehicle even when no markers were detected. Objectives 7 to 9 was therefore achieved, where the second part of objective 9 (the waypoint navigation will be proved in the next section).

9.5 Flight Test

The full integration of all the different systems and hardware was tested in a series of flight tests. The controllers and the GPS-based estimator were first tested (with the GPS enabled) by a safety pilot that commanded the references for each of the control loops directly. The test was successful and proved that the flight control system was functional.

The following test was done with the waypoint scheduler controlling the quadrotor. The waypoint scheduler successfully commanded a series of waypoints to be followed at an altitude of 10 m, which proved that the control system and waypoint scheduler was successfully integrated.

After successful testing of the waypoint scheduler, the visual system was armed and the quadrotor UAV was placed in front of a jig that held 14 markers. The safety pilot flew the quadrotor UAV level with the top row of markers and switched over to the waypoint scheduler. The quadrotor UAV then flew a series of steps in the lateral and longitudinal direction and made a yaw movement. This flight proved that the integration of the vision-based localisation, state estimation, flight control and waypoint navigation systems was successfully implemented.

The final test flight achieved the primary goal that was set out, which was the successful flight of a quadrotor UAV in a GPS-denied environment with vision-based flight control and state estimation.

9.6 Recommendations/Future Work

The work described in this thesis forms part of a larger project, which still requires the addition of post-flight processing and analysis.

1. The vision-based pose estimation process as set out in this thesis, is reliant on accurately placed markers that are relatively close to one another. The author recommends that further investigation into different techniques are required, such as markerless pose estimation.

2. The more aggressive controller showed unstable vertical position dynamics that requires further investigation. The more aggressive controller shows promise, since the more aggressive controllers did not cause motion blur.

3. At present, the state estimation process does not include the attitude in the output prediction stage. It should be investigated whether the addition of the attitude

Stellenbosch University https://scholar.sun.ac.za

will cause a computational load that is too large for the FCU.

4. The system can be upgraded by replacing the FCU with one that is connected to the OBC directly and has an additional open serial port on the FCU to connect to an external computer, for example the Pixhawk range. This will allow the fully combined UAV system to be tested in HITL simulations.

5. The OBC of the Intel® Aero RTF drone does allow the architecture to be changed with FPGA and could possibly solve the HITL simulation problems by activating one of the two unused open UART channels.

Stellenbosch University https://scholar.sun.ac.za

List of References

[1] J.H. Blakelock, 1991, *Automatic control of aircraft and missiles*, 2nd edn, John Wiley and Sons, New York, United States of America.

[2] M.V. Cook, 2013, *Flight Dynamics Principles: A Linear Systems Approach to Aircraft Stability and Control*, 3rd edn, Elsevier Butterworth-Heinemann, Oxford, United Kingdom.

[3] B. Etkin & L.D. Reid, 1996, *Dynamics of Flight (Stability and control)*, 3rd edn, John Wiley and Sons, New York, United States of America.

[4] R.C. Hibbeler, 2016, *Engineering Mechanics Dynamics*, 14th edn, Pearson Prentice Hall, Hoboken, United States of America.

[5] P.D.S. Möller, 2015, 'Automated Landing of a Quadrotor Unmanned Aerial Vehicle on a Translating Platform', MA thesis, University of Stellenbosch, accessed 20 May 2018 from the University of Stellenbosch library database.

[6] J. Treurnicht, 2006, 'Two-Rope Inertia Calculation', technical report. Stellenbosch University, accessed 17 August 2018 from the University of Stellenbosch library database.

[7] R. Hartley & A. Zisserman, 2004, *Multiple View Geometry*, 2nd edn, Cambridge University Press, Cape Town, South Africa.

[8] M. Araki & H. Taguchi, 2003, 'Two-Degree-of-Freedom PID Controllers', *International Journal of Control, Automation, and Systems*, vol. 1, no.4, pp.401-411.

[9] I.M. Horowitz, 1963, *Synthesis of Feedback Systems*, Academic Press Inc. (London) Ltd., London, United Kingdom.

[10] P. Tripicchio, M. Unetti, N. Giordani, A. Avizzana & M. Satler, 2014, 'A Lightweight SLAM Algorithm for Indoor Autonomous Navigation', *Proceedings of Australasian on Robotics and Automation*, The University of Melbourne, Melbourne, Australia, 2-4 December.

[11] M. Li & A.I. Mourikis, 2013, 'High-precision, consistent EKF-based visual-inertial odometry', *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690-711.

