Automated Recharging and Vision-Based Improved Localisation for a Quadrotor UAV

by

P.R. Grobler

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 W Jordaan

March 2021

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.

Abstract

In the growing field of autonomous multirotor unmanned aerial vehicles (UAVs) applications, one of the biggest challenges to achieving full autonomy is the limited flight duration. While improvements to existing battery technology is on the way, a solution is required to allow any UAV to perform self recharging without any human intervention and indefinitely increase any mission duration. Such a system could potentially unlock more UAV applications which currently seem impractical. This could allow UAVs to autonomously perform any task in remote locations and extend the radius of operation to far beyond what is currently possible.

This thesis presents an autonomous recharging solution which involves a UAV performing an accurate vision based autolanding on a custom charging platform prior to charging. A direct contact approach design is proposed which makes use of onboard electrodes and charging pads located on the charging platform for electrical energy transfer. This system is designed to enable any UAV to self recharge with zero human intervention.

A quadrotor UAV was built for the purpose of practically demonstrating the proposed solution. A Pixhawk flight controller using PX4 was chosen as the avionics of the vehicle. Custom controller gains were designed for the UAV in order to satisfy the desired flight characteristic requirements. An autonomous vision based autolanding strategy for this vehicle was designed and simulated using software-in-the-loop (SITL) and hardware-in-the-loop (HITL) simulations in Gazebo simulator. This allowed for the actual PX4 flight control software to be executed on an accurate simulated model of the UAV prior to practical flight tests. A ROS package was developed for the autolanding which was responsible for image processing, publishing control setpoints and processed visual feedback data to the flight controller for accurate control during the autolanding. An average landing accuracy of 4cm was achieved in practical flight tests using this strategy.

In many UAV applications highly accurate localisation of the UAV is required which is unachievable using traditional Global Navigation Satellite System (GNSS). This thesis presents a solution which involves using the charging stations as visual landmarks in order to localise the UAV. A marker based Simultaneous Localisation and Mapping (SLAM) strategy is implemented which aims to map the charging stations in the inertial frame while simultaneously localising the UAV within the same reference frame. The SLAM output is published to the flight controller where it is included in the UAV's state estimation process using PX4's EKF2. Simulation and practical results are provided which showed that the localisation accuracy could be greatly improved provided that the stations were accurately mapped and that the stations were clearly visible.

Uittreksel

In die groeiende veld van outonome multirotor UAV toepassings, is die beperkte vlugduur een van die grootste uitdagings om volle outonomie te bewerkstellig. Terwyl verbeterings aan bestaande battery tegnologie op pad is, is 'n oplossing nodig om enige UAV self te laat laai sonder menslike ingryping en die duur van die missie onbepaald te verhoog. So 'n stelsel kan moontlik meer UAV-toepassings ontsluit wat tans onprakties lyk. Dit kan UAV's in staat stel om enige taak op afgeleë plekke outonoom uit te voer en die werkstraal verder te brei as wat tans moontlik is.

Hierdie proefskrif bied 'n outonome herlaaioplossing aan wat beteken dat 'n UAV 'n akkurate visie-gebaseerde outolanding op 'n aangepaste laadplatform moet uitvoer voordat dit gelaai word. 'n Ontwerp vir direkte benaderings word voorgestel wat gebruik maak van aan boord elektrodes en laaikussens wat op die laaiplatform geleë is vir die oordrag van elektriese energie. Hierdie stelsel is ontwerp om enige UAV in staat te stel om self te herlaai sonder menslike ingryping.

'n Quadrotor UAV is gebou om die voorgestelde oplossing prakties te demonstreer. 'n Pixhawk vliegbeheerder en PX4 vlugbeheer sagteware, is gekies as die lugvaart van die voertuig. Aangepaste beheerderstygings is vir die UAV ontwerp om aan die verlangde vlugkenmerkvereistes te voldoen. 'n Outonome visie-gebaseerde outolandingstrategie vir hierdie voertuig is ontwerp en gesimuleer met behulp van SITL en HITL simulasies in Gazebo-simulator. Hierdeur kon die werklike PX4-vliegbeheersagteware volgens 'n akkurate gesimuleerde model van die UAV uitgevoer word voor praktiese vlugtoetse. 'n ROS-pakket is ontwikkel vir die outolanding wat verantwoordelik was vir beeldverwerking, die publisering van kontrolepunte en verwerkte visuele terugvoerdata aan die vlugbeheerder vir akkurate beheer tydens die outolanding. 'n Gemiddelde landingsakkuraatheid van 4 cm is in praktiese vlugtoetse met behulp van hierdie strategie bereik.

In baie UAV-toepassings is baie akkurate lokalisering van die UAV nodig, wat onhaalbaar is met behulp van tradisionele GNSS. Hierdie tesis bied 'n oplossing aan wat die gebruik van laaistasies as visuele bakens gebruik om die UAV te lokaliseer. 'n Merkergebaseerde SLAM strategie word geïmplementeer wat daarop gemik is om die laaistasies in die traagheidsraamwerk te karteer terwyl die UAV terselfdertyd binne dieselfde verwysingsraamwerk geplaas word. Die SLAM uitset word aan die vlugbeheerder gepubliseer, waar dit ingesluit word by die beraming van die UAV's met behulp van EXF2 van PX4. Simulasie en praktiese resultate word verskaf wat getoon het dat die lokaliseringsakkuraatheid aansienlik kan verbeter, mits die stasies akkuraat gekarteer is en dat die stasies duidelik sigbaar is.

Acknowledgements

I would like to thank the following:

- Dr. Willem Jordaan for his knowledgeable insight, advice and support during this project.
- The other lecturers at the ESL for their advice, including Dr Japie Engelbrecht, Dr Corné van Daalen, Prof Herman Steyn, Mr Arno Barnard, Dr Lourens Visagie and Mr JC Schoeman.
- The technical staff at Stellenbosch University, including Wessel Croukamp, Johan Arendse and Wynand van Eeden.
- My fellow postgraduate students and friends for their own input and advice, including Mr Anton Erasmus, Henry Kotze, Francois Slabber, Johan Ubbink, Ruan Viljoen, Martin Babl, Armand Scholtz and Daniel Jansen.
- My family Riaan Grobler, Ilze Grobler and Elzé Grobler for their love and support.

Contents

List of Figures

List of Tables

Acronyms

CMS charge management system. 83

DCA direct contact approach. 80

DCM direct cosine matrix. 32

DGPS Differential Global Position System. 9

DoF degree-of-freedom. 9, 10

EKF Extended Kalman Filter. 25

ENU East-North-Up. 30

ESC electronic speed controller. 17

ESL Electronic Systems Laberatory. 12

EV external vision. 88

FCU flight control unit. 51

FPV first person view. 16

GNSS Global Navigation Satellite System. ii, 2, 17

GPU graphics processing unit. 16

HITL hardware-in-the-loop. ii, iii, 11, 12

IMU inertial measurement unit. 17

IR Infrared. 8, 10

LED light emitting diode. 8

LIDAR Light Detection and Ranging. 4

LiPo lithium polymer. 17

MAVLink micro air vehicle link. 28

NED North-East-Down. 25

OBC onboard computer. 13, 17

PDB power distribution board. 85

PWM pulse width modulation. 17

QR quick response. 8

RC remote control. 17

ROS robotic operating system. 13, 15

RTOS real time operating system. 12

SDF simulation description format. 26

SITL software-in-the-loop. ii, iii, 11, 12

SLAM Simultaneous Localisation and Mapping. ii, iii, 25

UART universal asynchronous receiver/transmitter. 27

UAV unmanned aerial vehicle. ii, iii, 1, 4, 15

UDP user datagram protocol. 26

VO visual odometry. 9

WPT wireless power transfer. 6

Chapter 1

Introduction

1.1 Background

The use and applications of multirotor UAVs have grown exponentially in the last few years. This is largely due to their advantages over existing aerial vehicles. Fixed wing aircraft require lots of space to take off and land whereas helicopters are expensive to run and maintain. Multirotors on the other hand have few mechanical parts which make them inexpensive and simple to maintain. Relatively speaking, multirotors are easy to pilot and control and do not require large open space for take off and landing, which makes them an ideal aerial platform for most applications.

In most cases the UAV is required to carry some form of payload. Examples of payloads include cameras of varying size, specialised scientific instrumentation, package delivery and farming equipment used for crop spraying and planting. With the increase in payload mass the total flight time of the UAV is decreased as the motors require more current from the batteries to produce the required lift.

1.1.1 Automated Self Recharging

Although improvement in battery technology is giving rise to batteries that are more powerful and longer lasting, the limitation on flight time is still the biggest limiting factor to achieving fully autonomous UAV missions. Human intervention is required to swap batteries or charge the UAV's batteries before resuming a mission. To fully automate the system the UAV must be able to perform self recharging. This will save time and money and can potentially unlock more applications for autonomous UAVs. This thesis will propose a system by which UAVs could be given the ability to self recharge.

A self charging UAV is able to perform any task indefinitely. Without the need for a human operator the UAV could be permanently deployed in an autonomous mission. The flight radius of the UAV can also be extended by placing a network of charging stations in the UAV's environment.

This solution is not new as a few companies are attempting self recharge. Amazon's Prime Air UAV delivery project aims to perform UAV self recharging on tall structures such as street lights and buildings according to a new patent submitted by the company in 2016 [1]. With this technology their UAVs are able to have an extended delivery range and ensure that their vehicles remain charged and ready for new deliveries without human intervention.

1.1.2 Autolanding

Before the UAV is able to self recharge it would first need to land on a charging platform. The UAV must be able to complete this landing fully autonomously and with enough precision to reliably land on the platform each time. A solution to achieving practical and accurate autolandings on a charging platform needs to be developed. Traditional GNSS sensors located on most UAVs lack the required accuracy to perform precise landings. Another method of localising the UAV during the landing process needs to be explored. An accurate simulation environment needs to be created in order to rigorously test the landing strategy before attempting practical landings.

1.1.3 Improved Localisation

In some applications the UAV is required to have accurate knowledge of its world frame position. Examples of these scenarios include environments with obstacles that need to be avoided or in some surveillance and mapping applications the UAV needs accurate position control to observe a target or to map a new target. A method by which the general localisation accuracy of the UAV can be improved will also be investigated. This solution needs to be low cost and aim to achieve improved localisation without any additional hardware on the UAV.

1.2 Project Definition

This project aims to design and practically implement a quadrotor UAV capable of performing practical vision based autolandings on a charging platform. The project aims to provide a detailed design of the physical charging station along with all custom electronics required by the charging process. Lastly this project aims to show practical results of an improved localisation system.

The following steps are used to solve the above mentioned problems:

1. Identify existing solutions to automated UAV recharging, vision based autolanding and marker based SLAM.

2. Design and build a custom quadrotor suitable for this project.

3. Determine the physical quadrotor parameters in order to create a simulated model of actual UAV.

4. Derive a mathematical model of a quadrotor in order to design custom controller gains appropriate for this project.

5. Create a simulation environment where the UAV systems can be tested.

6. Design, simulate and practically demonstrate an accurate autolanding system with the practical UAV.

7. Design a physical charging station as well as electronics required for self recharging.

8. Design, simulate and practically demonstrate a marker based SLAM system to improve the existing localisation accuracy of the UAV.

1.3 Thesis Outline

This section describes the layout of this thesis.

Chapter 2 is a literature study on current solutions to this project's objectives. A critical analysis is given on the solutions presented in the context of this project's requirements.

Chapter 3 discusses the physical UAV's hardware selection and design and also determines the UAV's physical parameters. At the end of Chapter 3 the software toolchain selected for this project is discusses in more detail.

Chapter 4 presents a mathematical formulation of a quadrotor model.

Chapter 5 presents a design method for the custom controller gains of the UAV. Here PX4's controller architecture is presented.

Chapter 6 discusses the design, simulation and practical implementation of an accurate vision based autolanding strategy.

Chapter 7 proposes a novel charging station design as well as custom landing gear that enables UAVs to perform self recharge.

Chapter 8 presents a marker based SLAM solution to improve the existing localisation of a UAV.

Chapter 9 provides a conclusion to the project and discusses possible future work.

Chapter 2

Literature Study

This chapter provides a literature study on automated recharging and improved localisation of a quadrotor UAV. The literature study firstly provides a brief overview of multirotor UAV applications and how they can benefit from automated recharging. Thereafter, different methods used to recharge UAVs is discussed. Methods and sensor configurations used to perform accurate autolandings with quadrotors, required by the autonomous recharging process is also discussed. Thereafter, different localisation methods used by autonomous vehicles is presented and finally the different available software packages used for autonomous UAV development is investigated.

2.1 Multirotor Applications

Surveying and mapping is one application which has largely been changed with the introduction of UAVs by allowing for measurements with the same level of quality and accuracy as traditional methods but can now be done at a fraction of the time. It involves using cameras and Light Detection and Ranging (LIDAR) payloads to generate 3D models, elevation models and geo-referenced orthomosaics of a projected area. This is useful for GIS professionals who can extract information such as highly accurate distances or volumetric measurements [46]. The radius of the targeted surveying area could be extended by placing a recharge station on the perimeter of the UAV's possible operating area. This would allow for the UAV to replenish its battery capacity before moving into a new area. Surveying areas can be linked by recharging stations, greatly extending the UAV's range.

Security, anti-poaching, wildlife monitoring and live event streaming are areas which have gathered increased interest in UAV technology [47] and [48]. These applications use specialised camera payloads to constantly monitor or film ground activity. In order to provide continuous surveillance, multiple UAVs can to be deployed that rotate at the end of one full battery cycle. To maintain this kind of system manually would be impractical. Instead, an autonomous recharging strategy can be used to automate this process to ensure continuous surveillance. These recharging stations can also be used to extend the operation radius in a similar manner as discussed above.

These are just some examples where automated recharging can help automate any UAV application. If this technology were to be successfully implemented on a larger scale it could even unlock new UAV applications that seemed impractical before.

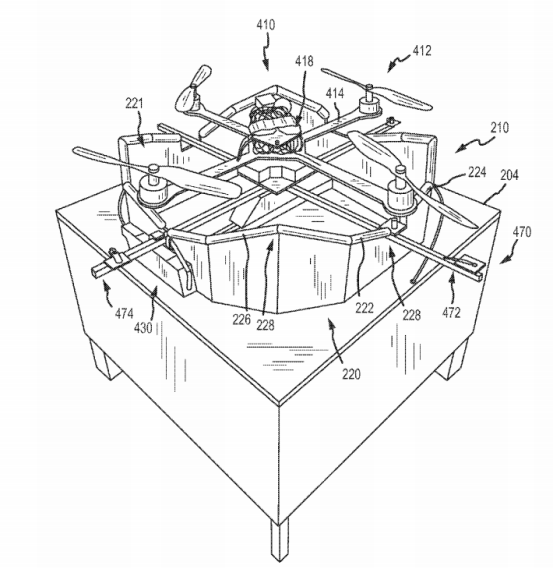
2.2 UAV Recharging

This section aims to explore current methods used to autonomously recharge UAVs. From the literature three main methods of autonomously recharging UAVs were identified, namely: direct contact based approach, wireless recharging and a battery swap approach.

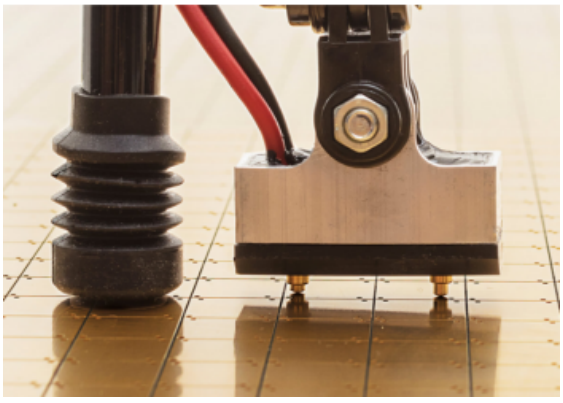
2.2.1 Direct Contact Based Charging Station

In [3] a recharging station is presented that makes use of electrical contact pads located on custom landing gear fitted to the UAV to make electrical contact with a docking station and is shown in Figure 2.1a. The UAV needs to make contact with the charging station without human intervention and thus the UAV is required to make an accurate autolanding prior to charging. The landing accuracy reported by this system is 7 cm. This inaccuracy was compensated for in the design of the docking mechanism which made use of slanted ramps for each arm of the UAV. The UAV would thus slide down into slots which kept the UAV locked in position using mechanically activated hanger arms.

In [5] a company called Skycharge demonstrates their version of a direct contact based recharge station. The station consists out of multiple electrical contact pads organized into a grid of positive and negative conductors. The UAV is fitted with custom landing gear containing two pointed electrodes used to make contact with the charging station in a similar fashion as in [4]. This solution is illustrated in Figure 2.1b. Landing inaccuracy is absorbed through the size of the charging station and the unique grid design which allows the UAV to land anywhere on the station. This is one of the most effective autonomous recharging solutions today and claims to have the ability to recharge a 10 000 mAh battery in one hour with a reported efficiency of 92%.



(a) Recharge station design [3]



(b) Skycharge recharge station design [5]

Figure 2.1: Direct contact based UAV recharging systems

In [4] a similar strategy is used by means of making direct electrical contact to the charging station, however the design of the contact mechanism was different. Each leg of the UAV is fitted with custom landing gear containing a pointed electrode and spring acting as a compression element thereby applying even pressure on each electrode. An autolanding

strategy was also used to autonomously place the UAV in the correct position prior to landing. This system relies on landing accuracy to ensure correct contact is made unlike in [3] where the design of the recharging station compensates for landing inaccuracy. Here the autonomous landing is performed using IR sensors located on the UAV and charging station to accurately localise the UAV during landing.

2.2.2 Wireless Recharging

Instead of making direct electrical contact with a charging surface this contact can also be made wirelessly using wireless power transfer (WPT). WPT is the transmission of electrical energy across an air gap. The energy is transferred via magnetic fields between two resonant circuits, one located on the charging surface and another on the UAV. Magnetic resonance induction has little interference or disturbance within its environment. Another advantage of WPT is that the UAV is not so heavily reliant on landing accuracy as the coils do not have to be perfectly aligned for power transfer, however to ensure the best possible efficiency the two resonant coils need to be placed as close together as possible and thus landing accuracy is still important in ensuring practical recharge times [7].

In [6] multiple coils are placed on the charging station which increases the size of the effective charging area. This method can be used to compensate for landing inaccuracy. This can greatly simplify the design of the charging station as well as the complexity of the autolanding system. Wireless charging however is less efficient than a direct contact based approach even with optimal coil placement. In [6] an efficiency of 75% was achieved with a UAV and wireless charging system.

Using a strongly coupled magnetic resonance induction form of WPT allows for high power transmission up to a distance of 2m with 40% efficiency [8]. Its use for UAV applications is motivated by its omnidirectional transmission capability which means that a UAV can land without needing to change its heading. These coils can be safely enclosed in a protective housing which will provide protection from outdoor factors such as rain or dust. Figure 2.2 provides a block diagram of the WPT system.

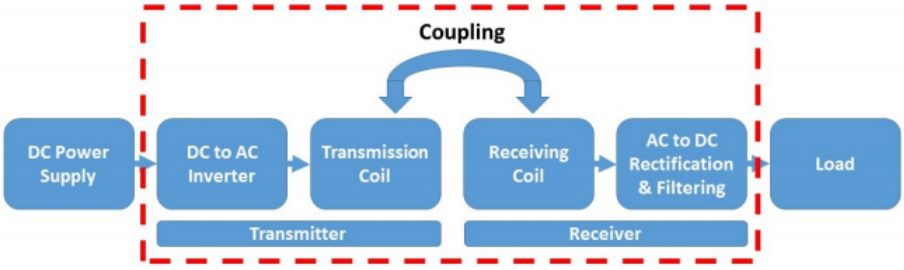
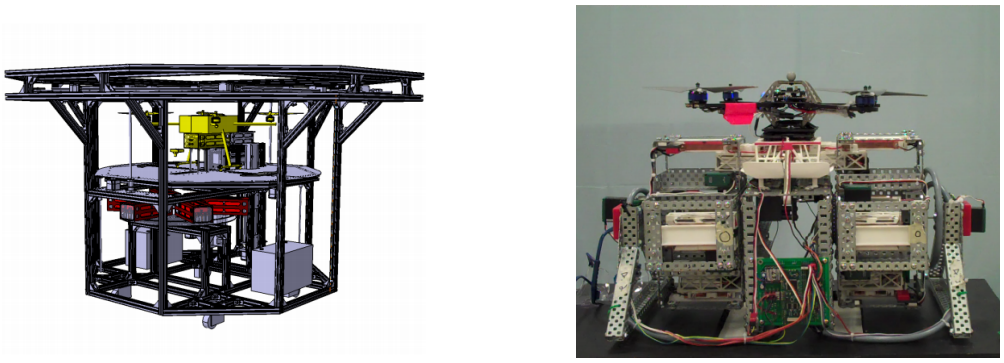


Figure 2.2: WPT system block diagram [6]

In [9] different wireless charging methods are presented. One proposed solution involves using the electromagnetic field present around power lines to wirelessly charge UAVs. Studies have been done which investigated the viability of doing this by estimating the amount of available energy around these power lines. It was determined that there would be sufficient energy to charge these UAVs provided the UAV had some method of landing on the power line [9].

2.2.3 Battery Swap

Battery swap technology for multirotor UAVs is one solution which aims to speed up the recharging process of UAVs. Battery swap solutions are complicated mechanical systems and are more expensive than the two previous solutions. Multiple batteries can be stored in charge which would work well in situations where multiple UAVs require recharging. A custom battery housings on the UAV is also required to interact with the swapping mechanism.



(a) Battery swap system from [10] (b) Battery swap system from [11]

Figure 2.3: Two different designs presenting a solution to automated battery swapping

In [10] and [11] two different battery swapping systems are presented. Both these systems rely on highly accurate autolandings with small margin for error. Figures 2.3a and 2.3b show the designs of both systems.

Table 2.1 presents a summarised comparison between the three different recharging systems discussed in this chapter by assigning a score out of 3 for each category. 1 being good and 3 being bad.

Recharge Method	Recharge Time	Simplicity	Outdoor Practicality	Efficiency	Required Landing Accuracy
Direct contact based	2	1	2	1	2
WPT	3	2	1	2	1
Battery swap	1	3	2	1	3

Table 2.1: Comparing three different recharging systems for multi-rotor UAVs

2.3 Multi-Rotor UAV Autolanding

An autonomous takeoff and landing system is required as part of the autonomous recharging process. These landings require high levels of accuracy in order for the UAV to be in the best possible position for optimal charging. Precise and accurate landings are also essential in achieving robustness. Traditional GNSS used on almost all UAVs cannot be used to localise the UAV during this landing process as this measurement is vulnerable to drift and has an average accuracy of 1.8 m [12]. Other methods of accurately localising the UAV for the purpose of performing accurate autolandings are thus discussed in this section.

2.3.1 Sensor Configuration For Automated Landings

In order to track the stationary landing target the UAV's flight control system requires knowledge about the position and attitude states of both the UAV and the landing target. This subsection thus focuses on different sensor configurations that can be used for the purpose of determining these states.

From the literature several vision based autonomous landing systems were identified. In each system a camera is used to identify a known landmark or feature located on the landing target. From these known features the relative position and attitude states of the UAV and the landing target can be determined. Each system uses different visual designs and camera configurations to track the landing location. Some of these visual systems are investigated more closely.

Early investigations in achieving autonomous vision based landings used the letter 'H' as a visual landmark to localise the UAV. This is so that existing helipads could be used for autonomous landings. In [14] this was attempted with a helicopter and a monocular camera. An average landing accuracy of 40 cm was achieved. This error is small relative to the size of the letter 'H' used on helipads but is not accurate enough for this project. The cause for this inaccuracy was reportedly due to several reasons. Firstly the camera used was fixed mounted to the underside of the vehicle, thus from the camera's perspective small inclination changes of the vehicle resulted in large movements of the target on the ground. This makes tracking an object difficult. Secondly the letter 'H' is large which given the limited field of view of the onboard camera, meant that at a certain altitude the marker is no longer fully visible and thus cannot be used to accurately localise the vehicle. Lastly, precise control of the vehicle at low altitude is difficult due to the effect of a high pressure air-cushion called ground effect which is caused by the downward thrust from the main rotor of the helicopter. These problems need to be considered when designing a marker for this project.

In [13] The landing of an autonomous helicopter was successfully achieved using a monocular camera and a landing marker consisting out of concentric black and white squares. A robust detection algorithm was developed that could identify and estimate the pose of the marker in real-time. The onboard GNSS was used to first pilot the UAV to the known location of the marker before using the onboard cameras to perform the landing. Here, the problem created by the limited field of view of the camera encountered by S. Saripalli in [14], is eliminated as the UAV's pose can be estimated from the small inner most concentric square at low altitude. This type of design can thus be used to help ensure landing accuracy.

In [17] and [18] markers consisting of a pattern made up of several Infrared (IR) light emitting diodes (LEDs) are used as the landing marker. In both cases a Wii remote IR camera is used to detect the incoming light. This solution however is only effective at night as normal sunlight prevents the detection of the IR LEDs. Other visual markers can however only be detected by a normal camera during the day and thus a combination of both solutions can be used to extend landing ability in both day and night conditions.

In [19], [20], [21] and [22] ArUco markers are used as artificial landing markers and are detected using a low cost monocular camera. ArUco markers are made up of an outer black border and inner binary matrix. The binary matrix is a pattern containing black and white squares which can be decoded as a unique ID given to each marker, similar to quick response (QR) codes. These markers are often used in augmented reality and robotics applications to determine the pose of a camera. The ArUco detection module has been incorporated into OpenCV. This module offers robust detection due to the markers unique inner binary matrix

codification which allows for continuous error detection and correction [23]. They offer any camera the ability to easily detect and accurately estimate the relative 6 degree-of-freedom (DoF) pose of the marker. This ability makes it well suited for the purpose of performing accurate vision based autolandings.

B. Pervan et al [24] proposed using a Carrier-Phase Differential Global Position System (DGPS) system for automated landings of aircraft on ships in zero visibility conditions. The DGPS system provides position measurements with centimetre level accuracy. In [25] the first ever successful landing of a Northrop Grumman X-47B UAV on an aircraft carrier was achieved using a similar system. In [61] a differential GNSS system was used to perform an accurate autolanding of a quadrotor UAV on a translating platform. In such a system a sensor is placed on the UAV as well as the charging station. Each new charging station would require a new DGPS system. This technology is still quite expensive and thus this is not a cost effective solution.

2.4 Localisation

Some UAV applications require large degrees of position measurement accuracy. Traditional GNSS cannot be used as the only form of inertial position measurement in such scenarios due to its inaccuracy. This project aims to explore other methods of increasing the localisation accuracy of a UAV. In this project highly recognizable landing markers are placed within the UAV's operating environment. This section therefore investigates methods by which these landmarks can be used to improve the localisation accuracy of the UAV, however other methods are also discussed.

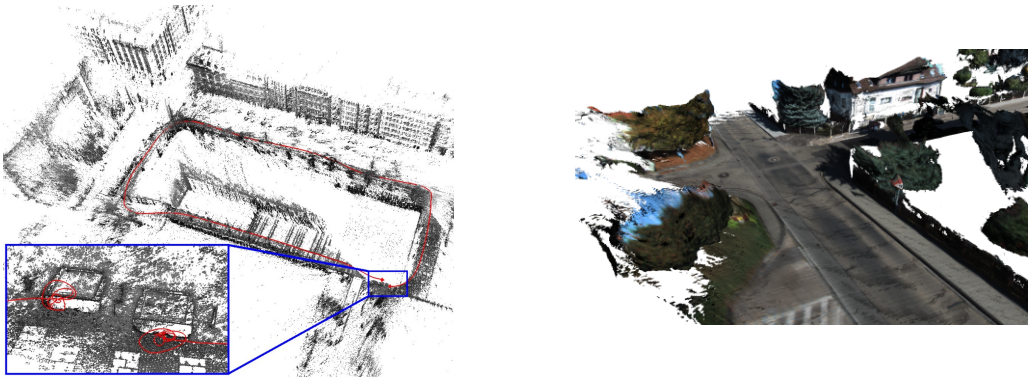
2.4.1 Visual Odometry

Odometry refers to the use of data or measurements from a vehicles actuators to estimate the change in position of the vehicle [44]. For ground based vehicles encoders on the vehicles wheels are used to measure the change in position. On UAVs however, these sensors are not available. visual odometry (VO) refers to a system that uses cameras to estimate the distance travelled by comparing each sequential image to either a previous image or to a known map of the environment.

VO can be subcategorized into either feature-based VO or direct based VO. In [26] a feature based method is used which involves extracting feature points (lines, corners etc) and tracking their movement in each successive frame. Direct methods use each pixels intensity as input and localises the camera by minimising the pixel intensity error [27].