[12] S. Weiss, D. Scaramuzza & R. Siegwart, 2011, 'Monocular-SLAM–based navigation for autonomous micro helicopters in GPS-denied environments', *Journal of Field Robotics (Special Issue: Safety, Security, and Rescue Robotics)*, vol. 28, no.6, pp. 854-874.

Stellenbosch University https://scholar.sun.ac.za

[13] A. Mourikis & S. Roumeliotis, 2007, 'A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation', in Mourikis, *2007 IEEE International Conference on Robotics and Automation*, Rome, Italy, 10-14 April, pp. 3565-3572.

[14] A. Assa & F. Janabi-Sharifi, 2014, 'A Robust Vision-Based Sensor Fusion Approach for Real-Time Pose Estimation', *IEEE Transactions on Cybernetics*, vol. 44, no.2, pp. 217-227.

[15] N. Metni & T. Hamel, 2006, 'A UAV for bridge inspection: Visual servoing control law with orientation limits', *Automation in Construction*, vol. 17, no. 1, pp. 3 -10.

[16] R. Konomura & K. Hori, 2016, 'FPGA-based 6-DoF Pose Estimation with a Monocular Camera Using Non Co-planer Marker and Application on Micro Quadcopter', *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Daejeon, Korea, 9-14 October, pp. 4250-4257.

[17] S. Vogt, A. Khamene, F. Sauer & H. Niemann, 2003, 'Single camera tracking of marker clusters: multiparameter cluster optimization and experimental verification', *2002 IEEE International Symposium on Mixed and Augmented Reality*, Darmstadt, Germany, 1 October, pp. 127-136, DOI:10.1109/ISMAR.2002.1115082.

[18] L. Jayatilleke & N. Zhang, 2013, 'Landmark-Based Localization for Unmanned Aerial Vehicles', *2013 IEEE International Systems Conference*, Orlando, United States of America, 15-18 April, pp. 448-451, DOI:10.1109/SysCon.2013.6549921.

[19] W. Roozing & A.H. Göktogan, 2013, 'Low-Cost Vision-Based 6-DOF MAV Localization Using IR Beacons', *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Wollongong, Australia, 9-12 July, pp. 1003-1009, DOI:10.1109/AIM.2013.6584225.

[20] T.T. Mac, C. Copot, R. De Keyser & C.M. Ionescu, 2018, 'The development of an autonomous navigation system with optimal control of an UAV in partly unknown indoor environment', *Mechatronics*, vol. 49, no. 1, pp. 187-196.

[21] G. Bleser, H. Wuest & D. Stricker, 2006, 'Online camera pose estimation in partially known and dynamic scenes', *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*, Santa Barbara, United States of America, 22-25 October, pp. 56-65, DOI:10.1109/ISMAR.2006.297795.

[22] H. Jin, P. Favaro & S. Soatto, 2000, 'Real-time 3D motion and structure of point features: a front-end system for vision-based control and interaction', *2000 IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, United States of America, 15 June, vol. 2, pp. 778- 779, DOI:10.1109/CVPR.2000.854954.

[23] P. Lu & Q. Geng, 2011, 'Real-time Simulation System for UAV Based on Matlab (Simulink)', *2011 IEEE 2nd International Conference on Computing, Control and Industrial Engineering*, Wuhan, China, 20-21 August, pp.399-404, DOI:10.1109/CCIENG.2011.6008043.

[24] M. Veth, J. Raquet & M. Pachter, 2006, 'Stochastic Constraints for Efficient Image Correspondence Search', *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no.3, pp.973-982, DOI:10.1109/TAES.2006.4439212.

Stellenbosch University https://scholar.sun.ac.za

[25] S. Shen, Y. Mulgaonkar, N. Michael & V. Kumar, 2013, 'Vision-Based State Estimation and Trajectory Control Towards High-Speed Flight with a Quadrotor', in P. Newman, D. Fox & D. Hsu, *Robotics: Science and Systems IX*, Technische Universität Berlin, Berlin, Germany, 24-28 June, DOI:10.15607/RSS.2013.IX.032.

[26] S. Shen, Y. Mulgaonkar, N. Michael & V. Kumar, 2013, 'Vision-Based State Estimation for Autonomous Rotorcraft MAVs in Complex Enviroments', *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 6-10 May, pp. 1758-1764, DOI:10.1109/ICRA.2013.6630808.

[27] E.S. Jones & S. Soatto, 2011, 'Visual-inertial navigation, mapping and localization: A scalable real-time casual approach', The International Journal of Robotics Research, vol. 30, no. 4, pp. 407-430.