When the UAV needs to be localised within an unknown inertial environment then a map of the environment needs to be generated while simultaneously localising the UAV within the mapped environment using a method called Simultaneous Localisation and Mapping (SLAM). A SLAM map can either be "sparse" or "dense". The main difference is the amount of information or points used to describe the environment. Dense mapping is computationally more expensive and requires specialised hardware such powerful graphic processing hardware and a special type of camera known as RGB-D which contains an additional depth channel. This channel describes each pixels relative distance between the image plane and the observed object. This allows for highly detailed mapping with extra computational cost [32]. This kind of mapping is not suitable for this project due to the UAVs constraint hardware. An example of such a map is shown in Figure 2.4b.

Sparse mapping, shown in Figure 2.4a, relies on a limited number of interest points used to construct the map and can be done using a low cost monocular camera. These points mainly consist out of distinct features such as corners or lines. The reduced number of interest points makes this approach more computationally efficient compared to a dense map solution.



(a) Sparse Mapping [28] (b) Dense Mapping [29]

Figure 2.4: Showing difference between sparse and dense mapping

ORB-SLAM is a monocular SLAM solution that makes use of a sparse map to compute the cameras pose and creates a 3D reconstruction of the environment. This method has become popular for use on small mobile robots and UAVs as it is not as computationally intensive as other SLAM algorithms, but does however still rely on powerful graphics processing hardware. ORB-SLAM is robust as it is able to operate in environments with severe motion clutter and can perform large loop closure. ORB-SLAM achieves unprecedented performance when compared to other existing monocular SLAM approaches using the most popular datasets [33].

ORB-SLAM2 is an improvement to the algorithms used in ORB-SLAM and provides more robust loop closure. ORB-SLAM2 also includes the use of RGB-D and stereo cameras which allow for more accurate mapping. ORB-SLAM2 is a real-time visual SLAM solution.

2.4.2 Marker based SLAM

In this project charging stations are placed in the UAV's operating environment. These stations can act as visual landmarks which can be accurately mapped to the inertial frame. Marker based SLAM involves using a monocular camera to map artificial markers to a global frame while simultaneously estimating the relative pose of the marker and the camera. Previously mentioned "sparse" or "dense" SLAM methods have certain limitations which a marker based approach does not. Firstly these methods require an environment with texture in order to gather feature information. Secondly, relocalisation is difficult in some scenarios with repetitive looking environments. Thirdly, translational movement of the camera is required if only a single camera is used in order to gather depth information and lastly these methods are computationally expensive [45]. Marker based SLAM methods do not have these constrains but do have their own unique limitations.

In [35] IR beacons were used as landmarks. A marker consisting out of multiple IR beacons was detected using a small low cost IR sensor which allows for the full 6 DoF pose of the UAV to be estimated. This approach is however not very effective in outdoor environments

as sunlight creates a lot of interference.

In [34] a landmark-based localisation method is presented where the position of a multirotor UAV relative to a point of origin is calculated using an onboard monocular camera and a known artificial marker placed on the ground. In [36] multiple artificial markers were placed on the ground which were detected using a monocular camera. From these markers the pose of the UAV was estimated in a fully GNSS denied indoor environment. In [40] artificial markers consisting out of a checker board pattern and black circles was used to accurately localise a UAV in a SLAM application. In these projects the design of the markers were arbitrary and worked well in controlled indoor environments with no background variation. The robustness to false positive detections in a real world outdoor scenario with lots of background variation is unknown. Other marker designs proven to be robust in this regard would potentially be better suited for this project.

In [37] squared planar markers (SPMs) were used in a SLAM application with a monocular camera. SPMs consist out of a black border with an internal code (most often binary) [45]. This is the same design ArUco markers are based on. A map of each markers inertial frame location was created by observing pairs of markers while simultaneously localising the camera using pose estimates from already mapped markers. Using SPMs such as ArUco markers are beneficial as they already allow for accurate and robust pose estimation of a camera in any environment. This is an attractive approach for this project as good results have been achieved using ArUco markers as landing targets [19], [20], [21] and [22]. This would simplify the design of the recharge station and would only require a single onboard camera as the primary sensor for localisation thereby making this approach cost effective and computationally efficient.

2.5 Project Software

In order to complete this project software packages are required for: simulation, flight control and robotics. A realistic simulation environment is required to test the proposed solution and UAV system before attempting practical experiments. The simulation software must be capable of thoroughly testing the flight control software in both a SITL and HITL testing setup. The flight control software is mainly responsible for executing the controllers of the UAV but also communicates with ground station software to monitor the UAV systems during a flight. Finally, software used for the robotics aspect of the project is required. This component is mainly responsible for processing external sensor data and relaying the output to the flight control software where it is used by controllers to execute the required manoeuvre. Each software component is discussed in the following subsections.

2.5.1 Avionics

The avionics of the UAV refer specifically to the physical flight controller and flight control software.

DJI is a well known company that mainly develops consumer multirotors. Most of their UAVs have gimbal actuated high resolution onboard cameras. DJI does allow for onboard SDK development for onboard computing and now supports integration with ROS [38]. These products are expensive and DJI's software is still mostly closed source which adds extra complexity to the development cycle.

Open source flight control software packages are also available. These software packages are well documented and regular improvements are made to add new features and increase stability. Examples of open source flight control stacks include: PX4, ArduPilot, Betaflight and iNav to name a few [41].

Betaflight is mostly used for hobbyist quadrotor racers and does not allow for complex autonomous UAV development [56]. iNav was developed from Betaflight and runs on the same hardware but mainly specialises in autonomous waypoint navigation. Currently iNav does not provide the same level of support or level of customizability as PX4 or ArduPilot which makes it better suited for hobbyists [41].

The Electronic Systems Laberatory (ESL) at Stellenbosch University has made a strong move towards using Pixhawk flight controllers. These flight controllers are popular with industry and has a large development community. Pixhawk provides an open-source hardware design for flight controllers running the NuttX real time operating system (RTOS). The ESL has become confident in using these flight controllers in recent years as a number of projects have been successfully completed using this flight controller architecture [39], [40] and [56].

The two remaining flight control software stacks capable of running on Pixhawk hardware is PX4 and ArduPilot shown in Figure 2.5.

Figure 2.5: Open source flight control software for multirotor UAVs

ArduPilot only supports SITL simulations and not HITL simulations unlike PX4 which supports both. Both kinds of simulations will be required for this project as this gives more confidence in simulation results. PX4 also provides simpler development due to its asynchronous software structure. This handles time scheduling automatically where as ArduPilot's time scheduling between its various software components needs to be tweaked manually for optimal performance [56]. PX4 has also been used in a number of projects in the ESL and has been adopted as the preferred software stack [39], [40] and [56].

2.5.2 Simulation Software

PX4 and ArduPilot work well with simulators such as Gazebo, Airsim and JMAVSim. Gazebo and Airsim provide developers with more customizability over the simulation environment compared to JMAVsim. JMAVsim also does not allow for external sensors such as cameras thus making it unsuitable for this project.

Airsim is a plugin built for Unreal Engine, this allows for highly realistic looking environments to be created using the Unreal 4 gaming engine. This is mainly used for visual based navigation applications where realistic looking environments are essential. Both Gazebo and Airsim are built on powerful physics engines which provide highly realistic simulations matching the real world. Gazebo however is run on two separate servers. One is used for visualisation and the other is used for the physics engine. These can be run independently in a number of configurations thereby giving more flexibility on systems with limit graphics hardware. Gazebo also has a more lightweight graphics engine thus making it better suited to these

kinds of systems. The performance of an Airsim simulation is dependant on the environment being simulated which can either be simplistic or detailed. The user thus has some control over this but requires knowledge of how to optimise these environments in Unreal.

Airsim does not have Linux support. The Robotic operating system (ROS) however is only supported on Linux which makes integrating these systems challenging. The physics engines used by Gazebo and Airsim are similar and thus neither are advantageous in this regard. Gazebo has a larger community and is more widely used in the ESL which aids in simplifying the development process.

2.5.3 Robotics Software

When image processing is required in a UAV application a separate piece of hardware called an onboard computer (OBC) is used as a dedicated image processing unit [42]. Processed image data is published to the flight controller where it is used by controllers. Robotics software is installed on the OBC that is responsible for handling incoming images and also provides a communication structure between the OBC and the flight controller. ROS is officially supported by PX4 and Ardupilot and is very popular in the world of autonomous robotics.

ROS is not an operating system as the name suggests but is instead a collection of standardised packages and tools used for developing robotic applications [43]. ROS uses hardware abstraction to provide support for a wide variety of sensors with are commonly used for autonomous robotics and provides standardised ways of using and visualising sensor information. ROS, like PX4 also uses a publish-subscribe architecture and is at its core an effective communication system.

One of the advantages of ROS, is its large community that actively develops new libraries and packages. ROS currently has multiple versions or distributions, these can be mainly classified as either belonging to ROS1 or ROS2. ROS2 is the latest improvement to the ROS system and is currently being accepted as the industry standard due to its improved real time factor and more robust non-centralised system. At the time of this project PX4 only worked with ROS1. ROS Melodic which is a ROS1 distribution is used for this project.

A ROS node is a basic executable that can either be written in numerous languages including C or Python. A ROS node is usually dedicated to a single task in a ROS system. Software belonging to ROS is organized into a ROS package. A ROS package usually consists out of multiple ROS nodes which communicate using the publish-subscribe system. Together they are used to provide a solution to a specific problem or application. In ROS1 a master node exists which coordinates communication between the different nodes. In ROS2 the master node was removed in order to safeguard the system against a single point of failure.

2.6 Summary

In this chapter a wide range of solutions found in literature aimed at achieving autonomous recharging of a quadrotor UAV was investigated. Different methods used to charge UAVs was firstly identified. Three possible approaches were discussed, namely: direct contact using electric conductors, wireless recharging using WPT and battery swap. While a battery swap approach is the most complicated it offers the fastest recharge time. Direct contact based approach offers the second fastest recharge time with reduced complexity, however wireless recharging is the best suited to an outdoor environment but recharge times are much slower

due to the decreased efficiency and limited current transfer. All these solutions require the UAV to make an accurate autonomous landing prior to recharging and thus this was investigated further.

Three different sensors apart from GNSS used to localise the UAV during an autolanding was identified from literature, namely: monocular camera, IR camera and DGPS. Artificial markers placed on the landing target can be detected using a monocular camera. From these markers the 6DoF pose of the camera can be determined at a high frame rate. Other methods used IR beams and cameras to localise the UAV during landing, however this is not suitable for outdoor daylight conditions as sunlight creates interference. DGPS has also been used in applications where a UAV needs to land on a moving platform in zero visibility conditions. This technology provides centimetre level accuracy but is very expensive which makes scaling autonomous recharging impractical. Monocular cameras are small and inexpensive making them the ideal choice for this project.

Using the highly recognizable charging stations as visual landmarks in a marker based SLAM approach was investigated next. It was determined that a lot of research has been done using squared planer markers (SPM) such as ArUco markers in marker based SLAM applications. Other forms of markers have also been used but only in indoor controlled environments. Their performance in cluttered environments is unknown however ArUco marker detection is robust in any environment which makes this an attractive choice for this project. Other methods such as sparse and dense SLAM was also investigated.

Finally different software packages required for this project were investigated. It was determined that PX4, Gazebo simulator and the robotic operating system (ROS) is the most suitable software toolchain for this project.

Chapter 3

System Overview and Design

In this chapter, the main UAV related hardware and software components are discussed. A custom UAV had to be commissioned for this project. This chapter discusses the design of the UAV's various components as well as the process by which the UAV's physical parameters were obtained for simulation and controller design. This chapter discusses the software toolchain used for simulation and practical flights. The toolchain is comprised of PX4 autopilot flight control software, Gazebo simulator and ROS, which is used to add custom autopilot functionality.

3.1 Multirotor Overview

A quadrotor UAV is a kind of multirotor aerial vehicle which is propelled by four fixed rotors placed at equal distances from the centre of the vehicle. A quadrotor is able to hover in level flight by having each rotor rotate at equal speed. Increasing the speed of each rotor equally such that the resulting net force overcomes the force of gravity will allow the UAV to increases altitude. By decreasing the speed in a similar manner the UAV's altitude can be lowered. Each motor induces a torque on the UAV airframe which contributes to a moment around the UAV's z axis. These torques are balanced out by making adjacent rotor pairs rotate in opposite directions as indicated in Figure 3.1 [2].

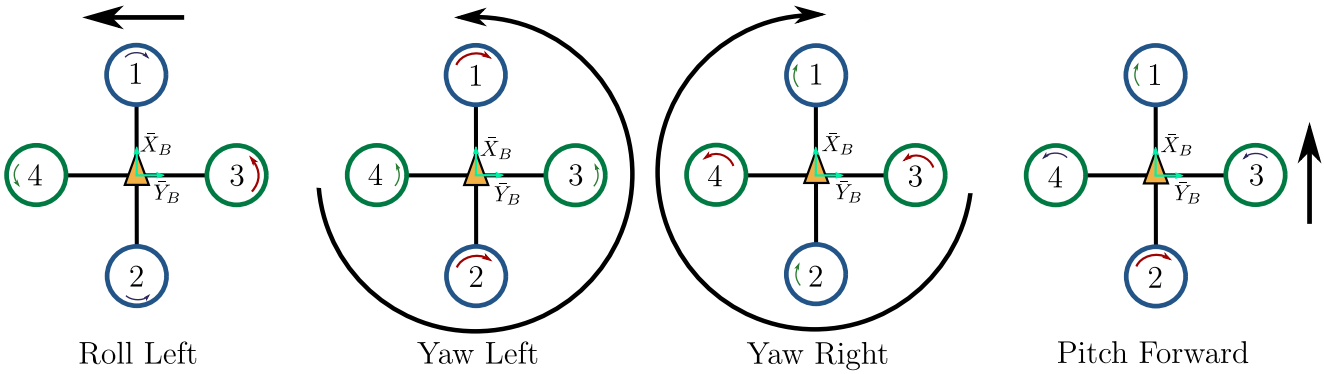


Figure 3.1: Illustrating rotor speed change required to perform basic manoeuvrers

Figure 3.1 demonstrates the rotor actions used to perform basic pitch, roll and yaw manoeuvrers. To pitch forward the UAV increases the speed of rotor 2 and equally decreases the speed of rotor 1. To roll left the UAV increases the rotational speed of rotor 3 and decreases the speed of rotor 4 by the same amount. A yaw manoeuvrer is performed by increasing and decreasing the rotational speed of an opposing rotor pair as shown in Figure 3.1.

3.2 Hardware Design

The UAV will be required to perform vision based autolandings using a small onboard camera. The UAV will be required to make contact with a charging surface using custom 3D printed legs added to the UAV frame, the UAV thus needs to be modular allowing for custom components to be added. It must also be able to produce enough lift to have the required amount of manoeuvrability needed to perform manoeuvrers during flight. The UAV needs to maximise flight time in order to provide enough opportunity to carry out experiments.

3.2.1 Quadrotor Components

Figure 3.2 shows the different hardware components used to build the UAV for this project.

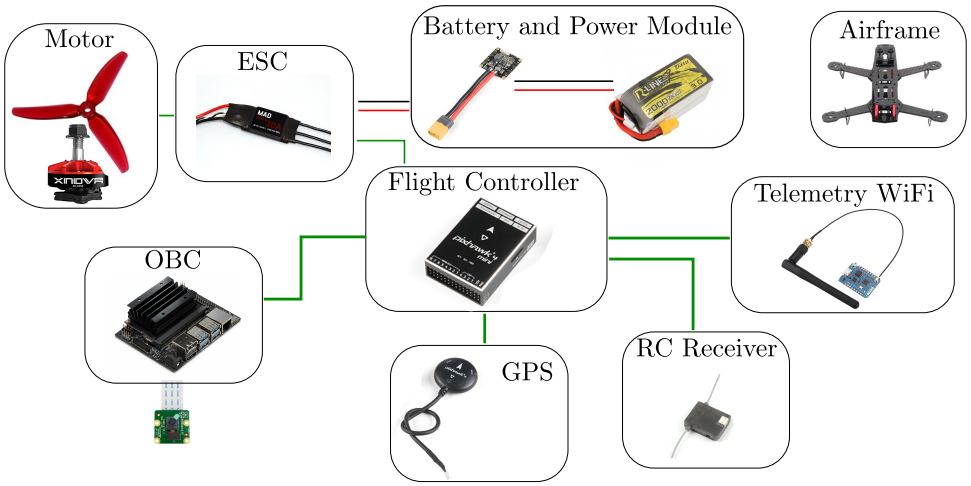


Figure 3.2: Components used to construct the quadrotor UAV

Table 3.1 below provides a summary of all the hardware components required to create the UAV. For this project it was decided to use a small, lightweight, inexpensive and modular first person view (FPV) racing drone airframe. These UAVs have impressive power to weight ratios which allows for a medium sized payload to be added to their lightweight and durable carbon fibre frames. There are many airframe options available, for this project a large FPV airframe was used allowing for enough area to add custom components. FPV racing frames do not come with landing gear so a custom 3D printed design was fitted to the UAV.

The Jetson Nano from Nvidia was used as the OBC. It contains a 128-core NVIDIA Maxwell™ architecture-based graphics processing unit (GPU). This is perfect for real-time image processing applications. The ROS version that is run on the OBC requires an Ubuntu operating system. It is however possible to flash Ubuntu 18.04 on the Jetson Nano, making it a suitable companion computer. A Raspberry Pi camera V2 connected to the OBC is used as the onboard camera. This camera is capable of streaming high resolution images at 30 frames per second. This camera is lightweight and has a small form factor making it an ideal choice for this system.

A number of flight controllers using Pixhawk's architecture is available. The most popular is the latest Pixhawk 4 flight controller from Holybro. Holybro also makes a smaller version of the Pixhawk 4 called the Pixhawk 4 mini. This flight controller was intended to be used on smaller vehicles such as the one used for this project. Included with this flight controller is a GNSS, magnetometer, IMU and power module. This flight controller supports airframes with

up to 8 motors and also has one MAVLink telemetry port which is either used for connecting Wi-Fi/telemetry radio or an OBC.

Component	Description
Flight Controller	The flight controller is responsible for: executing controllers, combine all onboard sensors and remote control (RC) controller, output actuator control signals to ESCs driving motors, log all flight data and communicate with a ground station.
IMU	Situated on the flight controller, the inertial measurement unit (IMU) contains the gyroscope, accelerometer and barometer which measure the angular rates, acceleration and air pressure needed for the UAV's attitude and altitude state estimation.
GNSS	The GNSS measures the global position and velocity of the UAV.
Magnetometer	Located on the same hardware component as the GNSS, the magnetometer measures the current heading of the UAV. Vulnerable to Electromagnetic interference, the magnetometer is placed far away from the ESCs and motors of the UAV.
Barometer	The barometer measures the current air pressure. This is used to estimate the current altitude of the UAV.
Telemetry Wi-Fi	This provides a wireless connection to a ground station used to monitor all the UAV systems.
RC Receiver	Communicates with the radio transmitter used to manually pilot the UAV.
ESC	Four electronic speed controllers (ESCs) are used to control the rotational speed of each individual motor based on the incoming pulse width modulation (PWM) value sent by controllers running on the flight controller.
LiPo Battery	A 4S (4 cell) lithium polymer (LiPo) battery is used to drive the motors and provide power to all onboard electrical systems.
Power Module	Converts battery voltage to stable voltage used to power all onboard electrical components as well as motors. The power module also contains current and voltage sensors to monitor the battery capacity.
OBC	The OBC is used to process onboard camera images and acts as the autopilot commanding the UAV.

Table 3.1: Quadrotor component description

Propulsion System Design

The propulsion system refers to the battery, motors and propellers used by the UAV to generate lift. Many different kinds of motors and batteries are available however the correct components need to be chosen to match the requirements of this project. Maximizing flight time is the most important requirement. The chosen motors need to supply enough thrust for sufficient payload carrying capacity and to preserve manoeuvrability but need to be conservative in power consumption. The largest battery available that could realistically fit on the UAV had a capacity of 2000 mAh. The task is to find an appropriate motor that would best utilize this limited capacity. To do this, the total mass of all the UAV's components were summarised first in Table 3.2.

Component	Mass (kg)	Quantity	Total Mass (kg)
Frame	0.155	1	0.155
Battery	0.2	1	0.2
Flight Controller	0.0372	1	0.0372
GNSS	0.032	1	0.032
Power Module	0.007	1	0.007
Companion Computer	0.141	1	0.141
Camera	0.003	1	0.003
DC-DC Converter	0.004	1	0.004
Legs	0.006	4	0.024
Motor	0.029	4	0.116
ESC	0.009	4	0.039
		Total Mass	**0.8092**

Table 3.2: UAV mass summary

The optimal motor needs to provide enough thrust to achieve a power to weight ratio of at least 2:1 but still has low current draw to maximise flight time. A list of available motors is provided in Table 3.3. All the motors listed are similar in size and mass. They differ in their stator width and the number of copper windings on the stator. These differences are small enough for the resulting mass difference to be neglected. These differences do however have an impact on the amount of current drawn by each motor and the total thrust generated. This has a direct impact on the total flight time. All data presented in Table 3.3 is provided by the manufacturer [54].

Motor	Voltage (V)	Throttle %	Current (A)	Thrust (kg)	Payload at 50% thrust (kg)	Flight Time (min)
Xnova Lightning V2N 2207	14.8	50	22.7	0.790	3.16	3.3
		100	50.7	1.548		
T-motor F40Pro2	14.8	50	13.2	0.780	3.12	6.23
		100	49.62	1.599		
Xnova hypersonic RM2205	14.8	50	7.8	0.412	1.648	11.18
		100	34.7	1.240		
Xnova RM2204	14.8	50	6.8	0.400	1.6	12.76
		100	16.2	0.786		

Table 3.3: UAV motor summary

All the motors summarised have a power to weight ratio better than the desired value of 2:1. This will allow the UAV to easily perform any required manoeuvre. The best motor for this project is the motor with the lowest current draw during hover. It is clear to see that at 50 % thrust, the UAV is beyond the required thrust needed for hover (0.8042 kg). The manufacturer does not provide data for throttle values less than 50 %. It would therefore be required to extrapolate the thrust and current measurements against throttle. The thrust versus throttle curve can be approximated as a linear function for the lower half of the curve. This assumption is only true for smaller motors such as these. Looking at the Xnova RM2204 it can then be said using this linear approximation that the UAV will be at hover at around 25 % throttle. Using the manufacturers supplied current values, the current versus throttle % curve can be extrapolated. A second order polynomial curve was fitted to the data. From this curve it was determined that the current draw at 25 % throttle was 2.1A per motor. The total flight time can thus be calculated as follows

$$t_{max} = \frac{Batt_{cap}}{4 \times C_m + C_a} \times 60 = \frac{120}{4 \times 2.1 + 1} = 12.76. \tag{3.1}$$

t_{max} is the maximum flight time of the UAV, $Batt_{cap}$ refers to the maximum battery capacity of the UAV measured in Ah and C_m and C_a represents the current of the motor and the

current drawn by the rest of the avionics respectively. The same method is used to calculate the total flight time of the other motors. The Xnova RM2204 is chosen as the motor for this project due its flight time performance. With this motor the UAV has more than enough thrust needed to carry all onboard components and with 12.76 minutes flight time there is enough opportunity to do any required flight test. Figure 3.3 shows the assembled quadrotor used for the project.

Figure 3.3: The custom quadrotor used for this project (Honeybee)

3.3 System Identification

In order to create a model of the UAV for controller design and simulation testing, the physical parameters of the UAV need to be measured and calculated. These parameters include: mass, motor arm length, mass moment of inertia, motor time constant, virtual yaw moment arm and thrust profile of each motor. The actual mass of the vehicle was measured as 0.808 kg. This is close to the estimated mass in Table 3.2. The motor arm length is defined from the centre of the UAV to each motor. In this case it was measured to be 0.11 m. All other parameters had to be calculated either experimentally or mathematically.

3.3.1 Thrust Profile

For simulation purposes an accurate thrust model of the motors is needed. The exact relationship between the PWM signal published by the controllers and the corresponding force acting on the UAV needs to be determined. This thrust profile cannot be approximated as completely linear for accurate simulations. A thrust test jig comprising out of a load cell was used to determine this relationship. Ideally the thrust profile of each individual motor needs to be determined. However each motor on its own does not generate enough thrust for the test jig to measure it accurately. For this reason the entire UAV was placed on the test jig. The generated thrust profile is then divided by 4 to obtain a thrust profile of each individual motor. In reality each motor does have a slightly different thrust profile due to non-idealities, however this is an acceptable approximation.

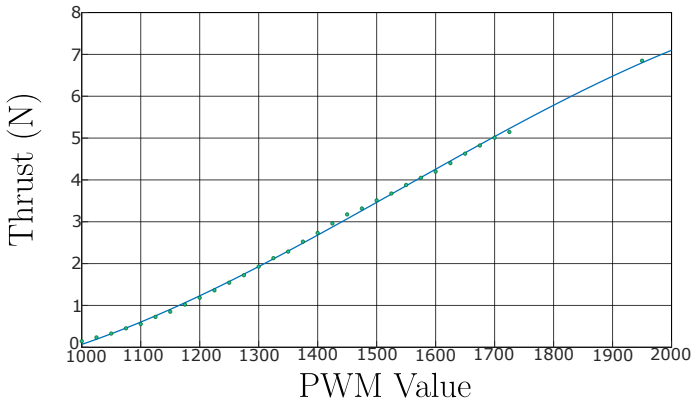


Figure 3.4: PWM to thrust relationship for each motor

Figure 3.4 shows the measured thrust outputs at each corresponding PWM value for each individual motor. A third-order polynomial was fitted to the measurements resulting in the above thrust curve. The maximum thrust was measured in a second test to complete the curve. The following function describes the relationship between the input PWM signal and thrust output T_m.

$$T_m(x) = -3.508 \cdot 10^{-9}x^3 \; + \; 1.627 \cdot 10^{-5}x^2 \; - \; 0.0172x \; + \; 4.528 \tag{3.2}$$

Where x represents the pulse width of the PWM signal. As can be seen from Figure 3.4 the linear approximation for hover throttle used earlier was reasonable and the estimated flight time should be accurate.

3.3.2 Motor Time Constant

The motor time constant refers to the time delay between the motor receiving a PWM signal representing a RPM value and when the motor reaches 67% of this desired RPM value. The time constant is dependant on the motor size, input voltage, input step size, current draw and the propellers used. Larger propellers will result in a larger time constant. To measure the time constant a large step input was given to the motors while the UAV was attached to the thrust test jig. The large step input would result in a larger time constant for a worst case estimate and make it easier to measure. During the test it was not possible to accurately determine the time constant for two reasons. Firstly, the measurements were noisy. This made it very difficult to determine the actual measurement over this small time period. Secondly the thrust test jig samples at 20ms. It was determined that the time constant was faster than the sampling period of the test jig from the measured output. This is expected for the type of motor and propellers used, time constant was thus conservatively chosen as 15 ms.

3.3.3 Mass Moment of Inertia

Two methods were used to calculate the mass moment of inertia. The results of both methods are compared to verify the answers. The first method uses a mathematical approach to try and estimate the inertia on each axis of the UAV. This method would be less accurate because of the approximations made about the UAV's shape. The second method follows a more accurate experimental approach to calculate the inertia of the whole vehicle.

Mathematical approach

It would be very difficult to calculate the moment of inertia of all the individual parts of the UAV. Instead the entire UAV is simplified and approximated as a rectangle with the same dimensions as the physical UAV.

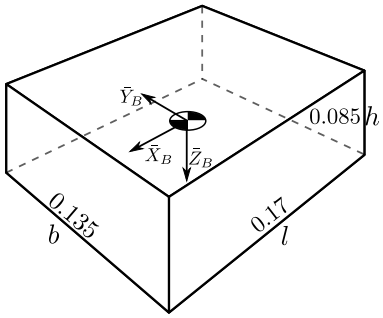


Figure 3.5: Rectangle approximation of the physical UAV

The mass moment of inertia of a rectangle can be expressed using:

$$I_{xx} = \frac{1}{12} \times m(b^2 + h^2), \quad I_{yy} = \frac{1}{12} \times m(l^2 + h^2), \quad I_{zz} = \frac{1}{12} \times m(b^2 + l^2). \tag{3.3}$$

From this the mass moment of inertia of the UAV is calculated as shown in Table 3.4:

Inertia	**I (Kgm^2)**
I_{xx}	1.71e-3
I_{yy}	2.43e-3
I_{zz}	3.17e-3

Table 3.4: UAV mass moment of inertia

Experimental Approach

The mass moment of inertia can also be determined experimentally using the two rope inertia experiment. This experiment is based off of work done by [55].

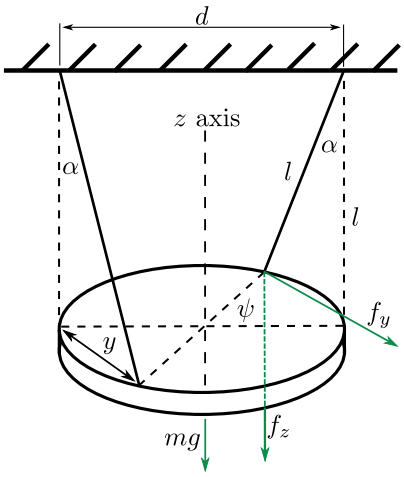


Figure 3.6: Two rope experiment model

The experiment uses a disk with unknown inertia suspended by two ropes as shown in Figure 3.6. If the disk is let go with an initial twist around its z axis then the disk will begin to oscillate around the z axis until coming to rest due to friction. By measuring the period of this oscillation it is possible to calculate the moment of inertia of the object. Making use of Figure 3.6. The inertia of the suspended object can be derived using the following equations:

$$\begin{aligned} \tan(\alpha) &= \frac{f_y}{f_z}, \\ &= \frac{f_y}{\frac{1}{2}mg}, \\ \therefore f_y &= \frac{mg}{2}\tan(\alpha), \text{ and} \end{aligned} \tag{3.4}$$

$$y = \frac{d}{2}\sin(\psi). \tag{3.5}$$

By applying the small angle theorem:

$$\begin{aligned} f_y &\approx \frac{mg}{2}\alpha, \text{ and} \\ y &\approx \frac{d}{2}\psi. \end{aligned} \tag{3.6}$$

α can be calculated using:

$$\begin{aligned} \sin(\alpha) &= \frac{y}{l}, \\ \alpha &\approx \frac{d}{2l}\psi. \end{aligned} \tag{3.7}$$

Substituting Equation 3.7 into 3.6, yields:

$$f_y = \frac{mgd}{4l}\psi. \tag{3.8}$$

The torque of the system can be expressed using Newton's second law of motion as:

$$\begin{aligned} T &= I\ddot{\psi}, \\ f_y d &= I\ddot{\psi}, \\ \therefore \ddot{\psi} &- \frac{mgd^2}{4Il}\psi = 0, \text{ and} \end{aligned} \tag{3.9}$$

$$\therefore w_n = \sqrt{\frac{mgd^2}{4Il}}. \tag{3.10}$$

w_n is the natural frequency of the system. From this the mass moment of inertia and the period of oscillations can be expressed as

$$\begin{aligned} I &= \frac{mgd^2t_p^2}{16\pi^2 l}, \text{and} \\ t_p &= \frac{2\pi}{w_n}. \end{aligned} \tag{3.11}$$

The UAV was suspended along all three of its axis in three separate experiments. In each experiment the distance d, length l and the period of oscillation t_p was measured. The results of the experiment are shown in Table 3.5.

inertia	d	l	time (s)	periods	t_p	I (Kgm^2)
I_{xx}	0.175	0.59	15	19.5	0.7692	2e-3
I_{yy}	0.135	0.59	15	17.5	0.8571	1.32e-3
I_{zz}	0.235	0.59	15	21	0.7142	3.35e-3

Table 3.5: Results of the two rope experiment and calculated mass moment of inertia

The experimental results are similar to the mathematical results apart from I_{yy}. This difference is likely caused by the battery which has a large contribution to the UAV's mass. The experimental results are however more accurate and thus they were used in the non-linear simulation.

3.3.4 Virtual Yaw Moment Arm

The virtual yaw moment arm is used to obtain a relationship between the thrust of a motor and the torque contribution of the motor. This relationship is linear and can be expressed as:

$$\tau = R_n T. \tag{3.12}$$

Here, τ is the torque produced by the motor, T is the thrust generated by the motor and R_n is the virtual yaw moment arm coefficient. This coefficient can be used to calculate the moment that is applied to the UAV. The method used to calculate the virtual yaw moment arm is called the Blade Element Theory. The virtual yaw moment arm coefficient is calculated under hover conditions. The hover thrust can be determined from the previous thrust test jig experiment. The rotational speed of the rotors were measured using a Tachometer. This method is based off of work done by [55] and [57], as discussed by [56].
The rotor torque and rotor thrust can be expressed as:

$$\tau = \rho A C_R (wR)^2 R, \text{ and} \tag{3.13}$$

$$T = \rho A C_T (wR)^2 \tag{3.14}$$

where ρ is the air density, A is the total area that the blades travel through, C_R represents the rotor torque coefficient and C_T represents the rotor thrust coefficient. w is the rotation speed of the rotor and R represents the radius of one propeller. The rotor power coefficients (C_P) relationship to the rotor thrust coefficient can be expressed as

$$C_P \approx C_R = \frac{k C_T^{\frac{3}{2}}}{\sqrt{2}} + \frac{\sigma C_d}{8}, \tag{3.15}$$

where k is a constant that describes the tip losses of the rotors. C_d describes the rotor blade drag coefficient and σ is the rotor solidity ratio. These values can be calculated using the following:

$$k = \frac{1.13}{B}, \tag{3.16}$$

$$B = 1 - \frac{\sqrt{2C_T}}{N}, \text{ and} \tag{3.17}$$

$$C_d = 0.4\alpha^2 - 0.0216\alpha + 0.0087, \tag{3.18}$$

where α represents the angle of attack of the rotor. This can be approximated as $\alpha = 0$ when the UAV is hovering in level flight. The rotor blade drag coefficient is calculated with

$$\sigma = \frac{N\bar{c}}{\pi R} \tag{3.19}$$

where N is the number of rotor blades per motor and $\bar{c}$ is the mean rotor chord length shown in Figure 3.7.

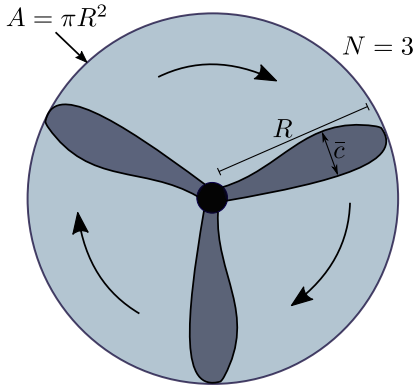


Figure 3.7: Rotor illustration

The measurements required to calculate the virtual yaw moment arm are shown in Table 3.6. By substituting these values into Equations 3.12 to 3.19, the virtual yaw moment arm coefficient R_N is calculated to be:

Variable	Measured Value
N	3
A	0.0121 m^2
$\bar{c}$	0.014 m
R	0.062 m
α	0
T	1.61 N
w	1235.69 rad/s
ρ	1.225 kg/m^3

Table 3.6: Measurements required to calculate virtual yaw moment arm coefficient

$$R_N = 7.997 \cdot 10^{-3}\ m. \tag{3.20}$$

3.4 Software Toolchain Overview

The following section covers the software toolchain that is used for this project. The toolchain consists out of PX4 flight control software, ROS, Gazebo simulator and QGroundControl ground station software.

3.4.1 PX4 Overview

PX4 flight control software is an open source flight control software that was developed for the Pixhawk flight controller boards. It is widely used in industry due to its convenient and

fast development cycle and open source aspect.

PX4 uses a publish-subscribe architecture. This allows for asynchronous message passing. It consists out of modules where each is responsible for handling a specific flight control function of the UAV, such as state estimation or flight control. Each module can publish messages containing information to a specific topic. Other modules can subscribe to the available topics and will receive published messages as soon as they become available. This type of communication architecture allows for highly modular software development.

It is possible to build PX4 on a Linux system such as Ubuntu. This allows for simulations where the actual flight software is being executed locally instead of on the flight hardware.

Estimator

PX4 uses an Extended Kalman Filter (EKF) for estimating the states of the vehicle. The EKF subscribes to sensor topics as well as optional external pose estimation topics to estimate the current vehicle states. By default the estimator only uses standard sensors as described in table 3.1. PX4 does allow developers to also fuse external pose estimation data from a SLAM or Motion Capture system for more accurate position estimation. Motion capture systems in this context rely on multiple external cameras to estimate the 3D pose of the UAV and is used sometimes used in indoor GNSS denied environments to localise the UAV. Chapter 8 contains further discussion on external pose estimation techniques.

Controllers

PX4's controllers use a cascaded loop architecture. The inner most controllers are responsible for controlling the attitude of the vehicle, i.e., the angle and angular rate of the UAV in the body coordinate frame. The outermost controllers are responsible for controlling the translation of the vehicle, i.e., position and velocity in the North-East-Down (NED) world or inertial coordinate frame. Chapter 5 discusses these controllers in more detail.

Navigator

The navigator is a PX4 module which is responsible for autonomously commanding the UAV. PX4 provides lots of autonomous functionality including autonomous waypoint scheduling, area surveying, autonomous take-off and landing, and more. The Navigator publishes set-points to controllers in order to command specific flight manoeuvrers. QGroundControl is a ground station software package that provides users with an interface to all Navigator functions. PX4 allows for an external system such as ROS to act as a Navigator module by subscribing to MAVROS topics while the UAV is in offboard mode. This allows for high level custom control of the UAV to be added.

3.4.2 Simulation

Gazebo simulator was chosen for this project as it has a lightweight yet realistic graphics engine which helps to guarantee efficient simulations with high real-time factors while providing accurate physics and large customization of the vehicle and environment. Gazebo supports both SITL and HITL simulations. SITL simulations work by executing the flight control software locally on the simulation PC, the generated actuator commands are converted to forces acting on the simulated UAV model. Gazebo publishes simulated sensor data back to PX4's estimator. SITL simulations allow for fast and reliable testing, however does not provide full guarantee that custom controllers will work on the actual flight hardware. To

verify this, HITL simulations are used after acceptable results are obtained with SITL simulations. In HITL mode the flight controller is connected to the simulation PC via a user datagram protocol (UDP) connection, the flight control software is then executed on the flight controller. This was required before any practical test to gain confidence in the system. Figure 3.8 illustrates both SITL and HITL simulations.

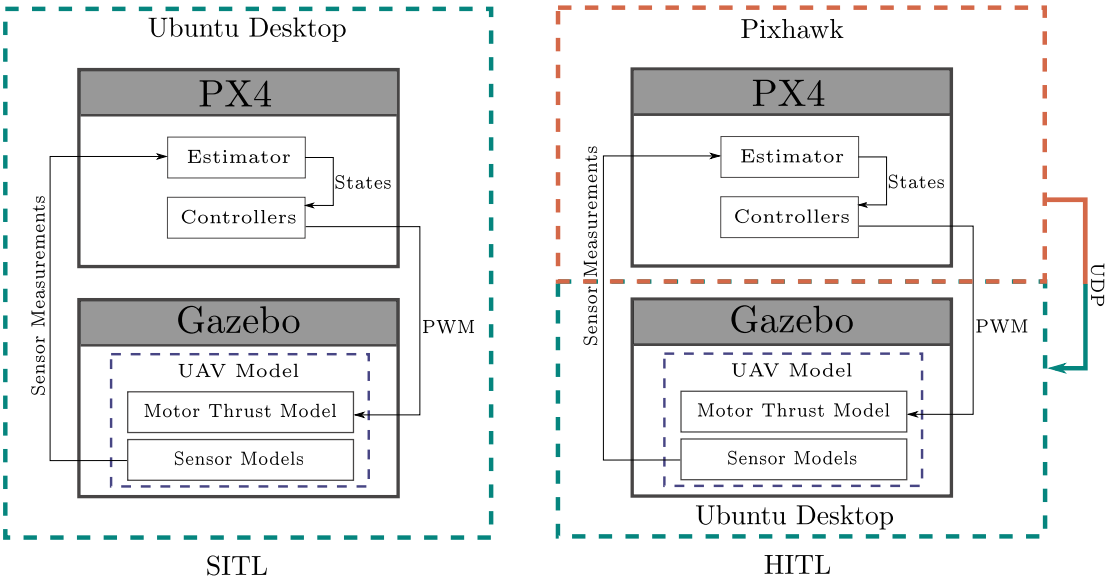


Figure 3.8: PX4, Gazebo simulation flow diagram

Gazebo uses a graphical modeling system. This means that instead of describing a simulated UAV model by its differential equations, it is instead described using a combination of links and joints. The physics engine of Gazebo then solves the differential equations related to the interaction of the links and joints making up the model. To do this each Gazebo robot model is defined using a simulation description format (SDF) file.

This file is used to specify the physical parameters of the UAV that were determined earlier. The SDF file also specifies what sensors need to be added to the UAV model. These sensors include: GNSS, IMU, magnetometer, barometer and camera. The exact orientation and position of the sensor relative to the UAV model can also be specified. These sensors are defined in Gazebo as plugins. The sensor plugins can simulate real world measurements by adding simulated sensor noise. The noise models include high frequency noise, random walk and sensor bias. The parameters defining the noise can be specified to match the specific application.

3.4.3 MATLAB/Simulink

A Simulink simulation was created based off of work done by [56]. It includes a non-linear quadrotor model and also contains a simplified version of PX4's flight controllers. The non-linear model uses the differential equations derived in Chapter 4. This simulation was used to design and test custom controller gains discussed in Chapter 5.

3.4.4 ROS

ROS combined with an OBC is used to autonomously pilot the UAV during the autolanding process. The Pixhawk flight controller is not capable of doing the required image processing

and thus the Nvidea Jetson Nano is used for this task. ROS is installed on the Jetson Nano and is responsible for extracting the required information from the incoming images and publishing the information to the flight controller. A SLAM algorithm is also implemented in ROS and is used to improve the existing localisation of the UAV.

ROS Usage

In this project ROS is used on the OBC. The OBC is connected to the flight controller via serial universal asynchronous receiver/transmitter (UART). Once the UAV is switched to offboard mode which could either be done using QGroundControl or via a switch on the radio controller, the UAV begins using data published by the ROS system via the MAVLink/MAVROS interface.

A custom package called "Honeybee Autolander" is created for the purpose of containing all nodes needed by the autolanding system. The package consists out of two nodes, namely, `Marker_Pose` and `Autolander_SM`. These two nodes as well as other ROS nodes used in this project are briefly discussed below.

Marker Pose

This node subscribes to the UAV's camera image topic and uses OpenCV to extract the pose of the landing target. This data is then published to the `Autolander_SM` node.

Autolander SM

This node contains the autolanding state machine. It subscribes to the vehicles local position and velocity topics as well as topics published by the `Marker_Pose` node. With this data the state machine publishes the required setpoints to PX4's controllers to make a highly accurate autolanding.

Waypoint Scheduler

This is a ROS node created by [56] and is used to command the UAV to defined waypoints. This is used to test the step responses of the UAV's custom controllers.

Fiducials

This package is used for ArUco marker generation and detection. This is used for detecting and estimating the pose of a stationary ArUco marker during the autolanding process. This package also contains a marker based SLAM node which is described further in Chapter 8.

Jetson Camera

This package contains a node that converts the onboard camera's image feed into a standard ROS image topic which was used by all other nodes requiring the onboard images.

Robot Upstart

This package is used to automatically start the required ROS nodes when the Jetson Nano starts up.

3.4.5 ROS Simulation

Simplistic integration of the ROS system with PX4/Gazebo SITL and HITL simulations is possible because ROS uses the same publish-subscribe communication architecture as PX4 and Gazebo.

PX4 uses the micro air vehicle link (MAVLink) protocol for communication. It is designed as a library defining all the messages and topics used on the UAV. It is mainly used to communicate with ground station software such as QGroundControl but it also enables autopilots such as ROS. MAVROS is the bridge between ROS and the MAVLink protocol. When the ROS system sends a message to PX4 it is published as a MAVROS topic. When a PX4 module subscribes to the topic it is converted to a MAVLink topic and vice versa.

MAVROS is started as a node on the ROS system. This node is designed to act as the ROS Master node and coordinates all communication in the ROS system. All other nodes need to register with the MAVROS master node in order to communicate with others. Figure 3.9 illustrates the communication structure in a typical simulation or practical flight test.

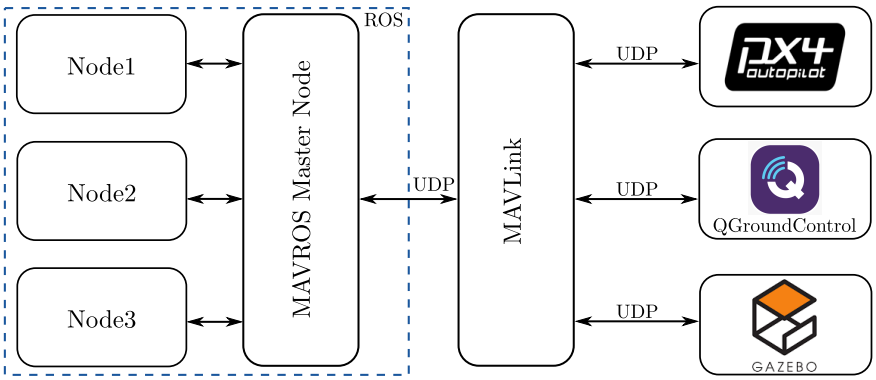


Figure 3.9: Communication between ROS, PX4, Gazebo and ground control station

3.5 Summary

In this chapter the physical components and avionics of the UAV were selected. The Pixhawk mini was chosen for the avionics system due to compact size and ability to run flight control software such as PX4. The propulsion system of the UAV was designed to maximise the flight time of the UAV. The UAV thus uses a small lightweight carbon fibre racing drone airframe and low current motors. The UAV's physical parameters were obtained using a combination of experimental and mathematical methods as part of the system identification process. These parameters are used for controller design and simulation purposes. The physical parameters of the UAV are summarised in Table 3.7.

This chapter also discussed the software toolchain required for simulating and performing practical flight tests. PX4 was chosen as the flight control stack due to the fact that its open source and has a convenient development cycle . PX4 also offers a rich set of features for users and developers and is compatible with ROS. Gazebo simulator is used to test custom software before practical flights are performed. QGroundControl is used to setup and calibrate the UAV and is also used to monitor the UAV's systems during practical flight tests. ROS is used on the OBC to act as an autopilot and to add high level custom control and functionality to

the UAV. How these systems interact and communicate in simulation or during a practical flight was discussed.

Parameter	**Value**
Mass	0.808 m
Motor Distance	0.11 m
Battery voltage	4S - 14.8 V
Max payload capacity	2.33 kg
Flight Time	12.76 min
Motor time constant	15 ms
Ixx	2e-3 Kgm^2
Iyy	1.32e-3 Kgm^2
Izz	3.35e-3 kgm^2
Virtual moment arm coefficient (R_N)	7.997e-3 m

Table 3.7: Summery of vehicle physical parameters

Chapter 4

Modelling

In this chapter the non-linear model of the UAV is derived. This is used to build a non-linear simulation and to design and test and practically implement custom controllers.

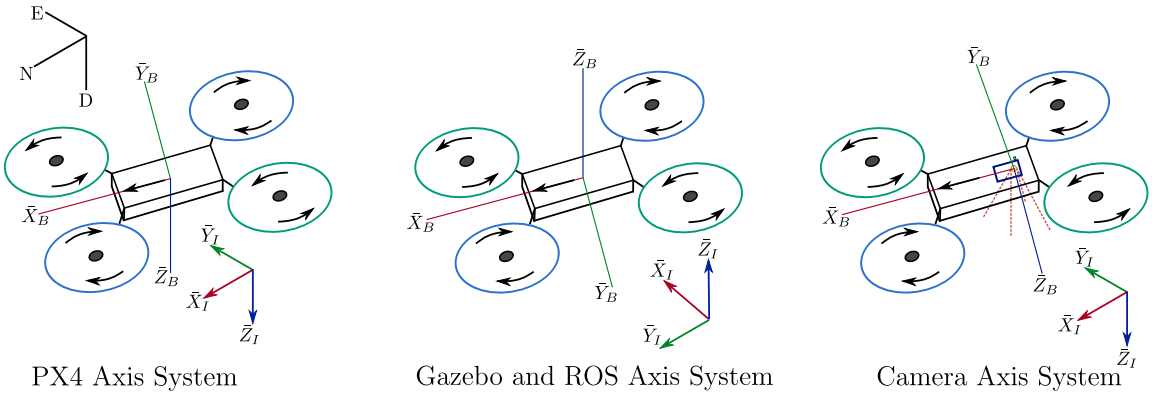
The coordinate frames used by PX4 are first defined. The six-degrees-of-freedom (6 DoF) equations of motion of the UAV are derived in terms of its kinetics and kinematics. A definition of the quaternion representation of the UAV's attitude is presented as this is the same method used within PX4. The forces and moments acting on the UAV is derived for the Forces and Moments model.

4.1 Coordinate Frames

Two main coordinate frames can be defined, the inertial or world coordinate frame, $I = \{\bar{X}_I, \bar{Y}_I, \bar{Z}_I\}$ and the body coordinate frame, $B = \{\bar{X}_B, \bar{Y}_B, \bar{Z}_B\}$. For the inertial coordinate frame a North-East-Down (NED) axis system is used. This axis system has its point of origin at the location where the UAV receives an initial GNSS lock during startup. This axis system describes the translation of the UAV from this origin relative to magnetic North, East and Down. For this axis system the curvature of the earth is neglected as the UAV will never fly far enough for it to be taken into account. A flat earth model is therefore assumed.

The body coordinate frame is fixed to the UAV with its origin aligned with the orientation and position of the flight controller by default, however a rotation can be specified for arbitrary configurations. It is most ideal to define the body coordinate frame at the centre of mass of the vehicle.

The way that these axis systems are defined differs between the various systems used in this project. PX4 uses a NED convention, this means that the x-axis coincides with the North direction, y-axis coincides with the East direction and the z-axis points Down. This convention is different for Gazebo simulator and ROS which uses a East-North-Up (ENU) convention. Its axis system is defined with the x-axis coinciding with the East direction, y-axis coinciding with the North direction and its z-axis pointing Up. The onboard camera also defines its own axis system. PX4's axis system was used as the reference axis during this project. The different axis systems are illustrated in Figure 4.1.



PX4 Axis System | Gazebo and ROS Axis System | Camera Axis System

Figure 4.1: Different axis systems

4.2 Aircraft Model Overview

Figure 4.2 provides a block diagram showing the non linear aircraft model used in this chapter.

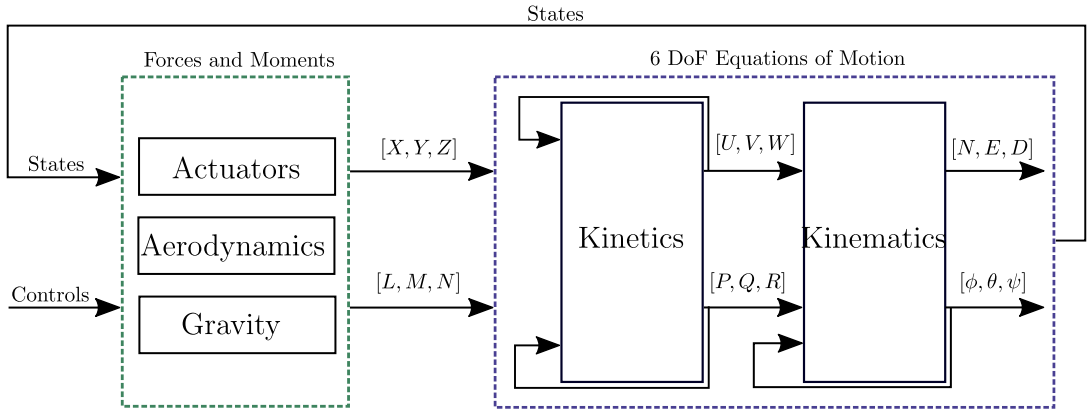


Figure 4.2: Aircraft model

The block diagram above shows the two main components of the UAV plant model. The 6 DoF Equations of Motion describe the motion of the aircraft given the forces and moments acting on it. These are calculated in the Forces and Moments model using the current actuator controls and states of the aircraft. The notation used for this model are presented in Table 4.1.

Variable	Description
6 DoF model variables	
$[U, V, W]$	Body axis defined linear velocity vector, (axial, lateral, normal velocity).
$[P, Q, R]$	Body axis defined angular velocity vector, (roll, pitch, yaw rate).
$[N, E, D]$	Inertial axis defined position vector measured in m relative to the takeoff location.
$[\phi, \theta, \psi]$	Body axis defined Euler 3-2-1 attitude vector, (roll, pitch, yaw angle).
Forces and Moments model variables	
$[X, Y, Z]$	Body axis defined force vector, (axial, lateral, normal force).
$[L, M, N]$	Body axis defined moment vector, (roll, pitch, yaw moment).

Table 4.1: Description of aircraft model variables

4.3 Six-Degrees-of-Freedom Equations of Motion

This section provides derivations for the 6 DoF equations of motion of the UAV. The UAV is modelled based on its kinetics and kinematics. The kinetics of the UAV describe how it moves based on the forces and moments acting on it. The kinematics of the UAV describe its acceleration, velocity and positional change [59]. The UAV has six-degrees-of-freedom, three for its translation along the NED axis and another three for its rotation in 3D space.

4.3.1 Kinematics

Kinematics tries to describe the position, acceleration and velocity of the UAV in the inertial coordinate frame. It uses the current linear velocity, angular velocity and orientation of the UAV to do this. The attitude of the vehicle is required to obtain the the transformation of the inertial velocity from the body frame defined linear velocity. This transformation of body to inertial frame makes use of the direct cosine matrix (DCM).

The attitude required to calculate the DCM can either be defined in Euler angles or in Quaternion coordinates. The disadvantage of using Euler angles is that a singularity exists when the pitch angle of the UAV reaches $\frac{\pi}{2}$ for an Euler 3-2-1 definition. For this reason PX4 does not use an Euler representation of the UAV's attitude. It uses a quaternion representation which does not contain this singularity.

Quaternions overview

A brief definition of quaternions are presented before proceeding to derive the kinematic equations. The quaternion derivations are based off of work done by [62].

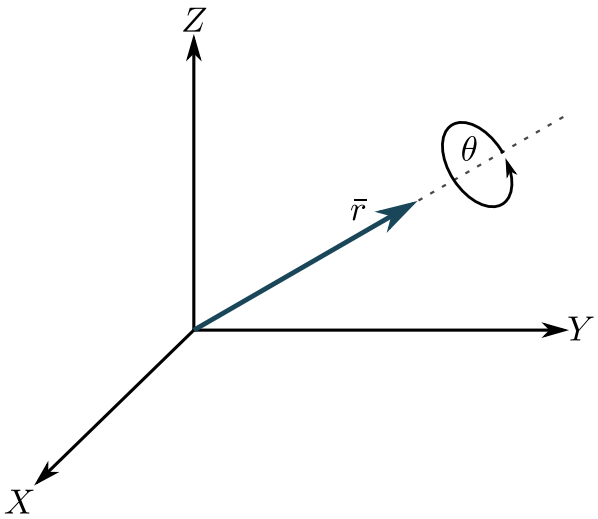


Figure 4.3: Quaternion angle rotation

From Figure 4.3 consider a rotation around the unit vector $\bar{\mathbf{r}} = [r_x \; r_y \; r_z]^T$ by angle θ. From this rotation the quaternion $\bar{\mathbf{q}} = [q_0 \; q_1 \; q_2 \; q_3]^T$ can be calculated using:

$$
\begin{aligned}
q_0 &= \cos\left(\frac{\theta}{2}\right), \\
q_1 &= r_x \sin\left(\frac{\theta}{2}\right), \\
q_2 &= r_y \sin\left(\frac{\theta}{2}\right), \\
q_3 &= r_z \sin\left(\frac{\theta}{2}\right).
\end{aligned} \tag{4.1}
$$

A quaternion consists out of a magnitude component which is represented by q_0 and a vector which is represented by, $\mathbf{q}_v = [q_1 \ q_2 \ q_3]^T$. In this project the unit quaternion is used which satisfies

$$
|\bar{\mathbf{q}}| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1. \tag{4.2}
$$

Body to Inertial frame transformation

Now that the quaternion representation of the UAV's orientation has been defined the attitude of the UAV can be calculated using the known angular rates, $\dot{\boldsymbol{\omega}}_B$, measured in the body frame of UAV. This can be done using the following equation:

$$
\begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ P \\ Q \\ R \end{bmatrix}. \tag{4.3}
$$

The full derivation of Equation 4.3 and 4.4 can be found in [60]. The Direct Cosine Matrix can now be calculated using:

$$
\mathbf{R}_V = \begin{bmatrix} q_0^2 + q_1^2 + q_2^2 + q_3^2 & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_1 q_3 + q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}, \tag{4.4}
$$

where $\mathbf{R}_V$ is the DCM transformation matrix. The transformation to the inertial frame can now be done using

$$
\mathbf{V}_I = \mathbf{R}_V^{-1} \mathbf{V}_B, \tag{4.5}
$$

where $\mathbf{V}_I$ is the inertial frame velocity vector and $\mathbf{V}_B$ is the body frame velocity vector. The inertial position vector can be obtained by integrating the inertial velocity.