[28] A. Zarándy, M. Nemeth, Z. Nagy, A. Kiss, L. Santha & T. Zsedrovits, 2016, 'A real-time multi-camera vision system for UAV collision warning and navigation', *Journal of Real-Time Image Processing*, vol. 12, no. 4, pp.709-724.

[29] C. Nitschke, 2014, 'Marker-based Tracking with Unmanned Aerial Vehicles', *2014 IEEE International Conference on Robotics and Biomimetics*, Bali, Indonesia, 5-10 December, pp. 1331-1338, DOI:10.1109/ROBIO.2014.7090518.

[30] D. Mellinger & V. Kumar, 2011, 'Minimum Snap Trajectory Generation and Control for Quadrotors', *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 9-13 May, pp. 2520-2525, DOI:10.1109/ICRA.2011.5980409.

[31] L.R García Carrillo, A.E. Dzul López, R. Lozano & C. Pégard, 2013, *Quad Rotorcraft Control: Vision-based Hovering and Navigation*, 1st edn, Springer-Verlag London, London.

[32] A. Benini, M.J. Rutherford & K.P. Valavanis, 2016, 'Real-time, GPU-based Pose Estimation of a UAV for Autonomous Takeoff and Landing', *2016 IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, 16-21 May, pp. 3463-3470, DOI:10.1109/ICRA.2016.7487525.

[33] S. Zhou, G. Flores, E. Bazan, R. Lozano & A. Rodriguez, 2015, 'Real-Time Object Detection and Pose Estimation using Stereo Vision. An application for a Quadrotor MAV', *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems*, Cancun, Mexico, 23-25 Nov, pp. 72-77, DOI:10.1109/RED-UAS.2015.7440992.

[34] C. Eschmann, C.M. Kuo, C.H. Kuo & C. Boller, 2012, 'Unmanned Aircraft Systems for Remote Building Inspection and Monitoring', *6th European Workshop on Structural Health Monitoring*, Dresden, Germany, 3-6 July, pp. 1-8.

[35] G. Balamurugan, J. Valarmathi & V.P.S. Naidu, 2016, 'Survey on UAV navigation in GPS denied environments', *2016 IEEE International Conference on Signal Processing, Communication, Power and Embedded System*, Paralakhemundi, India, 3-5 October, pp. 198-204, DOI:10.1109/SCOPES.2016.7955787.

Stellenbosch University https://scholar.sun.ac.za

[36] P.H. Nguyen, K.W. Kim, Y.W. Lee & K.R. Park, 2017, 'Remote Marker-Based Tracking of UAV Landing Using Visible-Light Camera Sensor', *Sensors*, vol. 17, no. 9, Article no. 1987.

[37] A.M. Singh, Q.P. Ha, D.K. Wood & M. Bishop, 2017, 'Low-latency Vision-based Fiducial Detection and Localisation for Object Tracking', *34th International Symposium on Automation and Robotics in Construction*, Waterloo, United States of America, 28 June - 1 July, pp. 706-711.

[38] J. Bacik, F. Durovsky, P. Fedor & D. Perdukova, 2017, 'Autonomous flying with quadrocopter using fuzzy control and ArUco markers', *Intelligent Service Robotics*, vol. 10, no. 3, pp. 185-194.

[39] S. Garrido-Jurado, R. Muños-Salinas, F.J. Madrid-Cuevas & M.J. Marín-Jiménez, 2014, 'Automatic generation and detection of highly reliable fiducial markers under occlusion', *Pattern Recognition*, vol. 47, no.6, pp. 2280-2292.

[40] G. Klein & D. Murray, 2007, 'Parallel Tracking and Mapping for Small AR Workspace', *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, IEEE Computer Society Washington, USA, 13-16 November, pp. 1-10, DOI: 10.1109/ISMAR.2007.4538852

[41] R. Mur-Artal, J.M.M. Montiel & J.D. Tardós, 2015, 'ORB-SLAM: A versatile and accurate monocular SLAM system', *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147-1163, DOI:10.1109/TRO.2015.2463671.

[42] I. Cvišić, J. Ćesić, I. Marković & I. Petrović, 2017, 'SOFT-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles', *Journal of Field Robotics*, vol. 35, no. 4, pp. 578-595, DOI: 10.1002/rob.21762

[43] T. Whelan, S. Leutenegger, R.F. Salas-Moreno, B. Glocker & A.J. Davison, 2015, 'ElasticFusion: Dense SLAM without a pose graph', *Robotics: Science and Systems XI*, Sapienza University of Rome, Rome, Italy, 12-17 July, article no. 1, DOI: 10.15607/RSS.2015.XI.001.