4.3.2 Kinetics

The kinetic equations describe the motion of the UAV based on the forces and moments acting on it. This can be expressed using Newton's second law of motion as

$$
\begin{aligned}
\mathbf{F}_B &= m_t \dot{\mathbf{V}}_B + \boldsymbol{\omega}_B \times m_t \mathbf{V}_B, \ \textit{and} \\
\mathbf{M}_B &= \mathbf{I}_V \dot{\boldsymbol{\omega}}_B + \boldsymbol{\omega}_B \times \mathbf{I}_V \boldsymbol{\omega}_B,
\end{aligned} \tag{4.6}
$$

where m_t is the total mass of the UAV, $\mathbf{F}_B = [X \ Y \ Z]^T$ and $\mathbf{M}_B = [L \ M \ N]^T$ represent the force and moment vectors acting on the body of the UAV. $\mathbf{I}_V$ is defined as

$$
\mathbf{I}_V = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{zy} & I_{zz} \end{bmatrix}. \tag{4.7}
$$

The UAV's linear and angular velocities, defined in the body frame of the UAV are:

$$\mathbf{V}_B = [U\ V\ W]^T, and \tag{4.8}$$

$$\boldsymbol{\omega}_B = [P\ Q\ R]^T. \tag{4.9}$$

Rearranging and simplifying the above equations the kinetic equations of the UAV are expressed as:

$$\dot{P} = \frac{L + QR(I_{yy} - I_{zz})}{I_{xx}} \tag{4.10}$$

$$\dot{Q} = \frac{M + PR(I_{zz} - I_{xx})}{I_{yy}} \tag{4.11}$$

$$\dot{R} = \frac{N + PQ(I_{xx} - I_{yy})}{I_{zz}} \tag{4.12}$$

$$\dot{U} = \frac{X}{m_t} - WQ + VR \tag{4.13}$$

$$\dot{V} = \frac{Y}{m_t} - UR + WP \tag{4.14}$$

$$\dot{W} = \frac{Z}{m_t} - VP + UQ \tag{4.15}$$

These equations require the forces and moments acting on the UAV body. These are modelled in the forces and moments model.

4.4 Forces and Moments Model

The three forces and moments acting on the UAV that are modelled in this chapter are gravity, aerodynamics and thrust generated by the UAV's actuators. The total forces and moments acting on the UAV are therefore:

$$\mathbf{F}_B = \mathbf{F}^T + \mathbf{F}^A + \mathbf{F}^G, \text{ and} \tag{4.16}$$

$$\mathbf{M}_B = \mathbf{M}^T + \mathbf{M}^A + \mathbf{M}^G. \tag{4.17}$$

Here the superscripts T, A and G represent the thrust generated by the actuators, aerodynamics and gravity respectively. Each contributing force is modelled separately in the following subsections.

4.4.1 Actuator Thrust

The actuators of the UAV are the four motors and their propellers. Each motor produces a vertical thrust T. The magnitude of this thrust is dependant on the rotational velocity of the motor. There are many different quadrotor motor configurations where the relative locations of their motors and the airframe differ. A quadrotor-X configuration is used for the UAV in this project and is illustrated in Figure 4.4.
The thrust produced by each motor is modelled by the following first order differential equation:

$$T = T_R - \tau\dot{T} \tag{4.18}$$

where τ represents the motor time constant and T_R represents the reference thrust value.

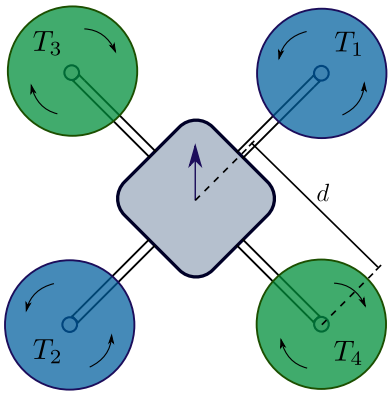


Figure 4.4: Quadrotor X

Virtual actuators are defined based on the actuators used to control a fixed wing aircraft. These are: throttle, ailerons, elevators and rudder. Each is coupled to one of the principle axis of the aircraft, namely: forward thrust, roll, pitch and yaw respectively. For a quadrotor a unique combination of its four motors are coupled to these virtual actuators. These relationships are defined by:

$$\delta_T = T_1 + T_2 + T_3 + T_4, \tag{4.19}$$

$$\delta_A = \frac{1}{\sqrt{2}}(-T_1 + T_2 + T_3 - T_4), \tag{4.20}$$

$$\delta_E = \frac{1}{\sqrt{2}}(T_1 - T_2 + T_3 - T_4) \text{ and} \tag{4.21}$$

$$\delta_R = T_1 + T_2 - T_3 - T_4. \tag{4.22}$$

The subscripts T, A, E and R represent the virtual actuators: throttle, ailerons, elevators and rudder respectively. It is now possible to define the forces and moments acting on the UAV in terms of the virtual actuators as

$$Z^A = \delta_T \tag{4.23}$$

$$L^A = d\delta_A \tag{4.24}$$

$$M^A = d\delta_E \tag{4.25}$$

$$N^A = R_N\delta_R \tag{4.26}$$

Where R_N represents the virtual yaw moment arm coefficient defined in Section 3.3.4 and d represents the motor arm length shown in Figure 4.4.

A mixing matrix is used to transform the virtual actuators to the real actuators of the UAV. The mixing matrix used is specific to the quadrotor-X airframe configuration and is given as:

$$\begin{bmatrix} \delta_T \\ \delta_A \\ \delta_E \\ \delta_R \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \tag{4.27}$$

Using the mixing matrix the virtual actuators are transformed using

$$\mathbf{T}_i = \mathbf{K}_M^{-1}\boldsymbol{\delta}_i, \tag{4.28}$$

ehere $\mathbf{K}_M$ is the mixing matrix, $\boldsymbol{\delta}_i$ represents the virtual actuators and $\mathbf{T}_i$ represents the motor actuators of the UAV.
The forces and moments due to actuators can be summarised as:

$$\mathbf{F}^A = \delta_t \bar{z}_B \tag{4.29}$$

$$\mathbf{M}^A = d\delta_A \bar{x}_B \;\; d\delta_E \bar{y}_B \;\; R_N \delta_R \bar{z}_B \tag{4.30}$$

4.4.2 Gravity

Gravity is a vertical force that acts normal to the inertial frame. The force of gravity however is required in the body frame of the UAV. The DCM is used to do this transformation. The forces and moments acting on the body of the UAV can be expressed using

$$\mathbf{F}^G = \mathbf{R}_V \begin{bmatrix} 0 \\ 0 \\ m_t g \end{bmatrix}, \text{ and} \tag{4.31}$$

$$\mathbf{M}^G = 0. \tag{4.32}$$

4.4.3 Aerodynamics

Aerodynamic force refers to the force of air resistance that increases proportionally to the relative velocity of the UAV and the oncoming wind. This force is dependant on the body's surface area exposed to the oncoming air. This is difficult to model mathematically as the UAV consists out of complex geometry. As the UAV flies faster the drag force experienced will increase in the opposite direction of motion. The equations describing the aerodynamic force is based on work done by [61]. This force of air resistance acting on the body of the UAV can be expressed using

$$\mathbf{F}^A = \frac{1}{2}\rho \mathbf{V}_{BW}|\mathbf{V}_{BW}| \begin{bmatrix} C_{Dx} \\ C_{Dy} \\ C_{Dz} \end{bmatrix}, \tag{4.33}$$

where,

$$\mathbf{V}_{BW} = -\mathbf{V}_B + \mathbf{R}_V \mathbf{V}_W. \tag{4.34}$$

ρ refers to the air density, C_{Dx}, C_{Dy} and C_{Dz} refer to the drag coefficients of the UAV's surface area. $\mathbf{V}_{BW}$ refers to the relative velocity of the UAV and the wind acting upon it. $\mathbf{V}_W$ refers to the linear velocity of the wind measured in the inertial frame and thus requires a DCM transformation.

4.5 Summary

In this chapter the aircraft model was presented. This consisted out of a six-degree-of-freedom model which describes the relationship between the forces and moments acting on the vehicle and the vehicles response to those forces in terms of its acceleration, velocity and position.

This model uses a quaternion representation of the UAV's attitude and so a brief overview of quaternions was presented. This chapter discussed the forces and moments model which aims to describe the applied forces and moments due to gravity, aerodynamic drag forces and lastly due to the motor and propeller pairs. This model can be used to simulate the UAV given its physical parameters. This model can also be used to design a control system for the UAV.

Chapter 5

Control System Design and Analysis

This chapter provides an overview of the quadrotor UAV controllers used by PX4. The UAV used for this project must possess a unique flight response suitable for the purpose of this project. To ensure this, custom controller gains for PX4 are designed.

The control system architecture used by PX4 is implemented in MATLAB/Simulink where the controllers could be analysed and designed. The controllers are then tested in a Gazebo SITL simulation with the actual PX4 flight control software. These controllers are then verified in a number of practical flight tests. The description of PX4's controller architecture in this chapter is based on work done by [56].

5.1 Control System Design Strategy

PX4 enables several default quadrotor airframes that vary in size, weight and motor configuration. One of these default airframes is the ZMR250 Racer. This airframe is similar to the UAV used for this project. These gains were uploaded to the flight controller and first tested in a controlled practical flight test. The UAV flew well and was stable throughout the flight, however, the flight response was quite aggressive due to its racing drone configuration. This meant that working controller gains were available that could be used as a starting point for designing custom controller gains.

The equations derived in Chapter 4 are linearised to obtain the linear plant of the UAV. This is implemented together with continuous-time controllers in MATLAB/Simulink for the design process. The root locus method is used to adjust the individual controllers until a desired step response is obtained and until there is sufficient bandwidth separation between the various controllers. Each successive controller has to be at least 2 times slower than the previous controller to ensure sufficient bandwidth separation. The inner-most controllers are thus designed first before designing the outer controllers.

The UAV's onboard camera is fixed mounted to the frame. The camera is thus not free to rotate independently from the UAV, unlike gimbal mounted cameras. This means that the quality of the image feed is directly coupled to the stability and orientation of the UAV. In this project the UAV is required to track a stationary object on the ground during landings. The UAV must thus have a smooth and slow flight response in order to minimise pitch and roll angles. This improves the overall stability and quality of the onboard camera image.

5.2 PX4 Control System Architecture

PX4's controllers consist out of inner attitude controllers which are defined in the body frame and outer translational controllers which are defined in the inertial frame. These two main controller categories are coupled together by a force and yaw to attitude and thrust conversion layer in a cascaded loop architecture shown in Figure 5.1.

The attitude controllers consist out of an angular rate controller which gets its reference from an angle controller. The translational controllers consist out of a velocity controller which gets its reference from a positional controller.

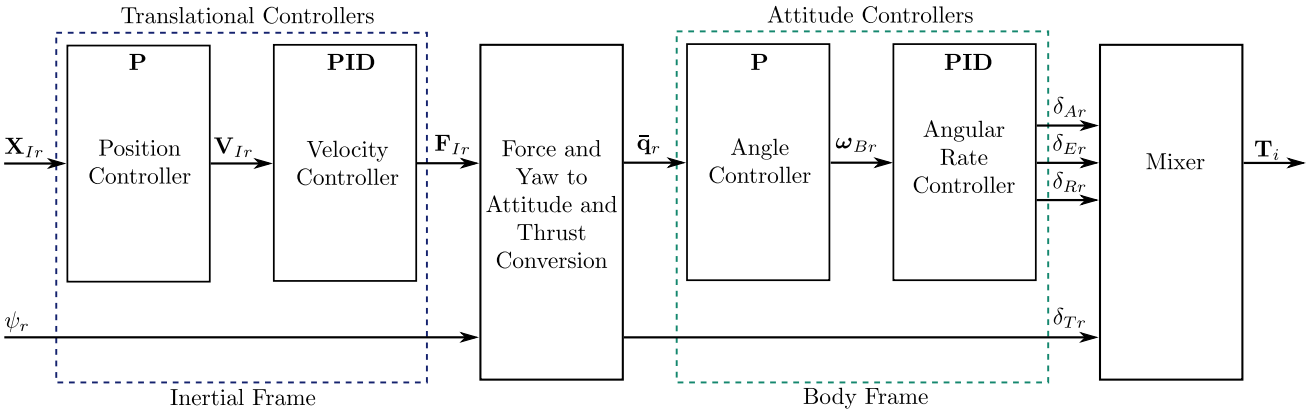


Figure 5.1: PX4 cascaded loop controller architecture

The subscript r denotes a reference signal. Each individual component in Figure 5.1 is analysed and discussed in the following sections.

5.3 PX4 Mixer

PX4 uses normalised virtual actuator thrusts. $\delta_{A,PX4}$, $\delta_{E,PX4}$ and $\delta_{R,PX4}$ are thus limited to [-1, 1]. The virtual actuator $\delta_{T,PX4}$ and the motor actuators $\mathbf{T}_{i,PX4}$ are normalised to [0,1]. The derived equations in Section 4.4.1 are defined in Newtons therefore they need to be redefined for this chapter. The maximum thrust of a UAV in Newton units is defined as T_{MAX} which was obtained in Section 3.3.1. Therefore this value can be used as a scaling factor to relate the motor actuators to the normalised motor actuators using

$$\mathbf{T}_i = T_{MAX}\mathbf{T}_{i,PX4}. \tag{5.1}$$

From [56] the simplified mixing matrix used by PX4 can be expressed as

$$\begin{bmatrix} T1 \\ T2 \\ T3 \\ T4 \end{bmatrix}_{PX4} = \begin{bmatrix} 1 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 1 \\ 1 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 1 \\ 1 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -1 \\ 1 & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -1 \end{bmatrix} \begin{bmatrix} \delta_T \\ \delta_A \\ \delta_E \\ \delta_R \end{bmatrix}_{PX4}, \tag{5.2}$$

which can be rewritten as:

$$\mathbf{T}_{i,PX4} = \mathbf{K}^{-1}_{M,PX4}\boldsymbol{\delta}_{i,PX4}. \tag{5.3}$$

By substituting Equations 4.28 and 5.3 into 5.1 the relationship between the virtual actuators and the normalised virtual actuators can be expressed as:

$$\begin{aligned} \mathbf{K}_M^{-1}\boldsymbol{\delta}_i &= T_{MAX}\mathbf{K}_{M,PX4}^{-1}\boldsymbol{\delta}_{i,PX4}, \\ \boldsymbol{\delta}_i &= \mathbf{K}_M\mathbf{K}_{M,PX4}^{-1}\boldsymbol{\delta}_{i,PX4}, \\ \begin{bmatrix}\delta_T \\ \delta_A \\ \delta_E \\ \delta_R\end{bmatrix} &= T_{MAX}\begin{bmatrix}4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4\end{bmatrix}\begin{bmatrix}\delta_{T,PX4} \\ \delta_{A,PX4} \\ \delta_{E,PX4} \\ \delta_{R,PX4}\end{bmatrix}. \end{aligned} \tag{5.4}$$

5.4 Angular Rate Controller

Linear PID control is used for the angular rate controller. Here the roll, pitch and yaw rates ($\boldsymbol{\omega}_B$) in the body frame are being controlled with the reference input $\boldsymbol{\omega}_{Br}$. The output of this controller are the virtual actuator commands $\boldsymbol{\delta}_i$ where $i = \{A\ E\ R\ T\}$. Figure 5.2 shows the controller implementation used by PX4.

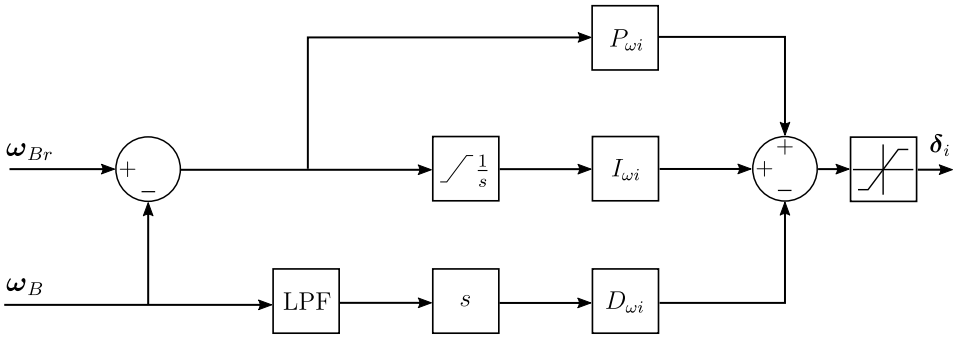


Figure 5.2: PX4 angular rate controller

Some components shown in Figure 5.2 are not typically used for a classic PID controller, however these non-standard components do improve the overall stability of the controller response. These improvements are summarised below:

- LPF, this low pass filter is included to remove noise before it is amplified by the derivative term, $D_{\omega i}$. PX4 includes a parameter for specifying the cut off frequency of the LPF. This noise, usually caused by vehicle vibration can be seen in Figure 5.3.
- The derivative term also receives its input directly from the state measurements. This is done in order to avoid a phenomenon called derivative kick.
- The integral term contains a saturation element. This is used to avoid a phenomenon called integrator wind-up.
- The virtual actuator reference signal also contains a saturation element that avoids the motors from saturating in the case of large reference inputs.

Figure 5.3 shows the frequency response of raw accelerometer data during a practical flight. This is obtained by inspecting PX4's flight logs. This figure gives an indication of how vibration in the system relates to noise in sensitive sensors such as the accelerometer and gyroscope. Motor vibration introduces signal noise in the frequency band between 60 and 90

Hz. The cutoff frequency of the LPF was thus chosen at 40 Hz to maximise the quality of the accelerometer and gyroscope measurements.

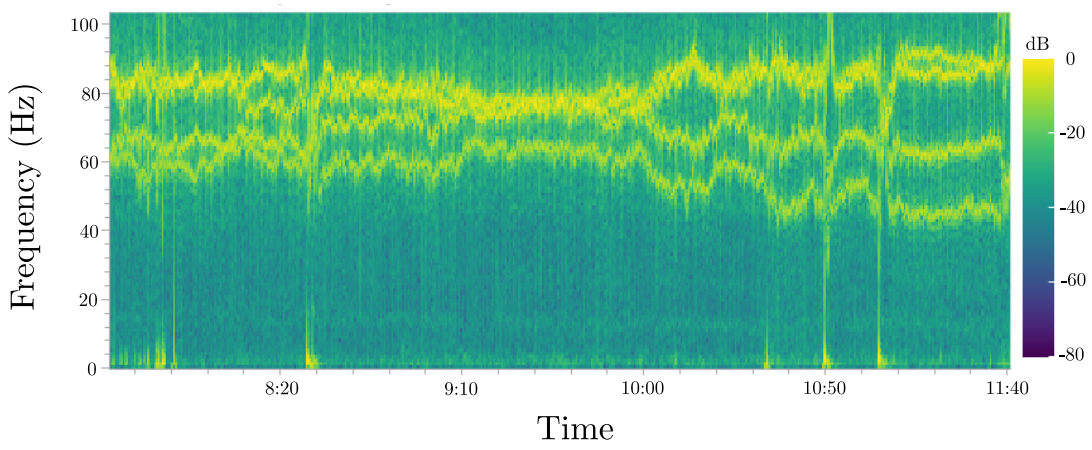


Figure 5.3: Acceleration power-spectral-density

Only the longitudinal dynamics will be considered as the lateral dynamics are approximated to be similar due to symmetry about the x and y axis of the UAV. The pitch rate linear plant can be given as,

$$G_{\omega p}(s) = \frac{\omega_{Bp}(s)}{\delta_{Er,px4}(s)} = \frac{2T_{max}\frac{d}{\tau I_{yy}}}{s(s+\frac{1}{\tau})}. \tag{5.5}$$

The subscript p denotes the pitch component of the angular rate controller, d refers to the motor arm length and τ refers to the motor time constant. The inclusion of $2T_{max}$ is to scale the normalised virtual actuator $\delta_{Er}(s)$. The block diagram in Figure 5.4 shows the simplified PID controller and plant of the pitch rate controller.

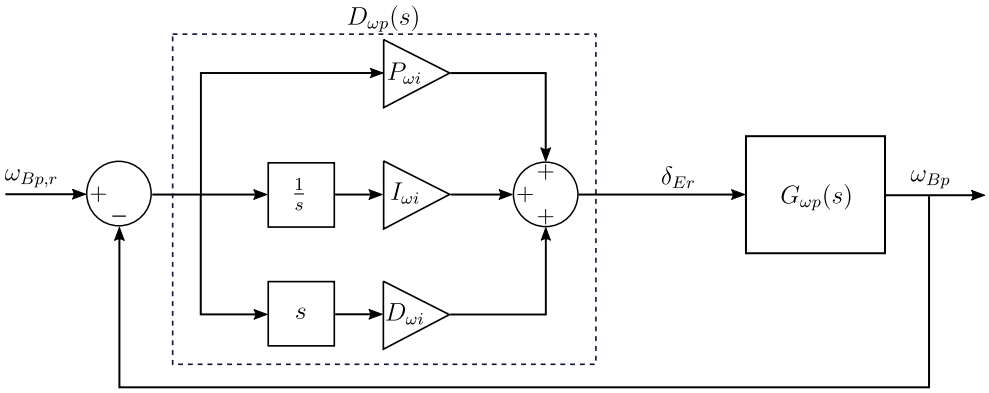
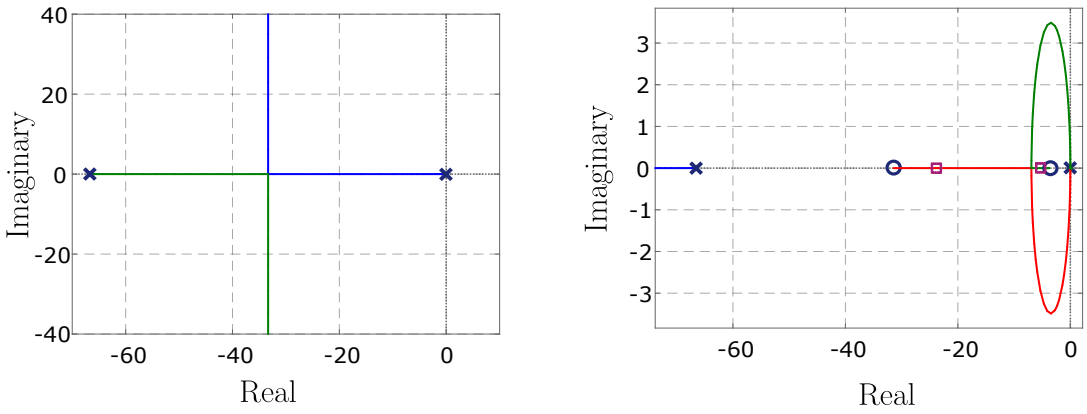


Figure 5.4: Pitch rate controller and plant

The root locus of the plant and plant with controller are shown in Figures 5.5a and 5.5b respectively.

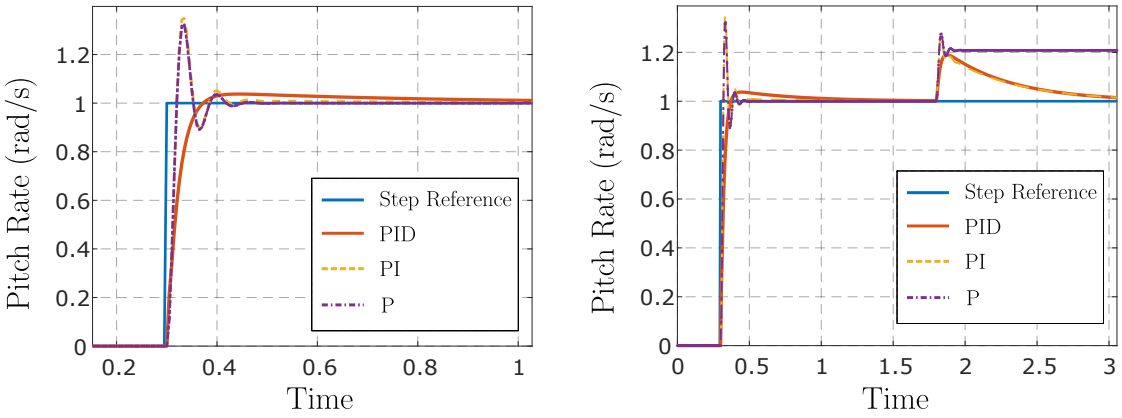


(a) Plant root locus (b) Plant and controller root locus

Figure 5.5: Root locus of the pitch rate plant and the controller with plant

The following specifications are required by the pitch rate controller:

- Must have a fast transient response as the angular rate controller is mostly responsible for maintaining stability.
- Must have fast disturbance rejection.
- Must have minimal overshoot.



(a) Step response (b) With disturbance

Figure 5.6: Pitch rate controller step response and disturbance rejection

Figure 5.6b shows that the integral term is needed to reject disturbance. The system has two free integrators one from the integral term in the controller $D_{\omega p}(s)$ and one from the natural plant dynamics. This allows the system to track a ramp reference with zero steady state error. The derivative term helps in damping the system response as seen in Figure 5.6a. The PID combination results in a step response with 3.6% overshoot, rise time of 24 ms, 2% settling time of 0.8 s and bandwidth of 138 rad/s.

The default ZMR250 Racer PID gains are used to generate the root locus in Figure 5.5 and the step response shown in Figure 5.6. These gains provide a fast, well damped response and also offers fast disturbance rejection. These gains are thus well suited for the UAV used in this project and will not be changed. With these gains the UAV will be robust to small disturbances ensuring that the UAV remains stable. The lateral roll rate controller is also kept the same.

5.5 Angle Controller

The angle controller is used to control the pitch, roll and yaw angles. The pitch and roll angle controllers are equal. PX4 uses a quaternion instead of an Euler representation of these angles. The following control law implementation is based on work from [63] as discussed by [56].

The error quaternion can be expressed using

$$\bar{\mathbf{q}}_e = \bar{\mathbf{q}}^{-1} \cdot \bar{\mathbf{q}}_r. \tag{5.6}$$

$\bar{\mathbf{q}}$ represents the quaternion representation of the estimated orientation of the UAV. $\bar{\mathbf{q}}_r$ represents the reference angle in quaternions and $\bar{\mathbf{q}}_e$ represents the error angle in quaternions. The proposed control law can thus be given as:

$$\boldsymbol{\omega}_{Br} = 2P_q sgn(q_{0e}) q_{ve} \tag{5.7}$$

Where $\boldsymbol{\omega}_{Br}$ is the angular rate reference commanded by the angle controller, P_q is the angle controllers proportional gain, q_{0e} and q_{ve} is the respective magnitude and vector components of the error quaternion $\bar{\mathbf{q}}_e$ as shown in Equation 4.1. The angle controller is shown in Figure 5.7.

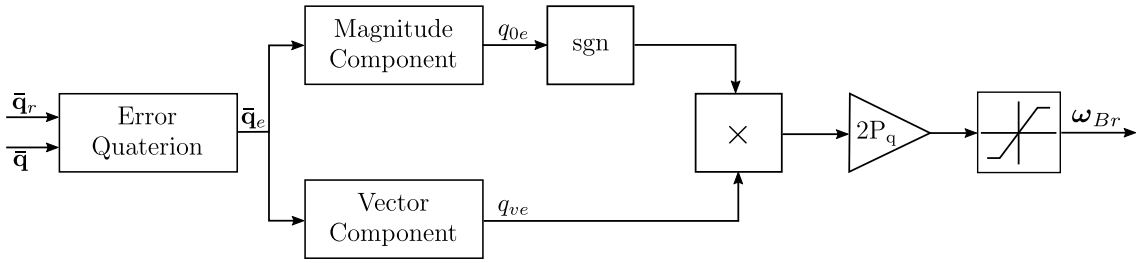


Figure 5.7: Angle controller architecture

In order to linearise the control law for controller design it is assumed that q_{0e} will be greater than zero. Therefore the signum of q_{0e} will be equal to 1. And secondly, the small angle approximation is made that results in the error quaternion vector to be:

$$q_{ve} = q_{vr} - q_v, \tag{5.8}$$

therefore the linearised control law can be given as,

$$\boldsymbol{\omega}_{Br} = 2P_q(q_{vr} - q_v). \tag{5.9}$$

From the model derived in Chapter 4, the linear angle plant is given as:

$$G_q(s) = \frac{\bar{\mathbf{q}}(s)}{\boldsymbol{\omega_{Br}}(s)} = 2G_{\omega,cl}(s) \cdot \frac{0.5}{s} \tag{5.10}$$

Where $G_{\omega,cl}(s)$ is the closed loop angular rate dynamics which can be expressed as:

$$G_{\omega,cl}(s) = \frac{\boldsymbol{\omega}_B(s)}{\boldsymbol{\omega}_{Br}(s)} = \frac{D_\omega(s)G_\omega(s)}{1 + D_\omega(s)G_\omega(s)} \tag{5.11}$$

The block diagram of the linear angle controller and plant is shown in Figure 5.8.

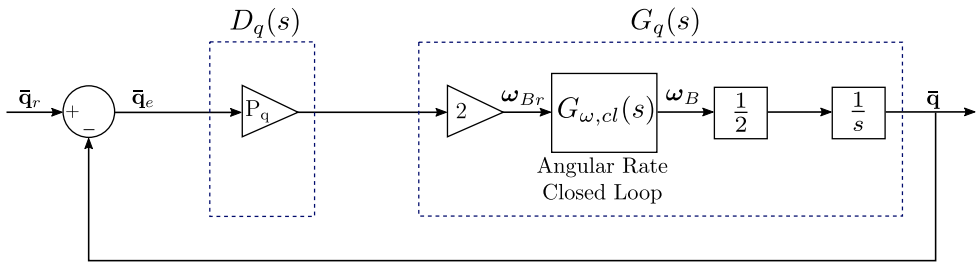
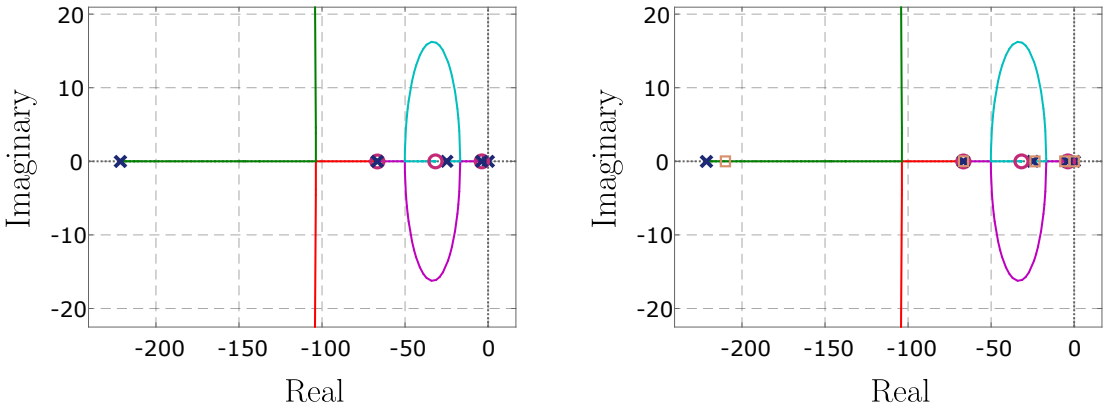


Figure 5.8: Linear angle controller and plant

The default ZMR250 Racer angle controllers proportional gain is implemented first. The step response is then analysed to test the practicality of this gain with respect to this projects requirements. The desired response is summarised as follows:

- Must have a critically damped response with no overshoot.
- Must have a fast transient response to remain responsive to required reference inputs.
- Must have good bandwidth separation from angular rate controller.



(a) Angle Plant (b) Angle Plant and Controller

Figure 5.9: Root locus of angle plant and controller

The root locus of the plant and controller is shown in Figure 5.9. The system does not require an integrator due to the already natural free integrator present in the plant. The inner angular rate controller and outer velocity controllers also already contain integrators. This system contains a dominant pole at -3.9 on the real axis which results in a well damped response shown in Figure 5.10.

The default ZMR250 Racers proportional gain value produces a slow step response with a bandwidth of 2 rad/s. A faster response with a higher bandwidth was required to allow for flexibility in designing the velocity controller. The proportional gain was thus increased to increase the bandwidth of the controller which results in the well damped response shown in Figure 5.10. This response has a rise time of 0.3 s, 2% settling time of 0.47 s and bandwidth of 11 rad/s this makes it well separated from the pitch rate controller while still leaving room for the outer velocity controller.

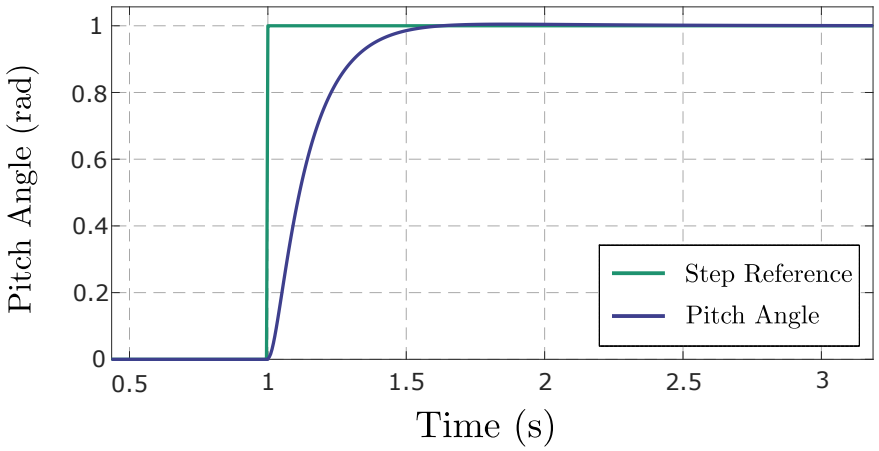


Figure 5.10: Step response of the pitch angle controller

5.6 Force and Yaw to Attitude and Thrust Conversion

The linear velocity controller outputs a desired force vector, $\mathbf{F}_{Ir}$ in the inertial frame. Located in between the velocity and angle controllers, a separate process is responsible for converting this force to a desired attitude and thrust vector. This process describes the desired orientation of the vehicle using a rotation matrix. This transformation is based off of work done by [64] as described by [56].

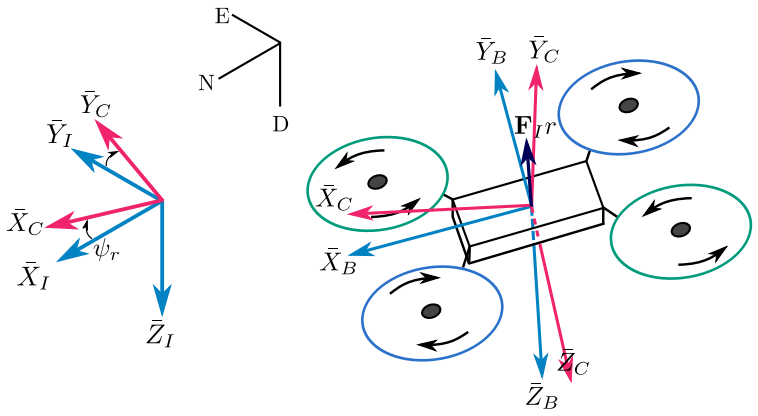


Figure 5.11: Illustrating the inertial force and desired yaw

The commanded force $\mathbf{F}_{Ir}$ is equal to the desired thrust vector Z^A of the UAV. With this force applied to the UAV the orientation of the UAV will be naturally coupled to the desired vertical thrust vector. The linear velocity controller thus commands this force so that the resulting attitude and thrust of the UAV commands the desired velocity. The commanded force can be described as

$$\mathbf{F}_{Ir} = [F_{IN,r} \;\; F_{IE,r} \;\; F_{ID,r}]^T. \tag{5.12}$$

The desired attitude $\{\bar{\mathbf{X}}_{Br} \; \bar{\mathbf{Y}}_{Br} \; \bar{\mathbf{Z}}_{Br}\}$ can be calculated from the force vector using

$$\bar{\mathbf{Z}}_{Br} = -\frac{\mathbf{F}_{Ir}}{|\mathbf{F}_{Ir}|}. \tag{5.13}$$

A new frame denoted by the subscript C describes the rotation about the inertial frames z component, $\bar{\mathbf{Z}}_I$ by the desired yaw angle, ψ_r. This rotation is given as

$$\bar{\mathbf{Y}}_{Cr} = [-sin(\psi_r) \;\; cos(\psi_r) \;\; 0]^T. \tag{5.14}$$

The attitude can be further calculated using:

$$\bar{\mathbf{X}}_{Br} = \frac{\bar{\mathbf{Y}}_{Cr} \times \bar{\mathbf{Z}}_{Br}}{|\bar{\mathbf{Y}}_{Cr} \times \bar{\mathbf{Z}}_{Br}|}, \text{ and} \tag{5.15}$$

$$\bar{\mathbf{Y}}_{Br} = \bar{\mathbf{Z}}_{Br} \times \bar{\mathbf{X}}_{Br}. \tag{5.16}$$

The rotation matrix describing the desired orientation of the UAV can then be expressed using

$$\mathbf{R}_v = [\bar{\mathbf{X}}_{Br} \;\; \bar{\mathbf{Y}}_{Br} \;\; \bar{\mathbf{Z}}_{Br}]. \tag{5.17}$$

This orientation is now converted to quaternions before being used by the angle controller.

5.7 Linear Velocity Controller

The linear velocity controller uses the same PID architecture as the angular rate controller shown in Figure 5.2.

The linear velocity plant describing the UAV's attitude and longitudinal inertial velocity is derived in [56, p.111]. This derived plant can be given as

$$G_{VN}(s) = \frac{V_N(s)}{q_{pr}(s)} = G_{qp,cl}(s) \cdot \frac{2g}{s}, \tag{5.18}$$

where the subscript p denotes the longitudinal pitch quaternion component and $G_{qp,cl}$ is the transfer function of the closed loop pitch angle controller. This can be formally given as

$$G_{qp,cl} = \frac{q_p(s)}{q_{pr}(s)} = \frac{D_q(s)G_{qp}(s)}{1 + D_q(s)G_{qp}(s)}. \tag{5.19}$$

From the derivation of the linear velocity plant in [56, p.111], the relationship between the commanded longitudinal force, $F_{IN,r}$ and the pitch quaternion component is given as:

$$F_{IN,r} = 2M_t g q_{pr} \tag{5.20}$$

with this, the transfer function relating the commanded longitudinal force to the linear longitudinal inertial velocity can be given as

$$G_{VN}(s) = \frac{V_N(s)}{F_{IN,r}} = G_{qp,cl}(s) \cdot \frac{\frac{1}{m_t}}{s}. \tag{5.21}$$

The force vector used by PX4 is normalised to [-1,1], therefore a scaling factor is required to transform the normalised force to the actual force the UAV is capable of producing. This relationship can be given as:

$$\mathbf{F}_{IR} = 4T_{MAX}\mathbf{F}_{IR}^{PX4} \tag{5.22}$$

Including the normalised force the new transfer function $G_{VN}(s)$ is given as:

$$G_{VN}(s) = \frac{V_N(s)}{F_{IN,r}^{PX4}} = G_{qp,cl}(s) \cdot \frac{\frac{4T_{MAX}}{m_t}}{s} \tag{5.23}$$

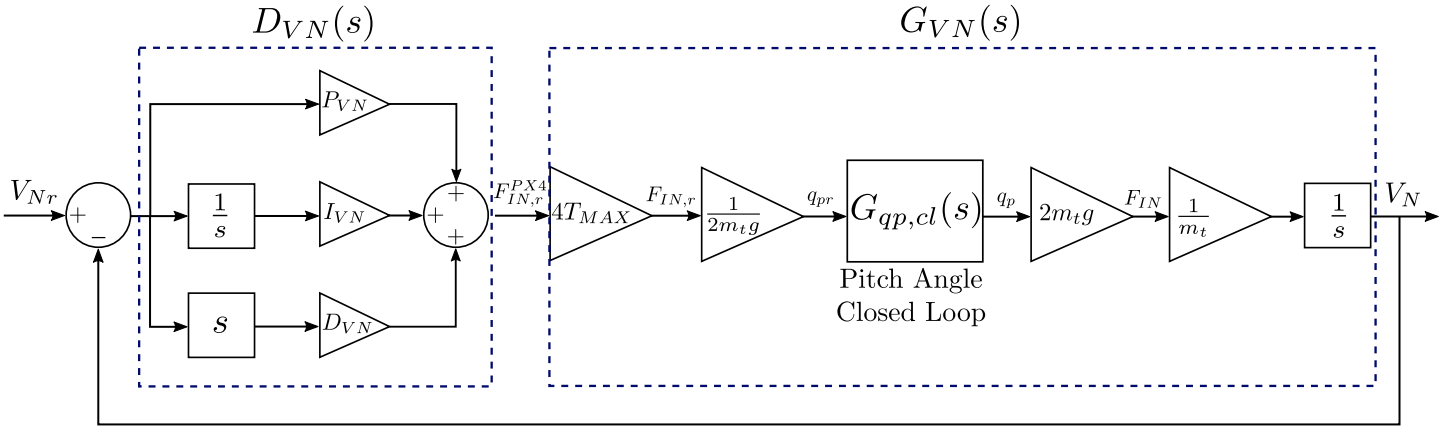
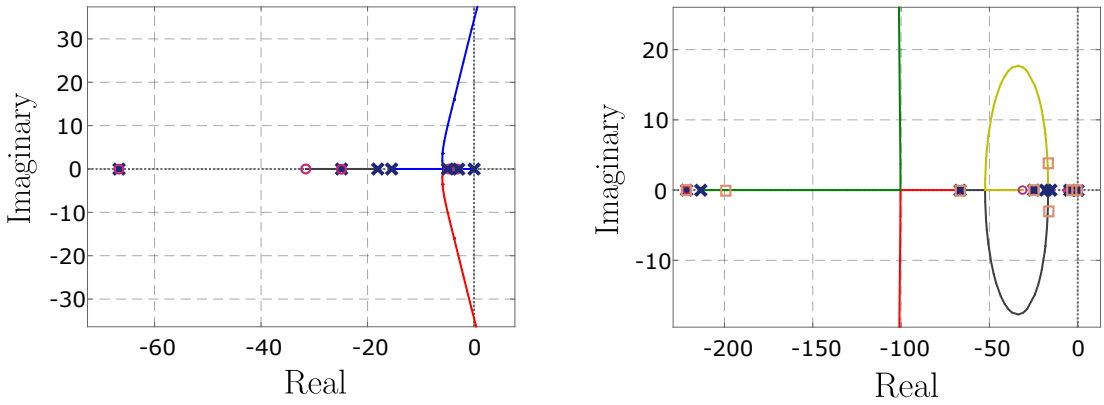


Figure 5.12: Linear velocity controller and plant block diagram

The block diagram of the linear velocity controller and plant is shown in Figure 5.12.

Initially the default ZMR250 Racer velocity controller PID gains were originally implemented as this controllers response was considered acceptable. During initial practical autolanding tests it was observed that the UAV oscillated slightly around the marker target due to excessive overshoot in this controller and thus these gains had to be redesigned. The proportional gain was first adjusted to satisfy the bandwidth requirement before increasing the D term to reduce the controller's overshoot. By lowering the integral term the overshoot can be lowered further but results in an undesirable longer settling time. The attitude controllers make the UAV susceptible to drift in the presence of disturbance. Wind is also a contributing factor in UAV drift. The velocity controller is used to reduce these effects. It is thus important that the velocity controller be capable of rejecting these disturbances as fast as possible. It is thus decided not to lower the integral term further as this reduces the disturbance rejection ability of the UAV. The root locus of the linear velocity plant and controller with plant can be seen in Figures 5.13a and 5.13b respectively.



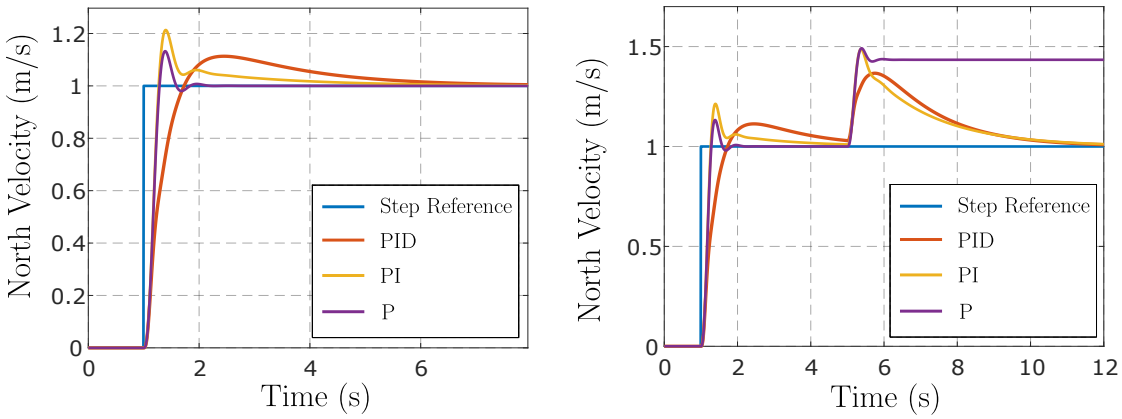
(a) Velocity Plant

(b) Velocity Plant and Controller

Figure 5.13: Root locus of velocity plant and controller

Referring to the step responses in Figures 5.14a and 5.14b, the velocity controller has an overshoot of 11% and rise time of 0.7 s. The 5% settling time is measured as 3.1 s. This time can be lowered but would result in more overshoot. Excessive overshoot results in the UAV oscillating around the target location. This controller provides good disturbance rejection

which can be seen in Figure 5.14b. This controller has a bandwidth of 7.12 rad/s which is separated well enough from the angle controller with a bandwidth of 11 rad/s.



(a) North Step response

(b) North Step With disturbance

Figure 5.14: Velocity controller step response and disturbance rejection

5.8 Position Controller

The position controller is used to control the UAV to a specific position or location measured in the inertial frame and specified in NED coordinates. The position controller commands a velocity reference, $\mathbf{V}_{I,r}$ using a proportional controller. This velocity reference is limited using a non-linear saturation element. The maximum and minimum velocity of the saturation element can be specified using a parameter in PX4. The linear plant of the North positional controller is given as:

$$G_{XN}(s) = G_{VN,cl}(s) \cdot \frac{1}{s} \tag{5.24}$$

Here the position of the UAV is obtained by integrating over the velocity. The closed loop north velocity transfer function $G_{VN,cl}(s)$ is given as:

$$G_{VN,cl}(s) = \frac{D_{VN}(s)G_{VN}(s)}{1 + D_{VN}(s)G_{VN}(s)} \tag{5.25}$$

The block diagram of the position controller and plant is shown in Figure 5.15.

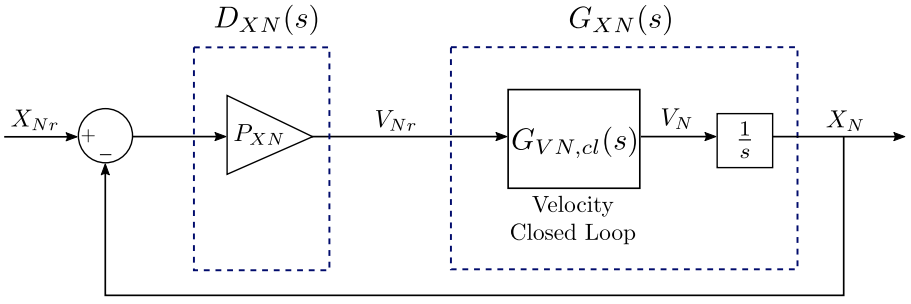
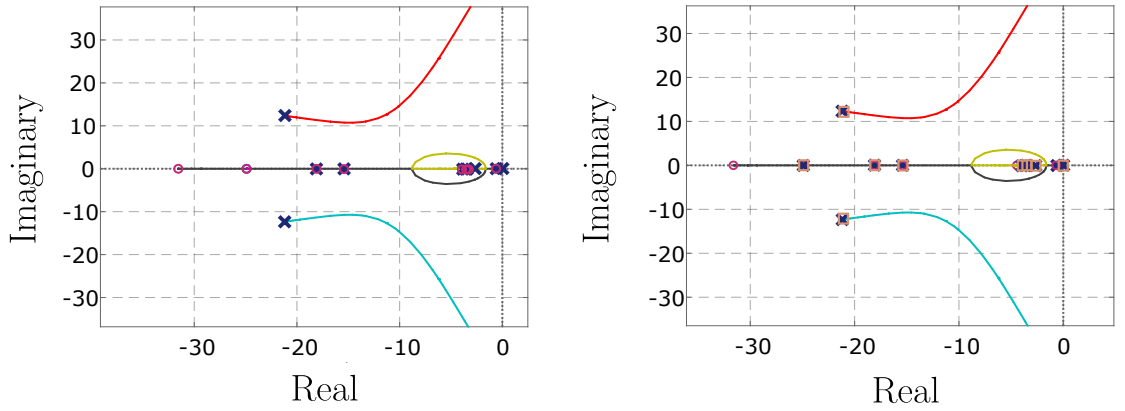


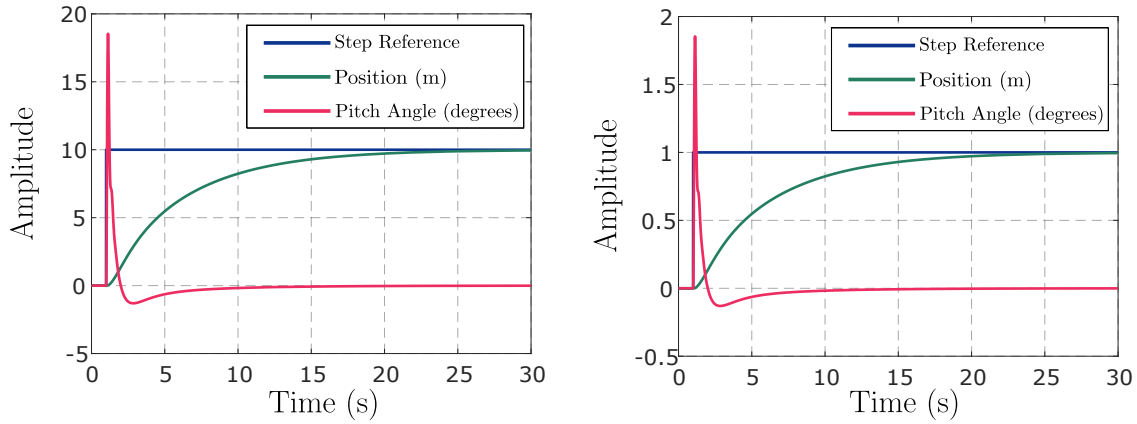
Figure 5.15: North position controller and plant block diagram

It is important to keep pitch and roll angles as small as possible due to the fixed mounted camera setup. This avoids the landing marker from disappearing out of frame and minimises pose estimation error during tracking manoeuvrers. The positional controllers step response must thus be slow in order to command a low linear velocity during translational movement, thereby minimising pitch and roll angles. The default ZMR250 Racers proportional gain resulted in a 2% settling time of 2 seconds. This is too fast as it induces pitch angles of 45°. For large positional steps the maximum angle the UAV is allowed to produce is 20° to preserve image quality. For small positional steps this angle would naturally be much lower. The position controller gain is thus designed together with the pitch angle response. The root locus of the plant dynamics and the plant with controller are first presented in Figures 5.16a and 5.16b. The desired step response of the position and angle controllers are shown in Figures 5.17a and 5.17b.



(a) Position Plant (b) Position Plant and Controller

Figure 5.16: Root locus of position plant and controller



(a) Large Position Step (b) Small Position Step

Figure 5.17: Small and large North position step response

When a large position reference is applied the UAV pitches with a maximum angle of 18°. When a small 1 meter reference is applied the UAV pitches with a maximum angle of 1.8°. This is an ideal response for this project. The position controllers settling time is long and is measured as 20 s. A gain scheduling method could be implemented where the proportional gain of this controller is changed during the flight when the UAV is required to fly faster. Alternatively the onboard camera could be gimbal mounted in order to have a stable image

regardless of the orientation of the UAV. This controller has a bandwidth of 0.186 making it well separated from the velocity controller.

Figure 5.18 shows the bandwidth separation between controllers using the frequency bode plot of each controller.

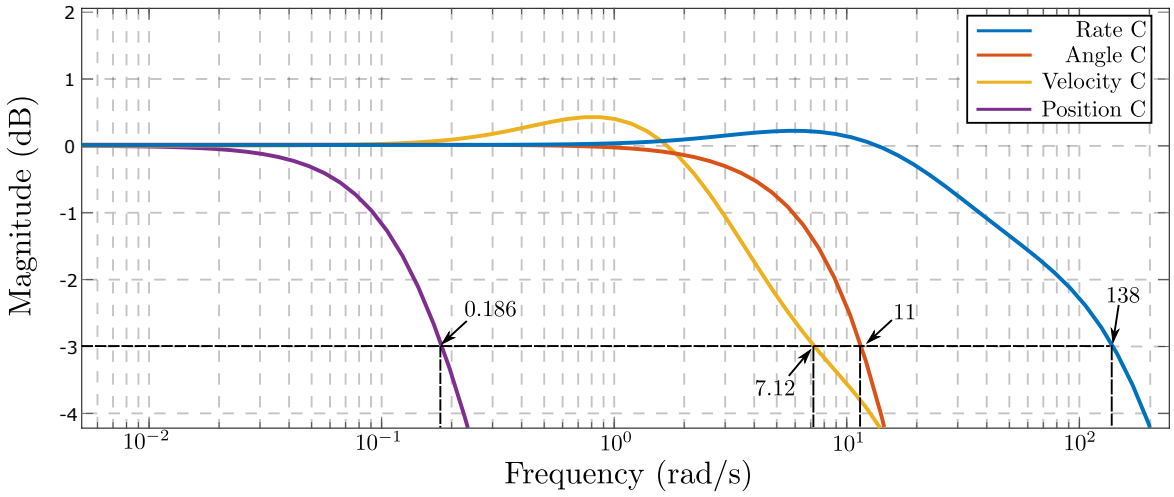
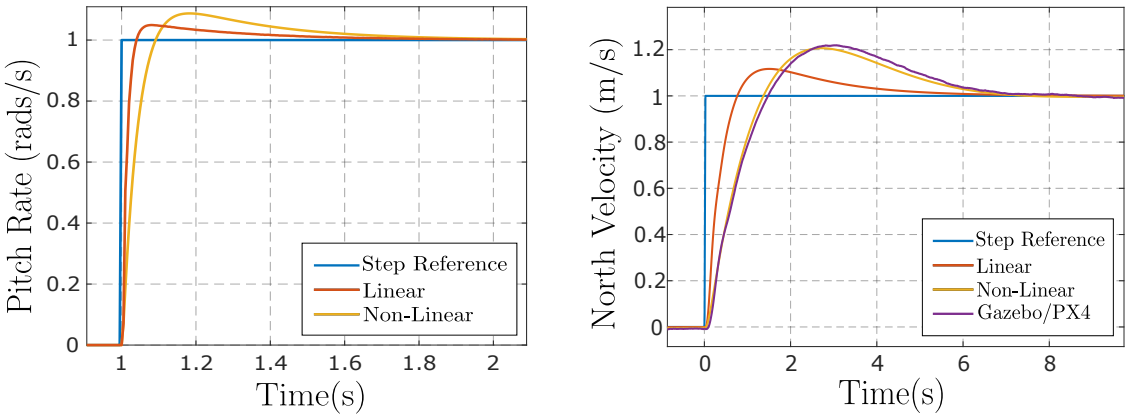


Figure 5.18: Bode plot of each controller showing bandwidth separation

5.9 Simulation

The controllers have thus far been designed and tested with a linearised UAV model. The controllers need to first be tested with a non-linear UAV model before a practical verification test can be attempted. A non-linear MATLAB/Simulink simulation is created for this purpose. In this simulation no sensor noise is added but includes PX4's custom modifications to the control architecture. The same controllers are also tested in a PX4/Gazebo SITL and HITL simulation. With these tests the actual flight software is being executed on a non-linear UAV model. Sensor noise is again excluded, the MATLAB and Gazebo responses should therefore be similar.

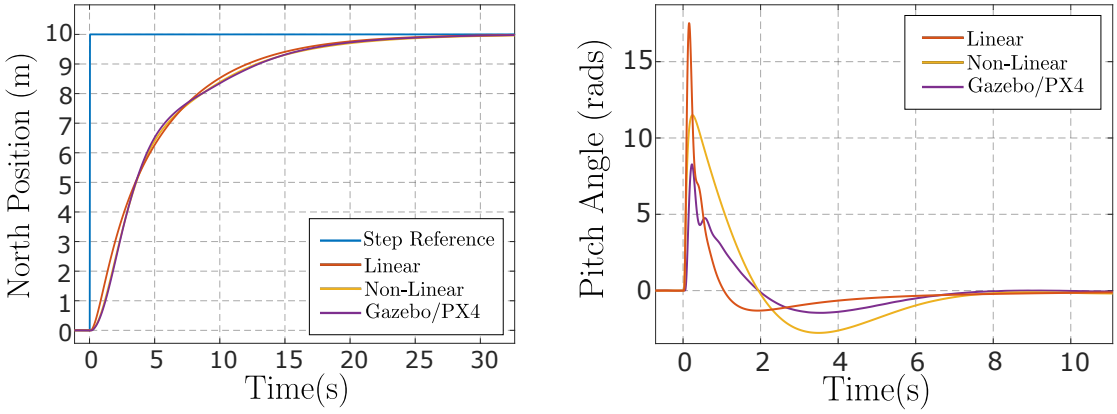


(a) Angular Pitch Rate

(b) North Velocity

Figure 5.19: Non-linear controller verification

Figures 5.19a and 5.19b show the step responses of the angular rate and velocity controllers respectively. In both these figures the responses of the linear and non-linear models are slightly different, however are still acceptable. The response of the non linear MATLAB/Simulink and Gazebo/PX4 simulations show high degrees of similarity, this shows that the PX4 controller architecture is sufficiently modelled.