[44] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohli, J. Shotton, S. Hodges & A. Fitzgibbon, 2011, 'KinectFusion: Real-time dense surface mapping and tracking', *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, Basel, Switzerland, 26-29 October, 10.1109/ISMAR.2011.6092378.

[45] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J.J. Leonard & J. McDonald, 2014, 'Real-time large scale dense RGB-D SLAM with volumetric fusion', *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598-626, DOI: 10.1177/0278364914551008.

[46] T. Whelan, M. Kaess, J.J Leonard & J. McDonald, 2013, 'Deformation-based Loop Closure for Large Scale Dense RGB-D SLAM', *2013 IEEE/RJS International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 3-7 November, pp. 548-555, DOI:10.1109/IROS.2013.6696405.

Stellenbosch University https://scholar.sun.ac.za

[47] T. Whelan, H. Johannsson, M. Kaess, J.J. Leonard & J. McDonald, 2013, 'Robust real-time visual odometry for dense RGB-D mapping', *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 6-10 May, pp. 5724-5731, DOI: 10.1109/ICRA.2013.6631400.

[48] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson & J.J. Leonard, 2012, 'Kintinuous: Spatially extended KinectFusion', technical report, Massachusetts Institute of Technology, accessed 10 September 2019 from CSAIL Technical Reports (July 1, 2003 – present).

[49] J.J. Leonard, M. Kaess, J. McDonald & T.J. Whelan, 2013, *Method for mapping an environment*, US9412173B2.

[50] A.J. Davison, I.D. Reid, N.D. Molton & O. Stasse, 2007, 'MonoSLAM: Real-time single camera SLAM', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052-1067, DOI: 10.1109/TPAMI.2007.1049.

[51] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid & J.J. Leonard, 2016, 'Past, present, and future of simultameous localization and mapping: Towards the robust-perception age', *IEEE Transaction on Robotics*, vol. 32, no. 6, pp. 1309-1332, DOI: 10.1109/TRO.2016.2624754.

[52] J. Engel, V. Koltun & D. Cremers, 2018, 'Direct Sparse Odometry', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611-625, DOI:10.1109/TPAMI.2017.2658577.

[53] C. Tisse, T. Fauvel & H. Durrant-Whyte, 2005, 'A micro aerial vehicle motion capture system', *1st International Conference on Sensing Technology*, Palmerston North, New Zealand, 21-23 November, pp. 533-538.

[54] S. Hutchinson, G.D Hager, & P.I Corke, 1996, 'A tutorial on visual servo control', *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670.

[55] R. Mebarki, V. Lippiello & B. Siciliano, 2015, 'Nonlinear visual control of unmanned aerial vehicles in GPS-denied enviroments', *IEEE Transaction on Robots*, vol. 31, no. 4, pp. 1004-1017.

[56] M.G. Popova & H.H.T. Liu, 2016, 'Position-based visual servoing for target tracking by a quadrotor uav', *AIAA Guidance, Navigation, and Control Conference*, San Diego, United States of America, 4-8 January, pp. 1-12, DOI:10.2514/6.2016-2092.

[57] E. Malis, F. Chaumette & S. Boudet, 1999, '2 ½ d visual servoing', *IEEE Transactions on Robotics and Automation*, vol. 15, no. 1, pp. 238-250.

[58] A. Driss, L. Krichen, F. Mohamed & L.C. Fourati, 2018, 'Simulation tools, environments and frameworks for UAV system analysis', *14th International Wireless Communications and Mobile Computing Conference*,Limassol, Cyprus, 25-29 June, pp. 1495-1500, DOI:10.1109/IWCMC.2018.8450505.

[59] ArduPilot, 2018, 'Code Overview (Copter)', [Online] Available at : http://ardupilot.org/dev/docs/apmcopter-code-overview.html, [2019, October 10].

Stellenbosch University https://scholar.sun.ac.za

[60] PX4, 2019, 'Developers Guide', [Online] Available at : https://px4.io/developer-guide/, [2019, October 10].

[61] J.E. Bortz, 1971, 'A new mathematical formulation for strapdown inertial navigation', *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-7, no. 1, pp. 61-66, DOI:10.1109/TAES.1971.310252.

[62] P.G. Savage, 1998, 'Strapdown inertial navigation algorithm design part 1: Attitude algorithm', *Journal of Guidance, Control and Dynamics*, vol. 21, no. 1, pp. 19-28.

[63] P.G. Savage, 1998, 'Strapdown inertial navigation algorithm design part 2: Velocity and position algorithm', *Journal of Guidance, Control and Dynamics*, vol. 21, no. 2, pp. 208-221.