(a) Large Position Step

(b) Angle Response to large Position Step

Figure 5.20: Non-linear large position step and angle response

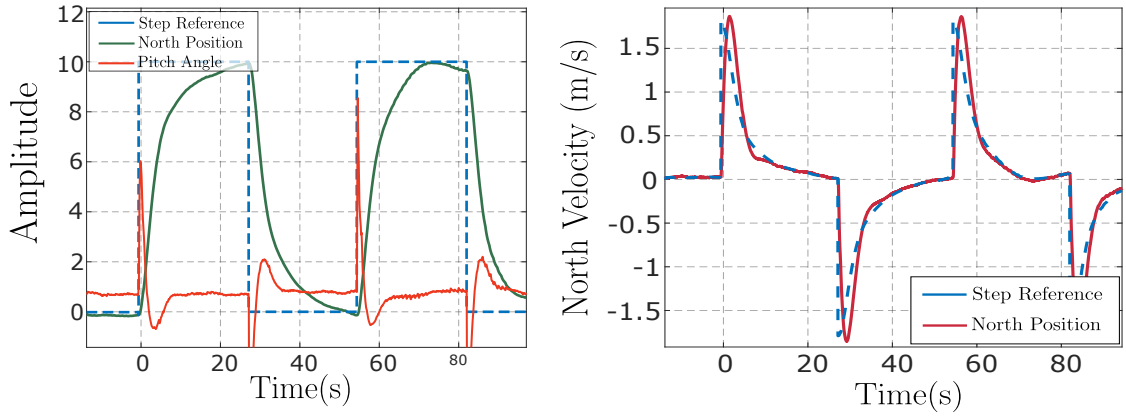
Figures 5.20a and 5.20b show the position and angle controllers response to a large 10 m position step. The main goal was to observe the maximum pitch angle induced during this step. Ideally this needs to be as low as possible. Figure 5.20b shows this maximum angle to be 7.5° in the Gazebo simulation. This is a good result as it would allow for a stable downward facing image during tracking manoeuvrers. Figure 5.20a shows high degrees in similarity between the linear and non-linear MATLAB/Simulink and Gazebo/PX4 simulations. These results are however generated with no added sensor noise which results in an ideal response. The positional controller shows a well damped and slow response which is ideal for stable marker tracking during the autolanding.

5.10 Practical Controller Verification

The designed controllers had to be verified on the practical UAV. To do this the UAV was first flown in stabilised mode. In this mode the manual controller inputs correspond to pitch, roll and yaw angle references. This is used to test the inner angle controllers before testing the external positional controllers. This allows for a safer testing strategy as not all controllers are being tested at once. During this test the UAV was stable and responded well to user input, thus verifying that the inner attitude controllers worked well.

To test the outer translational controller the UAV was flown in position mode which acts on the linear velocity and position controllers. The UAV was again stable and responded well to user input.

A third test was setup where the OBC was responsible for publishing positional setpoints to the flight control unit (FCU). A waypoint scheduler ROS node is used to do this. The results of this test are shown in Figure 5.21.



(a) North position and pitch angle response (b) North velocity response

Figure 5.21: Practical 10m position step response

The practical position step response of the UAV closely matches simulated results. The maximum pitch angle induced during a 10 m step is measured at 8.3°. This is a good result that should minimise pose estimation errors while performing tracking manoeuvrers of the landing target. The position step response is slow with a maximum velocity of 1.5 m/s and average velocity of 0.5 m/s. This project does not require the UAV to fly faster than this, however a gain scheduling strategy could be implemented that allows the UAV to fly faster if required.

5.11 Summary

In this chapter the four controllers, namely: angular rate, angle, velocity and position controllers were designed using a linear UAV plant. The longitudinal and lateral controllers are equal due to an approximation of the UAV's symmetry, thus in this chapter only the longitudinal component was discussed.

PX4 provides a selection of pre-designed controllers for a variety of existing UAV airframes. In this chapter controller gains from a UAV called the ZMR250 Racer was used as a starting point for designing custom controller gains suitable for this projects requirements. During the design process it was decided not to change the angular rate controller gains as these provided an already suitable and stable response with good disturbance rejection . The angle, velocity and position controllers were adjusted for this project, ultimately resulting in a slow translational response which minimised pitch and roll angles. This was done to improve the onboard cameras object tracking ability during the autolanding.

After designing the linear controllers in MATLAB/Simulink the same controllers were verified on a non-linear MATLAB/Simulink model as well as in Gazebo with the actual flight software in a SITL and HITL simulation. The linear and non-linear responses differed slightly but still gave acceptable results.

These controllers were then tested on the practical system in three separate tests. The results from all three practical tests confirmed that the UAV's custom controllers work well in practise. The UAV was stable and responsed well to user input. These controllers are well suited for the purpose of performing visual based autolandings with the current UAV system.

Chapter 6

Autolanding

In order for the UAV to make contact with the recharging station it first needs to complete an accurate autolanding prior to charging. Traditional GNSS cannot be used to localise the UAV during landing as the technology lacks the required accuracy. The UAV must instead make use of an onboard camera and a recognizable visual target located on the charging station to accurately localise and land. An onboard computer (OBC) will be responsible for processing the images and relaying the extracted pose to the flight controller for accurate control. The autolanding system will first be tested in a Gazebo SITL and HITL simulation environment before being implemented on the practical UAV. In this chapter the state machine design used for the autolanding is presented after which three landing target designs are tested in simulation. The most effecting design is chosen and finally tested with the practical landing system.

6.1 Autolanding State Machine

A state machine based landing approach was chosen and designed in order to have greater control of how the UAV approaches the landing target. The state machine consists out of four states, each successive state decreases the altitude of the UAV while applying the appropriate control law. General safety and robustness to unexpected anomalies must be included in state design. Each state thus includes continuous fault detection and is capable of triggering a repair or bail out state. The applied positional control law is changed during the final stages of the landing. This transition is dependant on the current altitude and the visual feedback reference that is available, this is discussed in more detail later in the chapter. The state machine approach allows for changes in one area of the landing process without affecting others. For this section please refer to Figure 6.1 which provides an illustration and flow diagram of the state machine used.

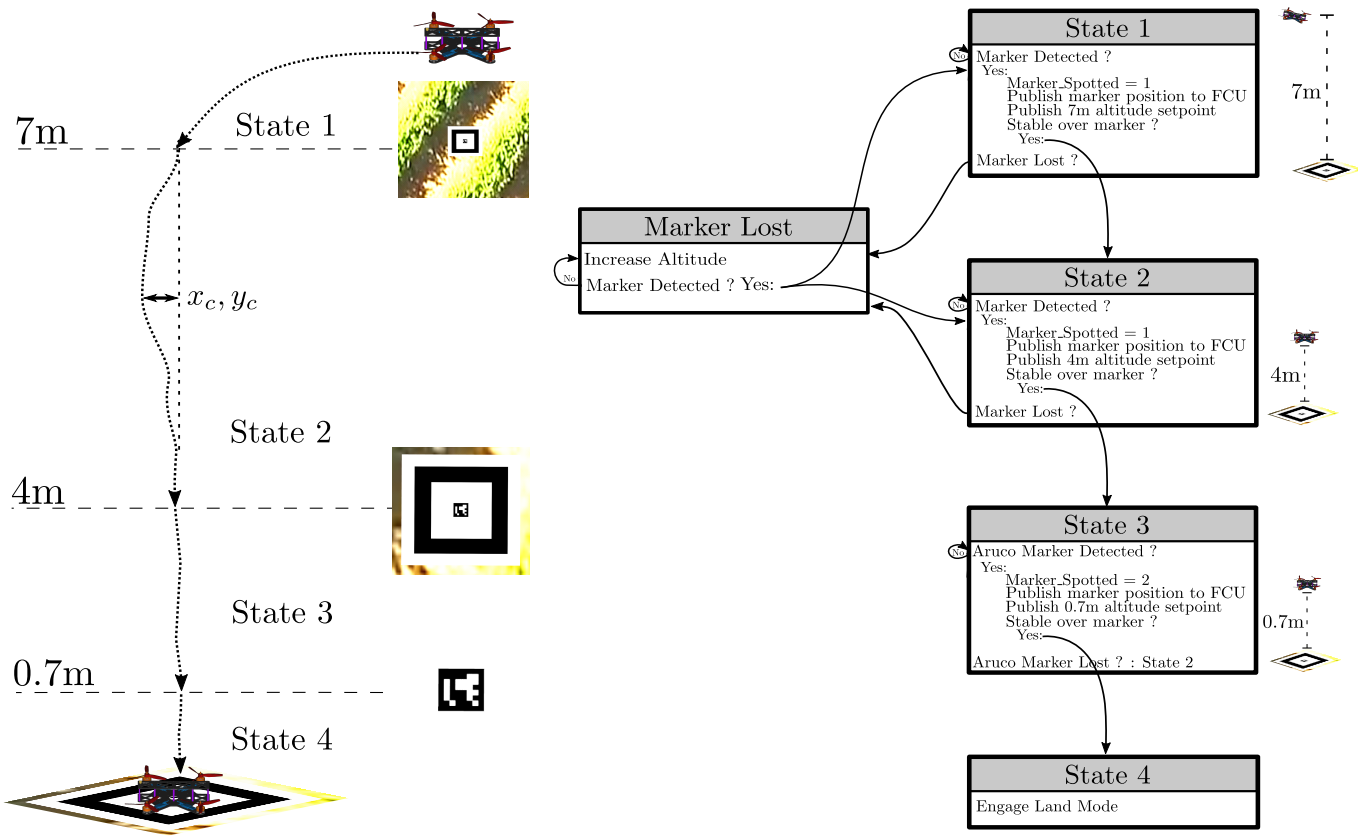
6.1.1 Initial High Altitude Approach - State 1

The first state assumes that the UAV has been sent the known GNSS coordinate of the recharge station. The UAV is therefore positioned at an appropriate altitude with some radial offset from the marker due to the limited accuracy of the onboard GNSS. Given this offset which is assumed to be at worst 4m at an altitude of 7m or higher and given the field of view of the camera, the UAV would be able to identify the landing marker. Once the marker has been successfully detected, the corresponding x and y position of the UAV relative to the centre of the marker is calculated and published to the (Flight Control Unit) FCU for positional control. This control law is activated by setting a custom MAVLink message called `marker_spotted` equal to 1. When this control law is active the UAV uses visual pose data

for control instead of GNSS data but still relies on barometer data for altitude feedback. The UAV has a separate process that keeps track of the elapsed time since the last detection of the marker. This will never be non-zero as there is always delays due to processing constraints. If the elapsed time is greater than 3 seconds, the marker is assumed to have been lost. This triggers a repair state where the altitude is increased until the UAV is able to regain visual contact with the marker. If this is unsuccessful in repairing the autolanding the bail out state is initialised which will resume GNSS based control and fly the UAV back to the known GNSS location of the recharge station and retry the landing. In order to proceed to the next state the UAV would need to meet the following stability constraints:

$$
\begin{aligned}
&x_c < threshold, \\
&y_c < threshold \\
&x\_vel < threshold \\
&y\_vel < threshold \\
&abs(z_I) >= 7
\end{aligned}
$$

Where x_c and y_c are the relative x and y positions of the marker, x_vel and y_vel are the x and y velocities of the UAV, z_I is the estimated altitude of the UAV and $threshold$ is a fixed constraint.



(a) State Transitions (b) State Machine Flow Diagram

Figure 6.1: State machine flow diagram and illustrated landing process

6.1.2 Decrease Altitude to ArUco Marker - State 2

The second state is similar to the first as it also uses control mode 1 as determined by the `marker_spotted` flag. This state also uses the same method for landing repair. The goal for

this state is to lower the UAV's altitude to the point where the inner ArUco marker is clearly detectable. This allows the primary localisation transition from outer square to inner ArUco to be made. The optimal altitude at which to perform this transition was identified to be 4m as discussed in Section 6.7.2. The state transition requirements are:

$x_c < threshold$
$y_c < threshold$
$x\_vel < threshold$
$y\_vel < threshold$
$abs(z_I) < 4.2$
`marker_ID` $== 1$

Here `marker_ID` is a unique ID given to each ArUco marker and is determined upon detection. The ID is used as verification that the ArUco marker is successfully detected.

6.1.3 Switch to ArUco Marker for Localisation - State 3

State 3 uses the inner ArUco marker for positional feedback and control in not only the x and y directions but also in z measured in the body frame of the UAV. During this state the UAV is required to hover 0.7 m above the marker. Maintaining an accurate and stable altitude is critical in this stage of the landing as the UAV is now very close to the ground. From initial experiments it was clear that the barometer was not effective in controlling the UAV to an altitude of 0.7m. This is largely due to the pressure change created by the rotors of the UAV at this low altitude. Thus the UAV cannot rely on the inaccurate GNSS and barometric altitude estimate. By setting the `marker_spotted` flag equal to 2, the control law is switched on the FCU. The UAV uses the same landing repair method discussed in state 1 and 2. The requirements for state transition are:

$x_c < state3\_threshold$
$y_c < state3\_threshold$
$x\_vel < state3\_threshold$
$y\_vel < state3\_threshold$
$abs(z_c) < 0.8$

Where $state3\_threshold$ is a tighter constraint than $threshold$ and z_c is the visual z component of the extracted marker pose.

6.1.4 Switch to Land Mode - State 4

Once the UAV reaches this state it is assumed that the UAV is now hovering accurately directly above the marker. At this moment the UAV can make use of PX4's land mode which is capable of autonomously landing and disarming the UAV at its current position. This works well provided the UAV is as close as possible to the landing target.

6.2 Autolanding Controller

The autolanding controller is an adaptation of the default positional controller used by PX4. The default positional controller uses EKF filtered GNSS data transformed to the local world frame (NED) as feedback. The autolanding controller replaces this data with the extracted relative x_c, y_c and z_c position of the UAV to the marker centre. When each of these components are used is determined by the state of the `marker_spotted` flag. This allows for the active control law to be changed easily from the ROS system running on the OBC. Three

required control modes were identified. Mode 0 uses the default control law (GNSS). Mode 1 replaces only the x_I and y_I position components in the feedback loop and mode 2 replaces x_I, y_I and z_I components in the feedback loop. Mode 2 is used where more accurate altitude control is required such as in State 3. In both modes 1 and 2, the position controller's reference vector, $\mathbf{X}_{I,r}$ is set as follows:

$$\mathbf{X}_{I,r} = (0, 0, z_{I,r}) \tag{6.1}$$

Where $z_{I,r}$ is the altitude reference published to the FCU from the OBC. Setting both the $x_{I,r}$ and $y_{I,r}$ components to 0 makes the UAV control its position so that the centre of the onboard camera's image frame is aligned with the centre of the target. During autolanding the yaw angle of the UAV is set to 0. This means that the body frame of the UAV is perfectly aligned with the world frame. This excludes the need for any transformations but does more heavily rely on a magnetometer. From testing this was considered sufficient for accurate control. Figure 6.2 illustrates the three different control modes.

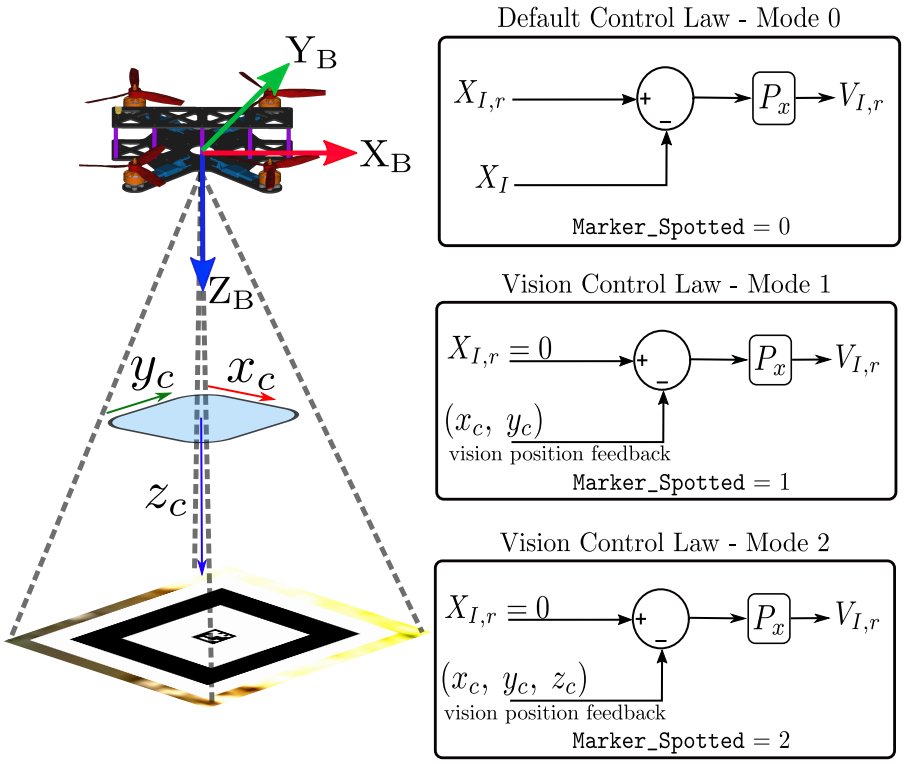


Figure 6.2: Three different positional controllers used

6.3 Simulation Setup

The autolanding system was designed and tested within a Gazebo simulation environment. Gazebo is able to generate realistic onboard camera images for the purpose of accurately simulating a visual based landing. The various components of the autolanding simulation are now briefly discussed.

6.3.1 UAV Model

An SDF file fully describing the UAV model was created. This file specifies the mesh file used to generate the physical model of the UAV. It also includes a description of what collision model to use. In order to simplify the simulation and increase performance a simple box placed around the UAV was used as the collider model. The file specifies the intrinsic parameters of the UAV. Sensor plugins describing noise models of the UAV's various simulated sensors are also included. The simulated camera is added to the UAV using a fixed link.

6.3.2 Camera

The simulated camera is placed on the underside of the simulated UAV model. Gazebo allows for noise and distortion to be added to the simulated images. The effect of these parameters on autolanding performance is investigated in Section 6.6. The resolution of the simulated images are scaled to the actual value of 640 x 480p. Although the actual camera used is capable of outputting higher resolutions, the best computational performance was achieved with this resolution.

6.3.3 World

To complete the simulation setup a world or environment in which the UAV can fly is created using a world file. The world file is not only responsible for creating the instance of the UAV but also the different markers that are tested in this chapter. A ground plane is also created to add more realism and test the robustness of the detection algorithm. Here lighting can be changed and the physics solver is specified. Figure 6.3 shows the created world.

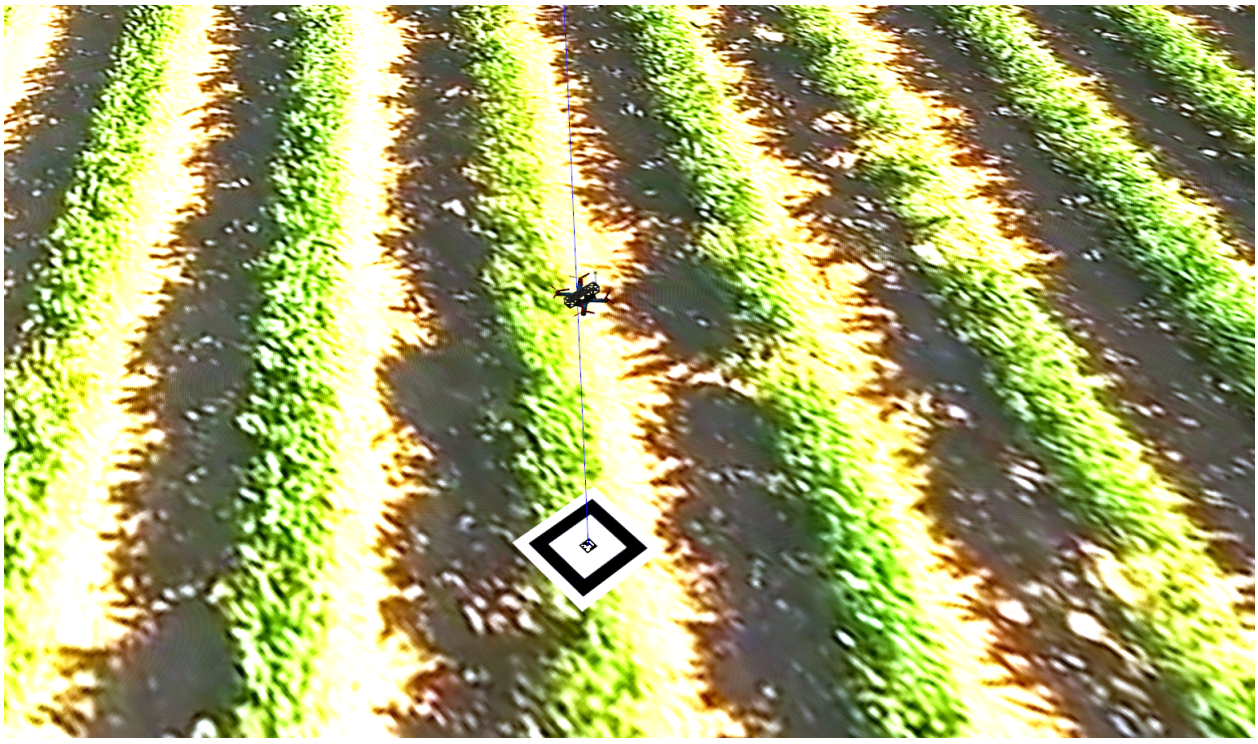


Figure 6.3: Simulated UAV and world

6.3.4 Custom Message

A custom MAVLink/MAVROS message had to be created for the autolanding. This message contains the extracted pose of the marker $(x_c, y_c, z_c,)$ as well as the `marker_spotted` controller flag. This message was not only used in simulation but is also included in the practical flight software.

6.4 Camera Calibration

Camera calibration refers to the process by which the intrinsic parameters of the camera are determined. The intrinsic parameters refer to the lens and image sensor characteristics which need to be determined in order to correct for distortion in the image. Image distortion can have a large effect on the marker pose estimation accuracy and therefore needs to be removed.

6.4.1 Camera Model

For this project a basic pinhole camera model is used to calculate the projection of a point located in 3D space onto a 2D image plane [54, p.153]. Figure 6.4 shows a representation of the pinhole camera model.

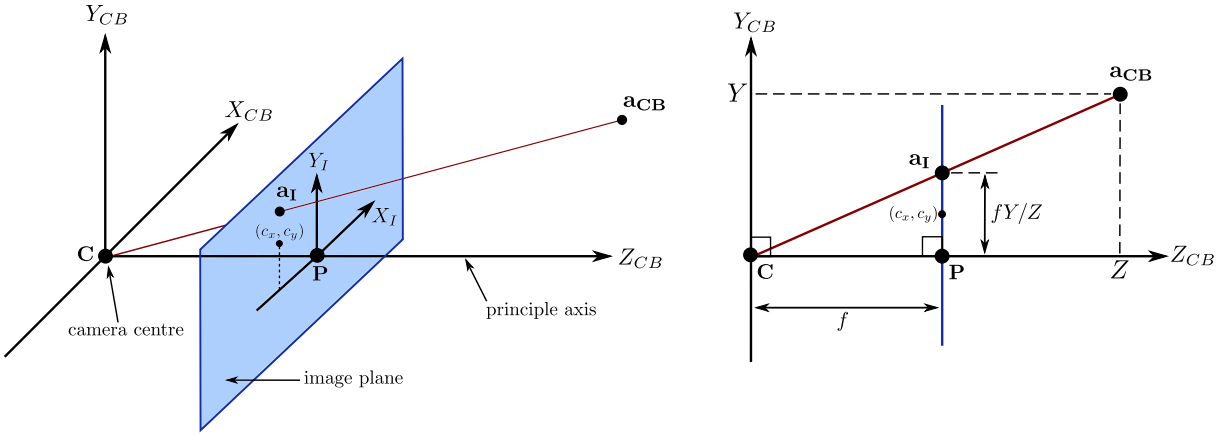


Figure 6.4: Pinhole camera model

Here let the camera axis system labelled with subscript CB be a Euclidean coordinate system with its centre located at the camera centre C. The plane $Z_{CB} = f$ is called the image plane and is parallel to the camera axis system. The point where the principle axis system intersects the image plane should be in the middle of the image plane labelled P. The relative offset of the image plane and camera centre is defined as C_x and C_y. The distance between the camera centre and the image plane is the focal length and is labelled f. The 3D point in space with coordinates $a_{CB} = (X, Y, Z)^T$ is projected onto the image plane with a line joining the point a_{CB} and the camera centre. This point of intersection with the image plane is labelled a_I. Using similar triangles the point a_I can be calculated using:

$$(X, Y, Z)^T = \left(f\frac{X}{Z} + c_x \; f\frac{X}{Z} + c_y \; f \right)^T . \tag{6.2}$$

Equation 6.2 describes the mapping of the 3D real world coordinates of the point a_{CB} to the 2D image plane coordinates $a_{I,x}$ and $a_{I,y}$. This equation can be rewritten in homogenous vector form using:

$$\mathbf{A}_I = \begin{bmatrix} \mathbf{X}_I \\ \mathbf{Y}_I \\ \mathbf{Z}_I \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_I \\ y_I \\ z_I \\ 1 \end{bmatrix}, \tag{6.3}$$

which can also be written as

$$\mathbf{A}_I = \mathbf{P}_c \begin{bmatrix} x_I \\ y_I \\ z_I \\ 1 \end{bmatrix}, \tag{6.4}$$

where

$$\mathbf{P}_c = \mathbf{K}_c[\mathbf{I}|0] \tag{6.5}$$

$$\therefore \mathbf{K}_c = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6.6}$$

Here $\mathbf{P}_c$ is the camera projection matrix and $\mathbf{K}_c$ is the camera calibration matrix. In order to account for rectangular and parallelogram shaped pixels the camera calibration matrix needs to include a scaling and skewing factor. The calibration matrix $\mathbf{K}_c$ is therefore:

$$\mathbf{K}_c = \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & s & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} fm_x & s & c_x m_x \\ 0 & fm_y & c_y m_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{6.7}$$

where m_x and m_y are scaling factors in the x and y direction and s is the skewing factor.

6.4.2 Calibration

It has been assumed that the camera matrix $\mathbf{P}_c$ is known. However, in practice this must be calculated using a set of n number of 3D points in space and their corresponding 2D representations [54, p.88]. Let the set of 3D points be $\mathbf{A}_{I,n}$ and their corresponding 2D representations be $a_{I,n}$. From here Equation 6.4 can be used to calculate the correspondence of $\mathbf{A}_{I,n}$ to $a_{I,n}$ using:

$$a_{I,n} = \mathbf{P}_c \cdot \mathbf{A}_{I,n} = \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix} \cdot \mathbf{A}_{I,n}, \tag{6.8}$$

where P_i^T is a 4-vector, the i-th row of $\mathbf{P}$. Note that this equation involves homogenous vectors, thus $a_{I,n}$ and $\mathbf{PA}_{I,n}$ are not equal, they have the same direction but their magnitudes differ. The equation can therefore be written as the vector cross product $a_{I,n} \times \mathbf{PA}_{I,n} = 0$. This will allow for a linear solution for $\mathbf{P}$ to be calculated. Writing $a_{I,n} = (x_{I,n}, y_{I,n}, w_{I,n})$, equation 6.8 can be expanded as

$$\begin{bmatrix} 0^T & -w_{I,n} \cdot A_{I,n}^T & y_{I,n} \cdot A_{I,n}^T \\ w_{I,n} \cdot A_{I,n}^T & 0^T & -x_{I,n} \cdot A_{I,n}^T \\ -y_{I,n} \cdot A_{I,n}^T & x_{I,n} \cdot A_{I,n}^T & 0^T \end{bmatrix} \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix} = 0, \tag{6.9}$$

since the first two rows of equation 6.9 are linearly dependant it can be simplified as

$$\begin{bmatrix} 0^T & -w_{I,n} \cdot A_{I,n}^T & y_{I,n} \cdot A_{I,n}^T \\ w_{I,n} \cdot A_{I,n}^T & 0^T & -x_{I,n} \cdot A_{I,n}^T \end{bmatrix} \begin{bmatrix} P_1^T \\ P_2^T \\ P_3^T \end{bmatrix} = 0. \tag{6.10}$$

The next question is how many n points are required to calculate $\mathbf{P}$. The lower bound of this number considers the number of degrees of freedom and the number of constraints. Since the pinhole camera model has 11 degrees of freedom: 6 from the rotation and translation of each pixel and another 5 from the calibration matrix. The number of constraints is two per point-to-point correspondence. Thus the number of points required is 5 and a half, therefore 6 to fully constrain $\mathbf{P}$ [54].

The camera used for this project was calibrated using an OpenCV calibration script and a chess board pattern. The calculated camera matrix could then be used by the ArUco marker detection algorithm using the `camera_info` ROS topic.

6.4.3 Distortion

Due to the curved shape of the camera lens the incoming light rays are distorted. The projection of the 3D point a_{CB} to the corresponding image plane coordinate a_I is assumed to be linear and thus the non-linear distortion needs to be removed. The distortion is relatively small at the centre of the image but increases as you move closer to the edges of the image. The distortion factor $L(r)$ can be approximated with the Taylor expansion:

$$L(r) = 1 + k_1 r + K_2 r^2 + k_3 r^3 + ... \tag{6.11}$$

Where r is the distance of the distorted point to the centre of distortion (x_{0D} and y_{0D}) which is usually close to the principle point. The same OpenCV script used for camera calibration was also used to solve for the distortion parameters ($D = x_{0D}, y_{0D}, k_1, k_2, k_3$). This concludes all parameters needed for the camera model and accurate image processing.

6.5 Marker Design

The next step is to design the most effective marker for the autolanding. Due to the realism Gazebo offers with respect to generated camera images and accurate physics, the autolanding can be tested and confirmed in simulation. This offers a confident transition to the practical system. The following section focuses on three different landing markers that were designed and tested. Advantages and disadvantages of each landing marker are analysed allowing for the best design to be chosen. The different marker requirements are summarised as follows:

- Marker must have smallest possible form factor.
- Marker must be distinguishable from external environment making it recognizable at high altitude.
- Highly visible with high contrast levels.
- Pose estimation must be computationally efficient.
- Provide landing accuracy under 10 cm.

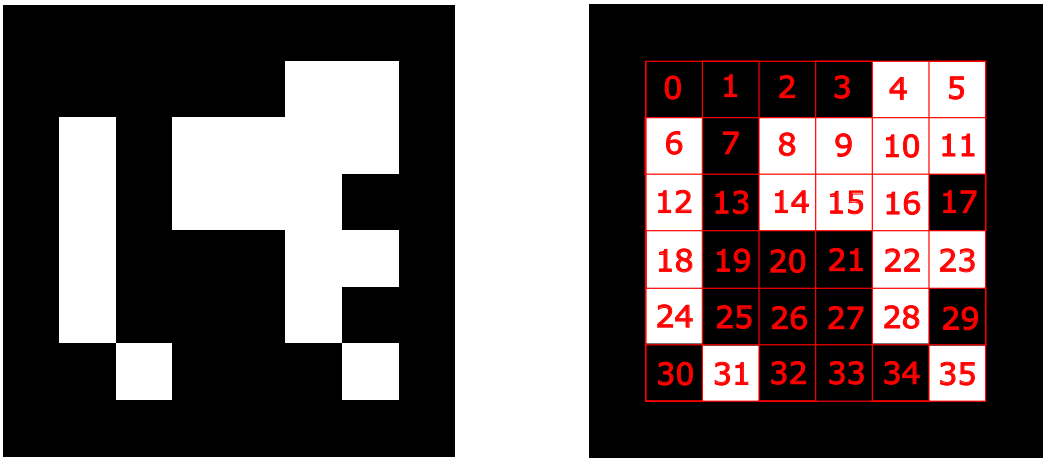
Simulated autolanding performance of each different marker design is presented in the following subsections.

6.5.1 ArUco Marker

The first marker that is tested consists of just one large ArUco marker. ArUco markers are commonly used in augmented reality systems due to its ability to accurately estimate the 3D pose of a camera in space. This feature can be used for this project. Their unique designs and high levels of contrast makes them highly distinguishable from any natural or unnatural

background. Their detection and pose estimation algorithms are well optimised thus making them highly suitable for low powered computing devices such as onboard computers.

The ArUco library contains four different marker dictionaries. The difference between dictionaries is simply the number of bits used for encoding. Figure 6.5 shows how the inner binary matrix is used for encoding a unique id to each marker within its dictionary.



(a) ArUco Marker (b) Inner binary matrix

Figure 6.5: ArUco marker encoding method

Each block in the grid can either be black or white which represents a 0 or 1 respectively. The marker does not only contain data bits but also has parity bits encoded in the grid. The parity bits are necessary for robust detection. This is why decoding does not directly relate to the decimal position of the bits in the grid. The markers are also asymmetrical which allow for orientation of the marker to also be determined.

Detection

A popular ArUco detection ROS package called `aruco_detect` [65] was installed and used for detection of the markers. This package is based on the ArUco contributed module to OpenCV. Detection of the ArUco markers are done in two main steps: Identify all potential marker candidates and decode markers to determine the marker ID. Adaptive thresholding is used to extract all contours in the image. From here all non-convex and square shapes are discarded. After this an extra filtering step is used that removes markers that are too small or too big. This leaves only a list of potential valid markers.

The next step is to decode all potential markers to find their respective ID's. To do this a perspective transformation is applied to obtain the canonical form of the marker. This is used to remove the rotation between the camera and the marker to get it to its standard form. The white and black bits are separated using an algorithm which firstly divides the marker up into cells based on the known marker size and the perceived marker size. The white pixels in each cell are counted to determine whether the cell is black or white. The marker ID can then can be determined from the extracted bits.

Pose Estimation

For pose estimation the intrinsic parameters of the camera must be known. This is determined during the camera calibration process. Once this is known the `solvePnP` function within OpenCV can be used to determine the relative pose of the camera to the marker. The method finds the translation and rotation between the 2D image plane coordinates of an object and the 3D coordinates of that object. The relationship between these two can be expressed as

$$\mathbf{a}_I = \mathbf{K}_c[\mathbf{R} \;\; \mathbf{T}_{vec}]\mathbf{A}_I, \tag{6.12}$$

this can be expanded as:

$$\begin{bmatrix} ax \\ ay \\ 1 \end{bmatrix} = \begin{bmatrix} fm_x & s & c_x m_x \\ 0 & fm_y & c_y m_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}, \tag{6.13}$$

where $\mathbf{T}_{vec}$ is a 3x1 translation vector and $\mathbf{R}$ is a 3x3 rotation matrix. This is very similar to Equation 6.5 but here $\mathbf{P}_c$ is replaced with $\mathbf{K}_c[\mathbf{R}]\mathbf{T}_{vec}$. This allows for translation and rotation of the camera axis. The AruCo marker detection process extracts the four corner pixel positions of the marker projected onto the image plane. The 3D and 2D points are then transformed by Equation 6.12. This gives a 3x1 rotation vector ($\mathbf{r}_{vec}$) and a 3x1 translation vector ($\mathbf{t}_{vec}$). OpenCV requires a quaternion representation of the calculated angle. Equations 4.1 can be used to do this which results in the quaternion,

$$\bar{\mathbf{q}} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}, \tag{6.14}$$

where $\mathbf{r}_{vec}$ is transformed into a 3x3 rotation matrix ($\mathbf{R}$). The pose matrix ($\mathbf{M}$) is then created with the rotation and translation matrix using

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t}_{vec} \\ 0 & 0 \end{bmatrix}. \tag{6.15}$$

6.5.2 Simulation Results with ArUco Marker

The simulation using a large 0.6 x 0.6 m ArUco marker was placed in the Gazebo world as shown in Figure 6.6. The marker had to be at least this large for reliable detection at high altitude (7m). Figure 6.6 shows the UAV tracking the marker during a simulated landing. The top left image shows the ArUco detection from the UAV's onboard camera perspective.

After the UAV was able to complete consistent landings, the UAV was tasked to autonomously take off and repeat the landing, this allowed for 150 autolandings to be completed. The exact landing location was recorded after each landing. The combined results of these landings are shown in Figure 6.7.

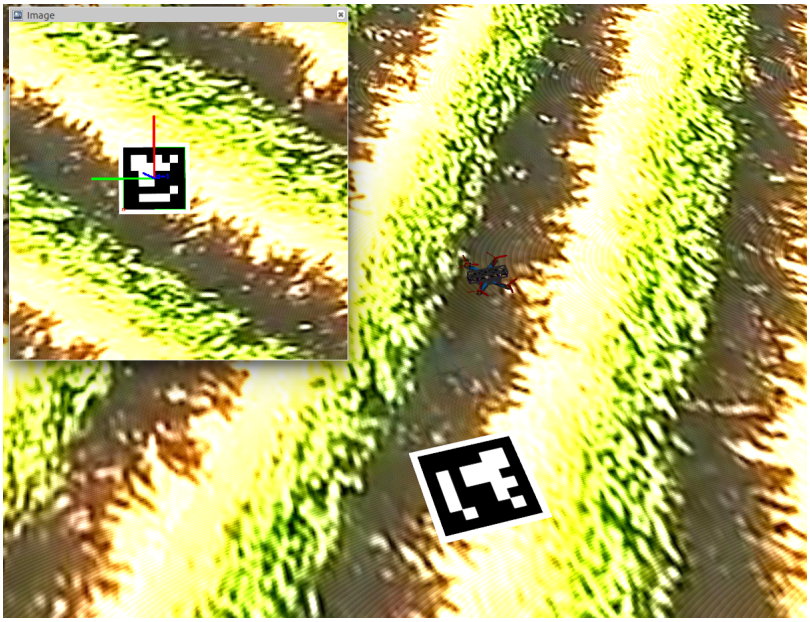


Figure 6.6: ArUco marker landing simulation showing ArUco marker detection

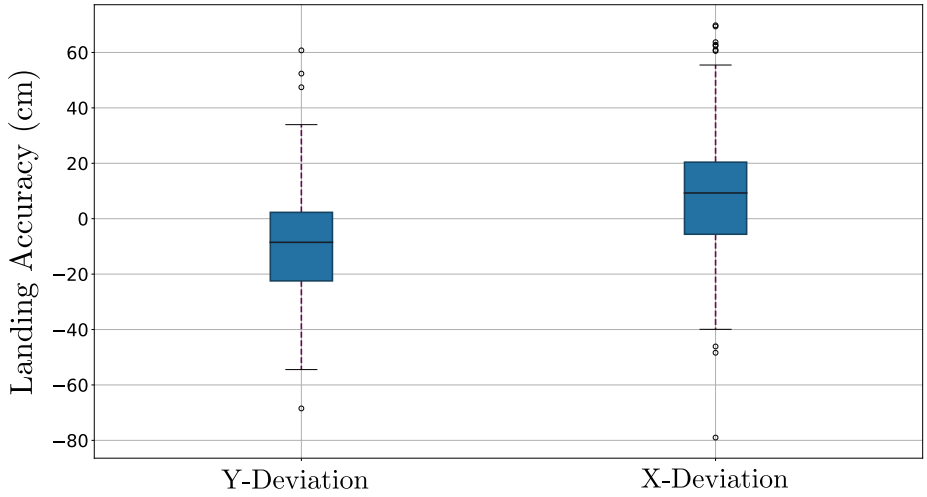


Figure 6.7: Simulated ArUco marker autolanding results

The results presented in Figure 6.7 show worst case landing accuracies of 80cm. This level of inaccuracy is not acceptable for this project. The UAV is required to make direct contact with contact pads on a charging platform, therefore the landing accuracy would need to be improved.

The landings were all carried out from the same starting position and with zero noise on the onboard camera images in order to isolate the marker as the only cause for inaccuracy. The onboard camera has a limited field of view which would cause the marker to become partially excluded from the image during low altitude flight. This places a lower limit on the controllable altitude range. Ideally this limit needs to be as small as possible. The state

machine is designed for the UAV to descend to just outside of this limit before initiating land mode (State 4). This limit is dependant on the size of the marker and can be lowered by decreasing the size of the marker. This would theoretically improve landing accuracy as the UAV would be closer to the landing target before the final stage of the landing. It is not possible to completely eliminate this constraint as the marker needs to remain sufficiently large to stay detectable at altitude. A trade off exists between marker size for sufficient detection ability and the final stage landing altitude.

One possible solution is to add a large external marker and place a small ArUco marker at the centre. When the UAV is at the initial high altitude of 7m, the larger external marker will be used for localisation. At some point during the autolanding the UAV would switch over to the small inner ArUco marker and complete the autolanding.

6.5.3 External Circle Marker

The next type of marker that was designed and tested makes use of a large external circle with an ArUco marker at the centre as shown in Figure 6.8. Circles have been used before in other vision based autonomous landing projects. In [50] a landing marker consisting of concentric white circles on a black background is used for visual based autolandings. Each circle could be uniquely identified through estimating the ratio between each circles inner and outer borders. This method allowed for detection at far and near distances from the marker. In [52] a marker consisting of the letter 'H' located inside a black circle is presented. This allowed for pose estimation of the marker at low, as well as high altitudes.

A ROS based circle detection package was used to extract the relative pose of the circle in a very similar simulation setup as before with only the ArUco marker. The circle marker was detected reliably, however detection rate was low. The maximum rate achieved was measured at 5Hz running on a PC with good graphics hardware. These rates are still sufficient for position control of the UAV in simulation. The GNSS has an update rate of about 1Hz, making circle detection comparatively fast. Running the same circle detection package on a less powerful companion computer would be problematic as detection rates would drop significantly. It would therefore be necessary to explore alternative marker designs. It is still however valuable to determine how the combined internal and external marker design effected landing accuracy before proceeding.

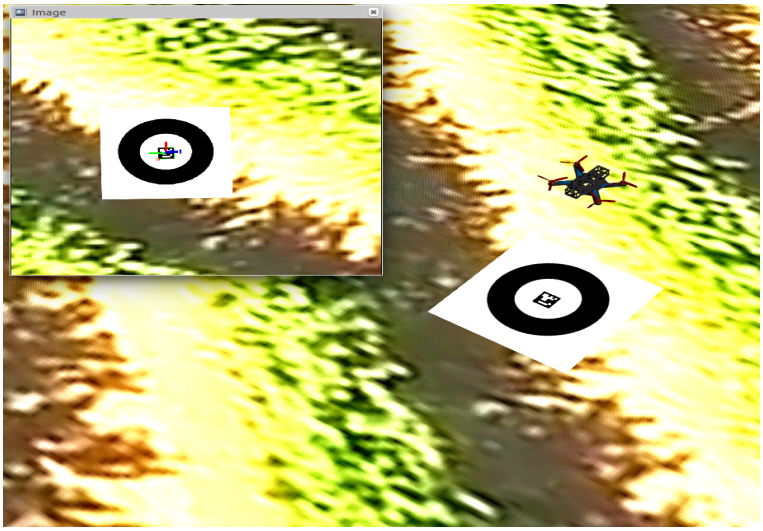


Figure 6.8: Circle marker simulation showing marker detection

6.5.4 Circle Marker Simulation Results

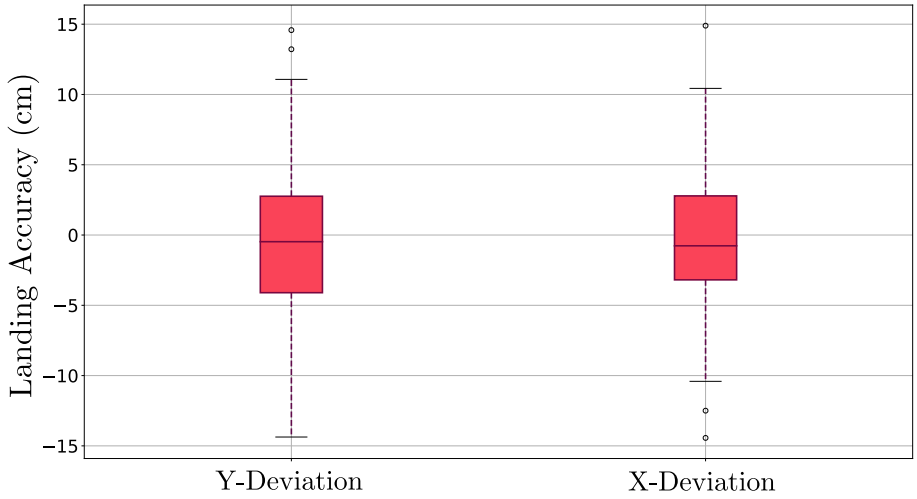


Figure 6.9: Simulated circle marker landing results

Figure 6.9 shows the combined results of 150 landings. Landing accuracy has been greatly improved to a maximum of 15 cm but most landing within 5 cm from the marker. This confirms the hypothesis that landing accuracy can be improved by decreasing the size of the inner ArUco marker, thereby allowing the UAV to get closer to the landing target. These landing results can be compared to differential GNSS based systems but can be implemented at a fraction of the cost. The next step is to redesign the outer marker that will result in less computationally expensive detection and increased detection rates. Redesigning the outer marker should theoretically have no impact on landing accuracy as landing accuracy is only dependant on the size of the inner marker, which will remain unchanged.

6.5.5 External Square Marker

The next type of marker that was considered uses an external black and white square, replacing the circle in the previous marker design. Figure 6.11 shows the UAV tracking the square marker. Markers using black concentric squares have been used before in previous ESL projects at Stellenbosch University [51]. Squares are simple to detect using well known OpenCV functions and techniques. This should improve the detection rate of the external marker while preserving landing accuracy. A ROS package was developed that would be responsible for subscribing to the UAV's onboard camera image topic, processing the images to detect the square marker and publish the relative pose of the marker to the flight controller and state machine.

Detection

The detection process of the square marker is done in two steps. The first step is to effectively extract all contours in the image, prioritising contours belonging to the square marker. These contours are used in the second filtering step. Extracting the contours begins by first converting the images to greyscale after which the images go through a thresholding process. This converts the images into a form consisting only of black and white pixels which is necessary for effective contour extraction. The thresholding process is done in such a way as to

remove most other unwanted background features from the image. This helps in preserving detection robustness and processing cost. Making the marker consist out of black and white pixels only, helps making the marker distinguishable in the thresholding step. Once the contouring process is complete, a filter responsible for extracting the contour belonging to the external square landing marker can be applied.

The filter uses three parameters unique to each contour, namely: aspect ratio, number of edges in each contour and the contour's pixel area. For a square the expected aspect ratio between its edges will always be equal to 1 and it is expected to have only 4 edges. These two parameters alone are not enough to guarantee detection robustness, the contour area must also be used. The perceived area of the square contour in the image plane is directly coupled to the distance between the camera and the marker. If the exact relationship between the area and distance from the marker is known then this can be used to determine the current expected area of the square contour. For this application the current altitude of the UAV can be used for the distance between the camera and the marker. An experiment was setup to measure the pixel area of the marker at fixed distance intervals. The exact relationship is shown in Figure 6.10.

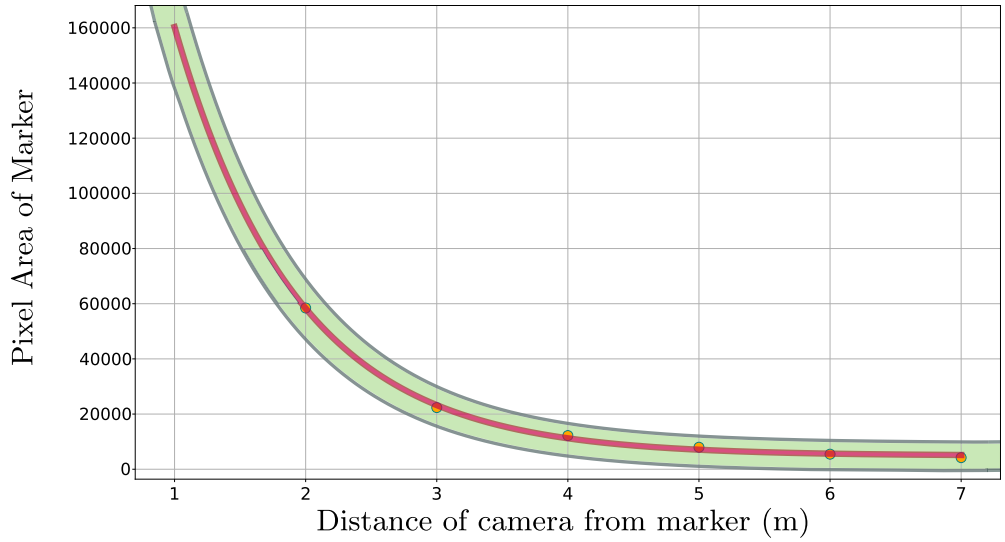


Figure 6.10: Filter profile of square marker contour area vs camera distance

The red line in Figure 6.10 represents the curve that was fitted to the measurements in the above mentioned experiment. The green area represents a safety margin for the expected value of the marker area. This is used to compensate for non ideal fluctuations in the practical measurements such as: vibration and image noise. Any contour that falls within this region is accepted as the square marker. The barometric altitude measurement is used to update the new expected value of the marker area at each timestep, unless the ArUco marker is detected, in which case the z_c pose component is used for a more accurate altitude estimate. The filter can therefore be expressed as:

$$0.9 < ar < 1.1 \quad and \quad C_{len} == 4 \quad and \quad C_{min} < C_{area} < C_{max}. \tag{6.16}$$

Here ar is the calculated aspect ratio of the current contour, C_{len} is the number of edges of the current contour, C_{min} and C_{max} are the respective minimum and maximum expected values of the square contour's pixel area at the current altitude, C_{area} is the measured pixel area of the current contour.

Pose Estimation of External Square Marker

When the square marker is successfully detected the relative position of the square centre must be calculated relative to the body axis frame of the UAV. The image frame coordinates must be converted to real world coordinates measured in meters. The exact size of the marker width is known, with this, the pixel width of the marker can be used to calculate the current pixel/meter ratio. This ratio is then used to convert all other image frame measurements to real world distances. For each new incoming image the square marker contour is detected after which the following equations can be used to extract the markers x_c and y_c coordinates measured in the body frame of the UAV:

$$pxl\_m\_ratio = \frac{pxl\_w}{w}. \tag{6.17}$$

pxl_w is the perceived width of the marker in pixels and w is the actual known width of the marker. $x_{c,pxl}$ and $y_{c,pxl}$, being the markers relative pixel distance to the centre of the image frame can be calculated as follows:

$$x_{c,pxl} = \frac{x_{res}}{2} - (m_x + \frac{pxl\_w}{2}), \tag{6.18}$$

$$y_{c,pxl} = \frac{y_{res}}{2} - (m_y + \frac{pxl\_w}{2}) + C_{off}. \tag{6.19}$$

x_{res} and y_{res} are the horizontal and vertical resolutions of the image, m_x and m_y are the top left pixel coordinates of the detected square contour. C_{off} is the physical camera offset relative to the UAV centre in the x_B direction. These coordinates can then be transformed to an inertial frame distance using

$$x/y_c = \frac{x/y_{c,pxl}}{pxl\_m\_ratio}. \tag{6.20}$$

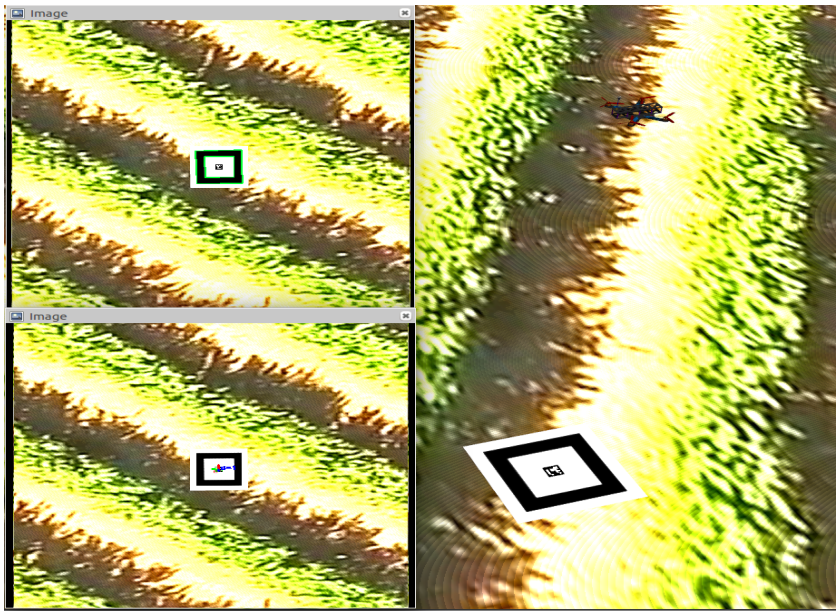


Figure 6.11: Square marker landing simulation showing ArUco and square marker detection

6.5.6 External Square Marker Simulation Results

Figure 6.11 shows the square marker simulation preforming 150 landings. The results of this simulation are compiled in Figure 6.12.

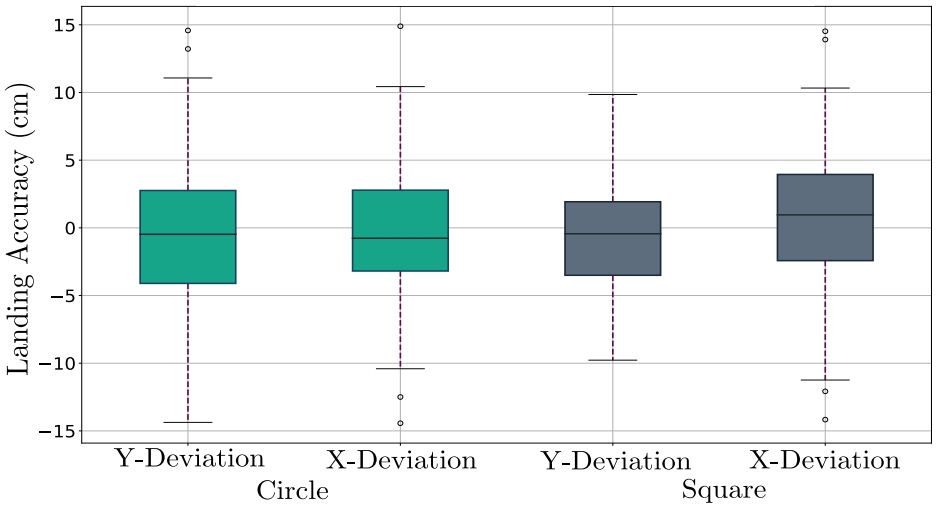


Figure 6.12: Comparing autolanding results of external square and circle markers

From Figure 6.12 it can be seen that similar accuracies were achieved with the square as opposed to the circle marker. This confirms the hypothesis that landing accuracy is more dependant on the inner ArUco marker during the final stages of the landing process. The average accuracy with the square marker was found to be 2.88 cm.

The biggest advantage of using the square marker as opposed to the circle marker is the improved detection rate which has increased from 5Hz to 25Hz on the same PC. This should allow for good performance when used on a companion computer. The simulated landing results of all three landing markers can now be summarised in Table 6.1.

Marker Type	Detection Frequency	Average accuracy (cm)	Worst case (cm)
ArUco	28Hz	19.7	80
Circle	5Hz	3.55	15
Square	25Hz	2.88	15

Table 6.1: Comparing marker performance

Table 6.1 provides a comparison between the three different landing markers in terms of their landing accuracy and their processing cost. These results clearly indicate that the external square marker with inner ArUco marker is best suited for this application.

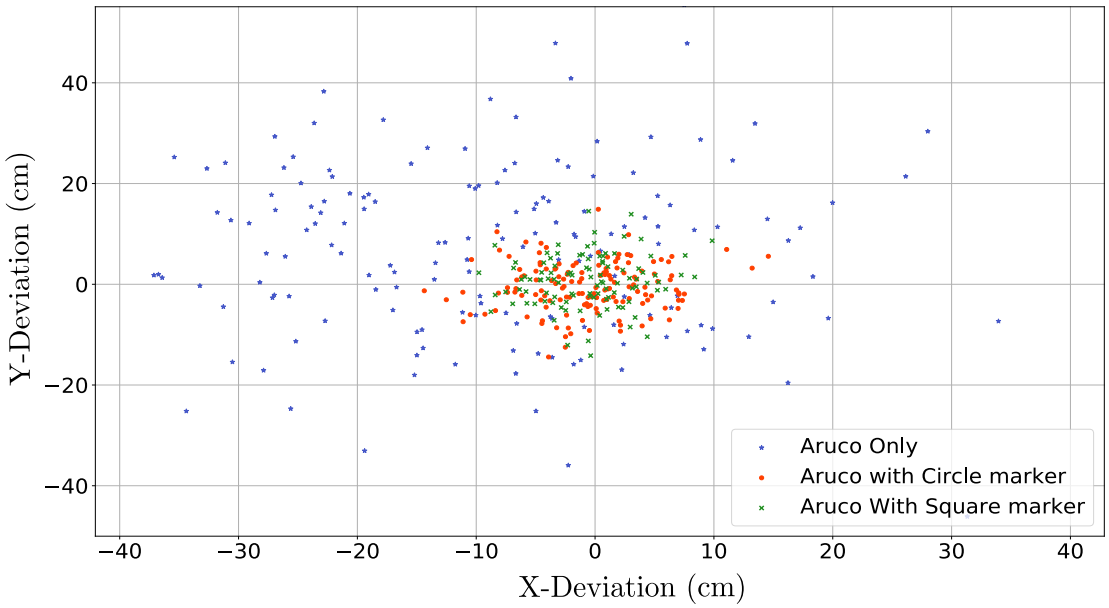


Figure 6.13: Scatter plot comparing landing locations for each marker type

Figure 6.13 compares the landing accuracy distribution of each marker. Here it is clear to see that the landing accuracy of the ArUco marker is greatly improved when augmented with the external square and circle markers.

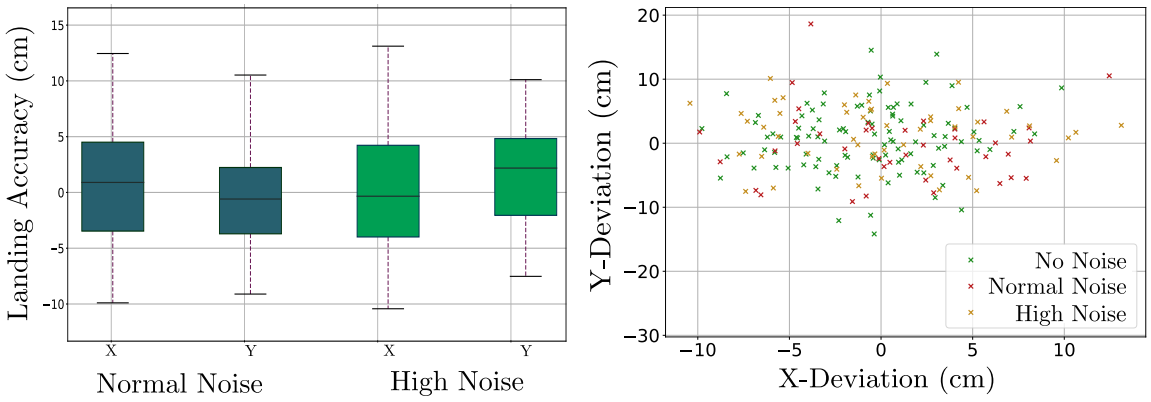
6.6 Robustness to Camera Distortion and Noise

In order to make the simulation more closely matched to the real world some non-idealities are added to the simulation. This gives more confidence in the practical systems performance. The previous results were achieved with the assumption that no noise or distortion is present on the camera image. Although it is possible to remove distortion to some degree, noise will always be present. It is thus necessary to test the effects of these non idealities on the landing system.