[64] R.B. Miller, 1983, 'A new strapdown attitude algorithm', *Journal of guidance, control, and dynamics*, vol. 6, no. 4, pp. 287-291, DOI:10.2514/3.19831.

[65] P.G. Savage, 2015, 'Computational elements for strapdown systems', *Strapdown associates*, vol. WBN-14010, pp. 1-39.

[66] A. Khosravian, J. Trumpf, R. Mahony & T. Hamel, 2015, 'Recursive attitude estimation in the presence of multi-rate and multi-delay vector measurements', *2015 American control conference*, Palmer House Hilton, Chicago, USA, 1-3 July, pp. 3199-3205.

[67] E. Chan, 2017, 'Strapdown inertial navigation system', *ECE Senior Capstone Project*, Technical notes, [Online] Available at : https://sites.tufts.edu/eeseniordesignhandbook/files/2017/05/Chan_SHP_FINAL-NoLinks.pdf, [2019, October 29].

[68] T. Kurita, N. Otsu, & N. Abdelmalek, 1992, 'Maximum likelihood thresholding based on population mixture models', *Pattern Recognition*, vol. 25, no. 10, pp. 1231-1240, DOI:10.1016/0031-3203(92)90024-D

Stellenbosch University https://scholar.sun.ac.za

Appendix A

Intel® Aero Ready To Fly Specifications

A.1 Intel® Aero Compute Board:

- Intel® Atom™ x7-Z8750 processor quad-core
- 4 GB LPDDR3-1600
- 32 GB eMMC
- Intel® Dual Band Wireless-AC 8260
- USB 3.0 OTG
- Reprogrammable I/O via Altera® Max® 10 FPGA
- 8MP RGB camera (front-facing) – one of the 3 camera modules included with the Aero Vision Accessory Kit
- VGA camera, global shutter, monochrome (down-facing)
- Insyde Software InsydeH2O* UEFI BIOS optimized for the Intel® Aero Platform for UAVs.

A.2 Intel® RealSense™ camera (R200)

- $0.5m$ - $3.5m$ Operating Range
- 640 x 480 Resolution @ 30fps
- 2 Infrared Cameras
- Active Depth Sense through Stereo

Stellenbosch University https://scholar.sun.ac.za

A.3 Intel® Aero Flight Controller:

- STM32 F427V microcontroller
- Temperature compensated: 6 DoF IMU, magnetometer, and altitude sensors
- Connected to the Aero Compute board over HSUART and communicates using MAVLink* protocol

A.4 Pre-assembled quadcopter:

- Carbon fiber air-frame
- GPS and compass
- Power distribution board
- 4 Yuneec Tyhpoon H electronic speed controllers
- 4 Yuneec Tyhpoon H motors
- Yuneec Tyhpoon H snap-on propellers
- Spektrum DSMX Serial Receiver
- Spektrum DXe Transmitter (2.4GHz DSMX)

Stellenbosch University https://scholar.sun.ac.za

Appendix B

Controller Gains

B.1 PX4 vs. Custom

Controllers	Symbol	Custom Gains	PX4 Gains
Roll Rate P	K_{P_p}	0.1300	0.1300
Roll Rate I	K_{I_p}	0.0700	0.0700
Roll Rate D	K_{D_p}	0.0012	0.0012
Pitch Rate P	K_{P_q}	0.1300	0.1300
Pitch Rate I	K_{I_q}	0.0700	0.0700
Pitch Rate D	K_{D_q}	0.0012	0.0012
Yaw Rate P	K_{P_r}	0.1200	0.11999999973
Yaw Rate I	K_{I_r}	0.0500	0.05000000745
Yaw Rate D	K_{D_r}	0.0000	0.0000
Roll P	K_{P_ϕ}	4.0000	8.0000
Pitch P	K_{P_θ}	3.0000	8.0000
Yaw P	K_{P_ψ}	1.0000	4.0000
North Velocity P	K_{P_u}	0.0800	0.1500
North Velocity I	K_{I_u}	0.0100	0.0200
North Velocity D	K_{D_u}	0.0050	0.0100
East Velocity P	K_{P_v}	0.0800	0.1500
East Velocity I	K_{I_v}	0.0100	0.0200
East Velocity D	K_{D_v}	0.0050	0.0100
Down Velocity P	K_{P_w}	0.8000	0.8000
Down Velocity I	K_{I_w}	0.0150	0.0150
Down Velocity D	K_{D_w}	0.0000	0.0000
North Position P	K_{P_n}	0.5000	1.5000
East Position P	K_{P_e}	0.5000	1.5000
Down Position P	K_{P_d}	1.0000	1.0000

Table B.1: Controller Gains