A realistic radial distortion model was added to the camera image. Two tests were then performed, the first was with a realistic level of Gaussian noise added. In the second test a large unrealistic amount of noise was added to test the robustness of the system. For both tests, 50 landings were simulated using the square marker. The combined results of both tests are presented in Figures 6.14a and 6.14b.

These results show that landing accuracy was unaffected with the addition of distortion and noise. However in the extreme noise case it was observed that the markers were detected less frequently. This meant that the UAV was sometimes in a state where no new positional setpoints were being published. This meant that the UAV would need to rely only on attitude control to stay stationary until the next marker detection. This is acceptable as long as there are no external disturbances acting on the UAV such as wind or weight imbalances. This infrequent detection behaviour still allowed for accurate landings in the ideal simulation environment. It did have an effect on landing time which increased from an average of 45 to 90 seconds. In the normal noise case detections remained robust and frequent, this meant that there was no increase in landing time.

All landings presented thus far were performed with sensor noise added to the outputs of the other simulated sensors. It can therefore be concluded that the autolanding system is robust to system related non-idealities.



(a) Box plot

(b) Scatter plot

Figure 6.14: Landing accuracy distributions showing effect of camera distortion and noise.

6.7 Practical Autolanding Tests

The autolanding system has thus far been tested and proved in simulation only. It is difficult to model real world factors perfectly in simulation thus a structured practical testing process is followed leading up to a fully autonomous landing. This eliminates possible points of failure in the system. Each individual component of the autolanding needs to be tested in a controlled experiment to ensure that it works as expected. The required tests are summarised as follows:

- Test square marker detection algorithm robustness and detection frequency with the Jetson Nano and Raspicam V2.
- Test square and ArUco marker detection robustness while manually flying UAV over the actual landing marker.
- Test UAVs ability to autonomously track and hover over marker at fixed low altitude.
- Perform autolanding.

6.7.1 Practical Marker Detection Test

It was necessary to test the marker detection performance on the actual flight hardware. The experiment would need to measure detection frequency and noise. It is also necessary to test the robustness of the square marker detection filter in a practical setup. The marker has to be detected reliably over a distance range of 3m to 7m. The extracted z_c component of the detected ArUco marker is used for distance feedback in the square detection filter.

The camera was placed at a fixed location while the marker was moved at fixed 1 m increments away from the camera. The extracted marker pose was recorded for 10 seconds at each 1 m interval. This experiment was performed indoors with lots of square shaped background features, this would test the robustness of the square detection filter. The noise profile of the extracted marker pose is shown in Figure 6.15.

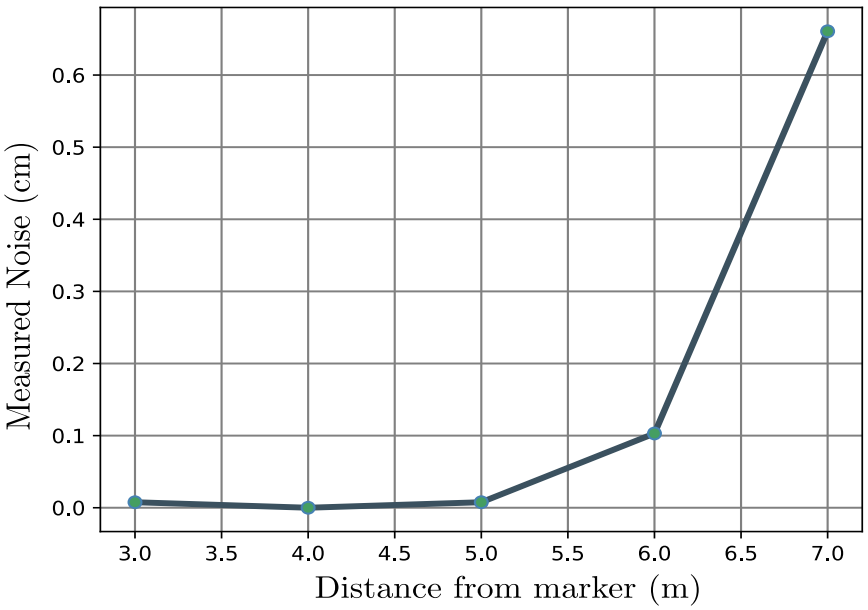


Figure 6.15: Square marker measurement noise vs camera distance

It is observed that the markers pose measurement noise increases as the camera moves further away from the marker. However the noise level remains low at the maximum distance of 7 m and is completely negligible at low altitude or during the final stage of the landing process where minimising noise is most important. During the experiment it was observed that the marker was detected reliably with no missed or erroneous detections. An anomaly was noticed which showed that the time between detections was not consistent. This same behaviour was observed during the simulated noise tests. Two possible causes are identified:

- Random Gaussian noise present in the camera image causes periods where no marker detections can be made.
- ROS's irregular real-time process time scheduling causes non-periodic detections.

This non-periodic or flickering type detection behaviour was recorded and summarised in the Table 6.2.

Maximum Detection frequency Hz	Minimum Detection frequency Hz	Average Detection frequency Hz
82.0	7.3	17.3

Table 6.2: Detection flickering analysis

The slowest detection has a frequency of 7.3Hz, this is considered fast and thus this effect can be ignored. In conclusion this experiment showed that the current system is efficient and effective at detecting the square marker in a practical setup.

6.7.2 RC Flight and Marker Detection Test

The marker detection performance has to be tested while the UAV hovers above the marker using manual RC control. This allows for the most safe and realistic test case before an autonomous test is attempted. It is important to verify that the UAV is able to consistently detect and square marker throughout the flight and over a wide altitude range. It is also important to determine at which altitude the ArUco marker is most reliably detected. The results of this experiment are shown in Figure 6.16.

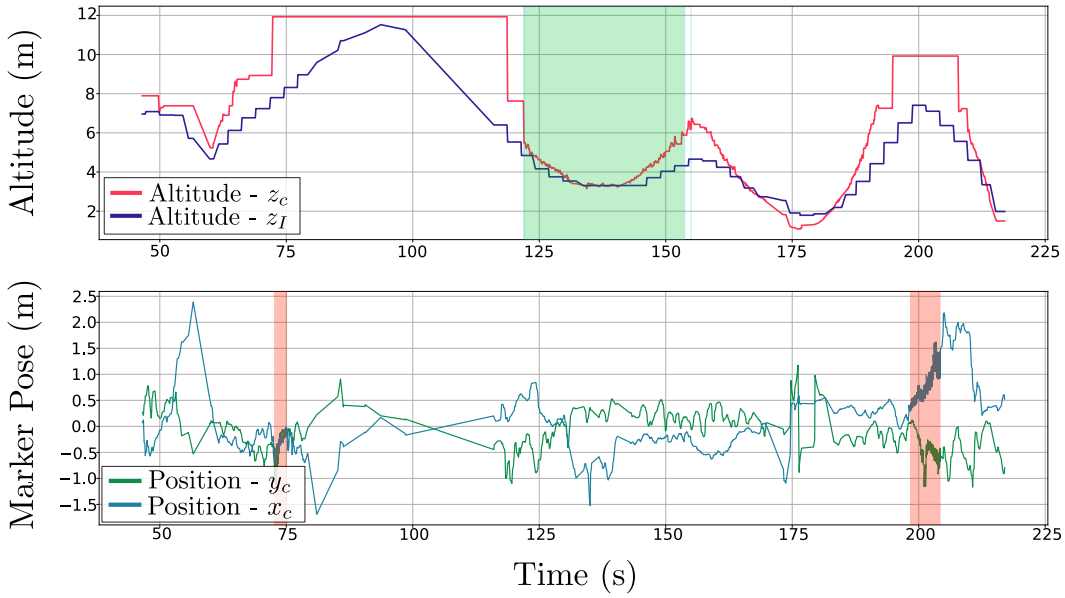


Figure 6.16: Recorded flight data of manual RC flight

Figure 6.16 shows that there is a consistent stream of square marker pose data over all altitudes. Two instances highlighted in orange show noise on the output of the extracted marker position. Both these events occur at high altitude and is expected but do not pose a danger to the stability of the UAV.

The area highlighted in green demonstrates an altitude range where the ArUco marker is reliably detected. At this same altitude the square marker is also being detected reliably. This suggests a good altitude range where the swap from the external marker to the inner marker can occur.

This test confirms that the square marker detection works as expected and that the ArUco marker is well detectable at lower altitudes.

6.7.3 Track Marker Test

For this test the UAV flies to an altitude of 6m before being switched to offboard mode. In this mode the UAV is piloted by the onboard computer which publishes the relative marker pose to the marker tracking controller on the FCU. A fixed altitude setpoint of 6 m is also published to keep the UAV stationary at this altitude. This tests the practical stability and tracking ability of the UAV's controllers. The relative marker pose was logged and is shown in Figure 6.17.

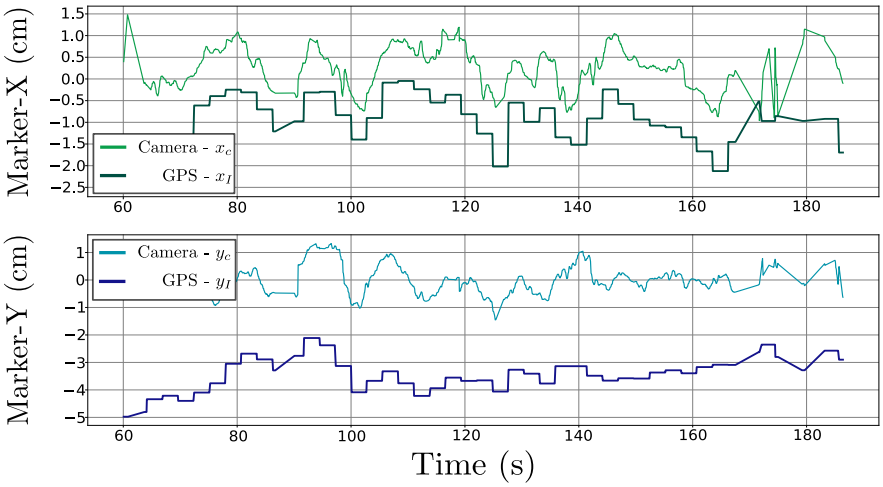


Figure 6.17: Showing relative marker position during autonomous tracking test

It is observed that the UAV tries to track the marker but overshoots the marker on each approach. This is clearly noticeable in Figure 6.17 where the UAV oscillates around position 0.0. It is also interesting to note the large GNSS inaccuracy present during the test which is roughly around 4m.

When inspecting the velocity controllers step response more closely, a large overshoot was present in the North-East Velocity Controller. The gains of this controller were adjusted as described in Section 5.7. The test was repeated and results are shown in Figure 6.18.

From Figure 6.18 it is shown that the UAV's tracking performance greatly improves as UAV is stable around position 0.0 throughout the test flight. The marker is moved by roughly half a meter just before the 60 and 80 second mark. On both occasions the UAV corrects its position as expected. This concludes the required tests needed to prove that the UAV is capable of autonomously tracking the landing marker. The next step is to attempt a practical landing and optimise state machine variables for best landing results.

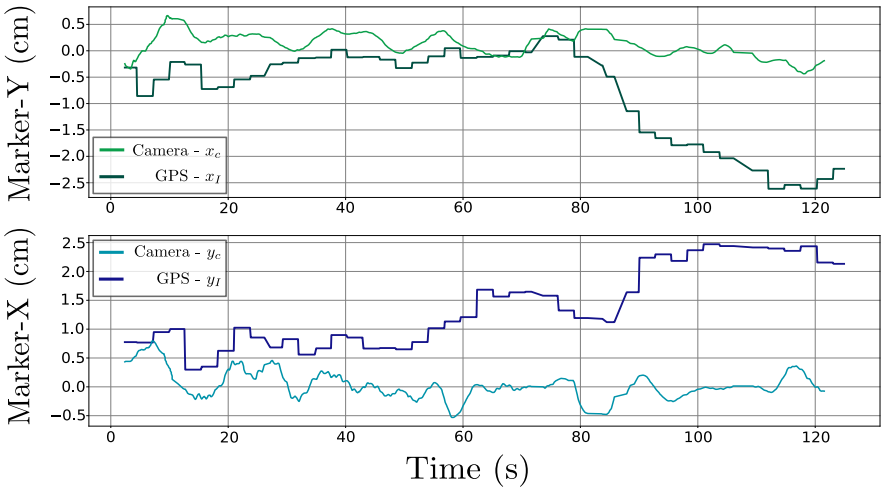


Figure 6.18: Showing relative marker position with improved tracking performance

6.8 Practical Autolanding

In the practical autolanding experiment the UAV was manually piloted to 8m above the marker with roughly a 1-2 m lateral positional offset from the marker to simulate initial GNSS inaccuracy. The OBC was programmed to automatically start the required ROS packages and nodes on startup. The custom autolanding ROS package waits for the UAV to be switched to offboard mode before starting the autolanding state machine. A switch on the RC flight controller is programmed to switch the UAV between offboard mode, position mode or hold mode. Position mode is used to manually pilot the UAV whereas hold mode commands the UAV to hover at its current position and altitude. These modes acted as a failsafe and were available in the event of unexpected UAV behaviour.

Successful autolandings were achieved after minor adjustments to some of the state machine variables. The UAV's landing strategy proved to be robust to external disturbances such as wind. The UAV was able to accurately land in moderately windy conditions and was capable of successfully repairing a failed autolanding. Figures 6.19 and 6.20 show flight log data of one particular landing.

Figure 6.19 shows the recorded relative x_c and y_c positions together with GNSS x_I and y_I measurements over the first two stages of the autolanding. The current estimated altitude of the UAV is indicated in orange. As expected the UAV is seen descending to 4m once state 2 is initialised. This landing was done in windy conditions which led to the UAV being blown of coarse and resulted in the UAV losing sight of the marker. This event is highlighted in red. Losing the marker for more than 3 seconds triggers a repair state which commands the UAV to increase altitude until the marker is visible. The point at which this state it initialised is indicated on Figure 6.19. The UAV increases altitude to 5.4m before regaining visual contact with the marker. At this point, state 2 is resumed and the UAV's position and altitude is corrected before state 3 is initialised.

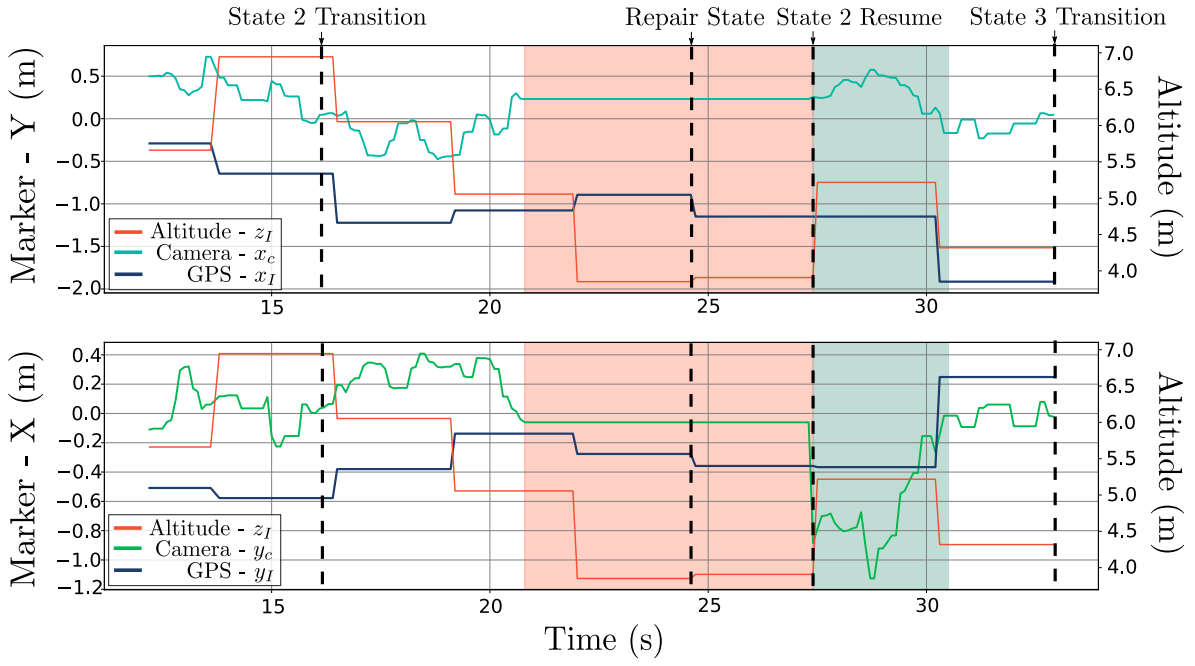


Figure 6.19: Flight log data during stage 1 and 2 of autolanding showing extracted marker pose, equivalent GNSS position and barometer based altitude estimation.

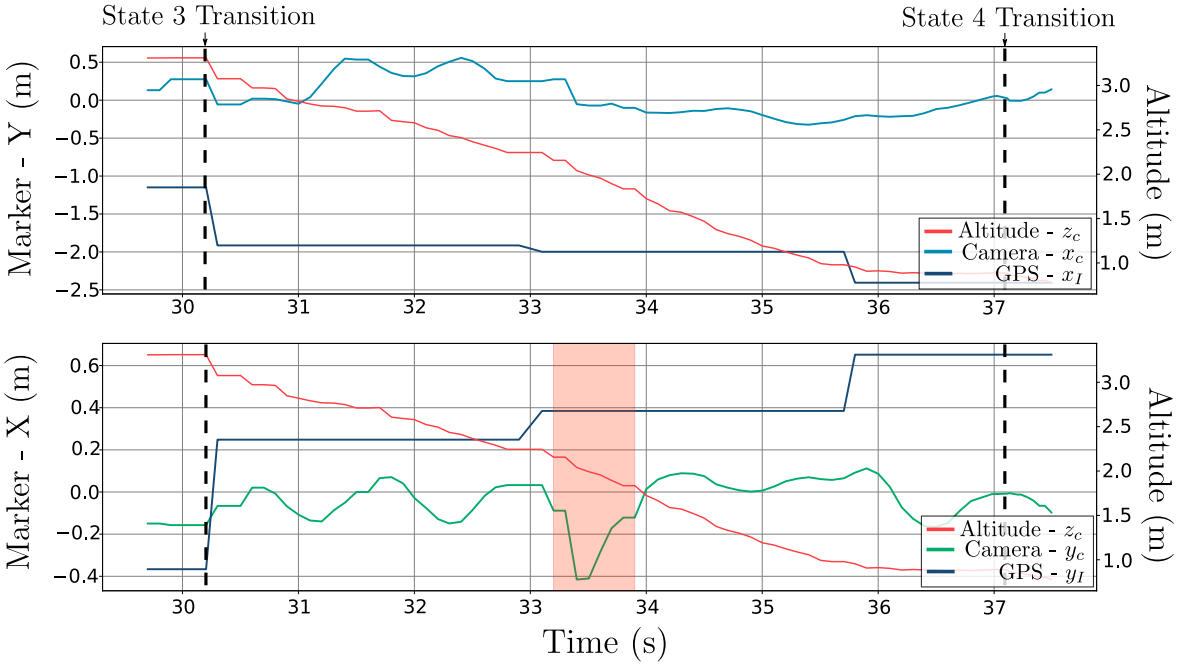


Figure 6.20: Flight log data during stage 3 and 4 of autolanding showing extracted marker pose, equivalent GNSS position and marker based altitude estimation.

Figure 6.20 shows flight log data of the final two stages of the landing. In state 3 The UAV uses the small inner ArUco marker for horizontal position as well as altitude feedback. These measurements are recorded in Figure 6.20 along with the UAV's current GNSS location. The UAV manages to stay stable until 33 seconds where a wind gust creates a sudden disturbance. The UAV quickly recovers and switches over to the final landing stage after only 7 seconds in stage 3.

Another 20 landings were completed and the exact landing location was recorded each time. The combined results are shown in Figures 6.21 - 6.23.

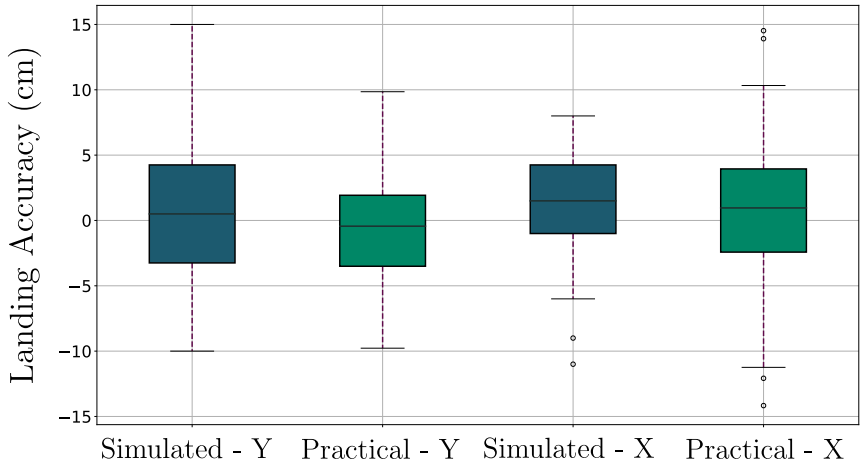


Figure 6.21: Comparing practical and simulated landing accuracy

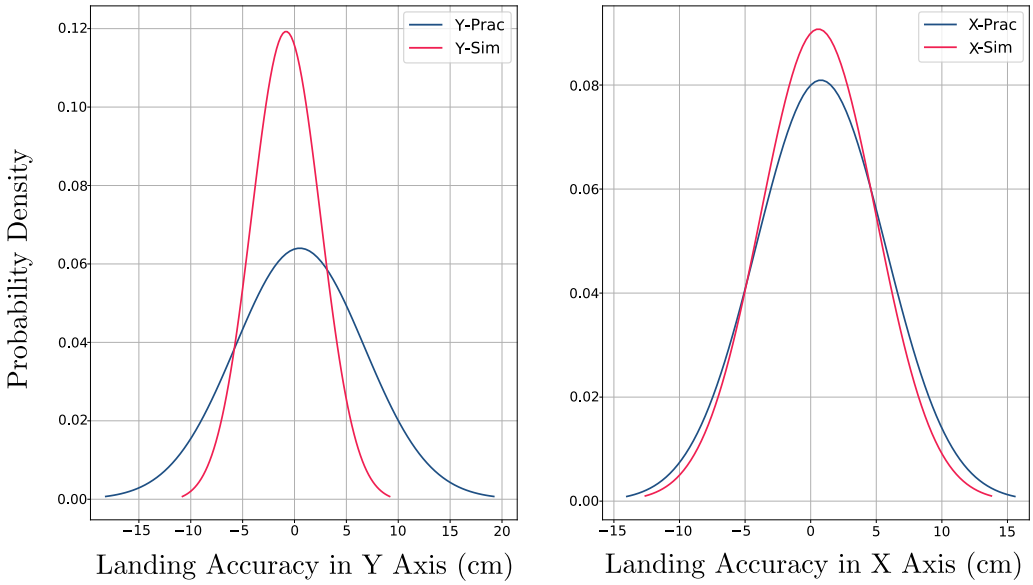


Figure 6.22: Normal distributions of practical and simulated landings

Figures 6.21 and 6.22 compare the combined accuracy of all practical landings and simulated landings. Overall simulated landings were more consistently accurate. This is not surprising as the simulation does not account for external disturbances such as wind and is more ideal in terms of practical sensor noise and weight imbalances in the practical UAV's airframe. The practical results are still considered good considering these external factors with an average landing accuracy of 4 cm. The achieved practical landing accuracy is acceptable for the purpose of this project. As it would be difficult to improve accuracy any further the remaining landing offset should rather be absorbed in the design of the recharging station. The same level of landing accuracy can be achieved using other technologies such as differential GNSS, however this would require placing an antenna at each charging station. This becomes very expensive if multiple recharge stations are required. Using an inexpensive camera proves to be just as effective. Figure 6.23 shows the landing locations of the 20 landings used to generate the practical landing results.

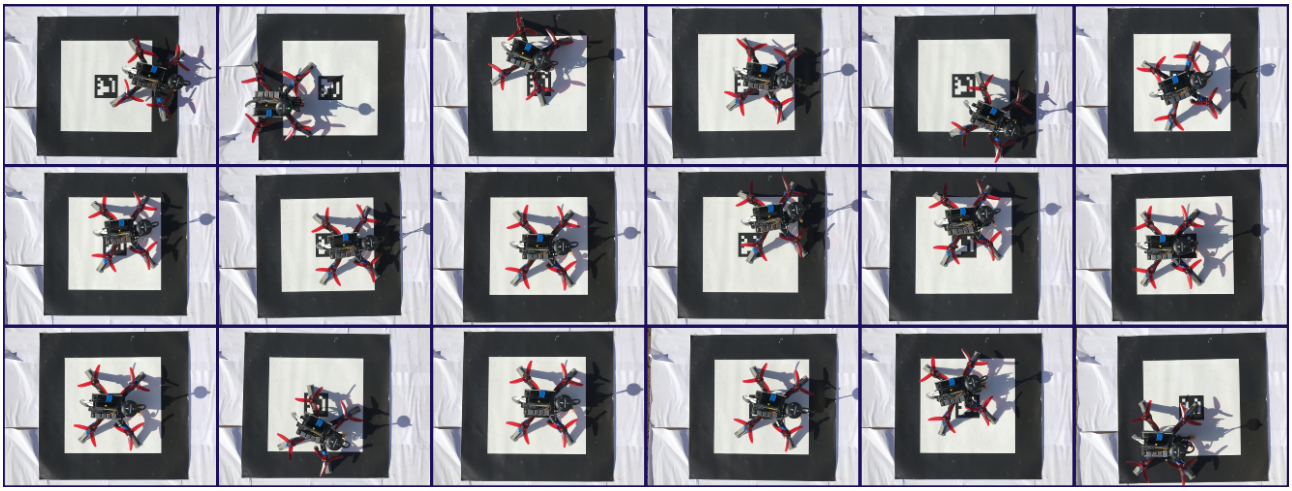


Figure 6.23: Practical landing locations

6.9 Summary

This chapter gave a detailed overview of the state machine that was designed for the autolanding. The three different positional controller modes used were discussed. The first is the default position controller used by PX4, the other two are adaptations of this default controller. A Gazebo simulation setup was used to design and test the autolanding system. This chapter describes the camera model that was used for this project and discusses camera calibration and distortion removal. Three different landing markers were tested in simulation where each one was evaluated based on its landing accuracy performance and detection frequency. The marker consisting of an outer black square and smaller inner ArUco marker was chosen due to its computational simplicity and landing performance. The landing systems robustness to camera noise and distortion was also tested. These tests showed that the system was fairly robust to realistic levels of image noise. Finally the chapter presents practical landing results and testing methods used leading up to the autonomous landing. Practical landing results had an average accuracy of 4cm. Although this is not perfect the remaining landing offset can be absorbed in the design of the recharging platform. The landing results achieved can be compared to DGPS but can be implemented at a fraction of the cost.

Chapter 7

Recharging System

This chapter presents conceptual designs and practical results of a self recharging system for a quadrotor UAV. A 3D modelled prototype of the charging station is presented but was not built in this project. Designs of the charging mechanism and circuitry are provided. A practical charging circuit was successfully tested. This chapter starts by discussing important battery concepts and terminology before investigating different charging approaches in more detail. This chapter then provides designs of the physical and electrical components of the charging system before providing practical charging results.

7.1 LiPo Battery and Recharge Overview

This section aims to define basic Lithium Polymer (LiPo) battery terminology and concepts before proceeding to select a charging method. Figure 7.1 shows the LiPo battery used for this project.

Figure 7.1: LiPo battery used for this project

7.1.1 Cell Arrangement

A LiPo battery pack is made up of individual LiPo cells. The cell arrangement in a battery is described using the format xSyP. x describes how many cells are connected in series which contributes to the overall battery voltage. y describes how many cells are connected in parallel which contributes to the overall battery capacity. Multiplying x and y gives the total number of cells in the battery. Referring to Figure 7.1, this battery has 4 cells connected in a 4S1P configuration meaning that all the cells in this battery are connected in series contributing to a total nominal voltage of 14.8 V.

7.1.2 Capacity

The capacity of the battery Q_{max} is measured in mAh and is determined by the parallel cell arrangement and the physical size of the battery cells. This battery has a capacity of $Q_{max} = 2000$ mAh which means that the battery could supply 2 Amps for 1 hour before being fully discharged. The battery has a 1P arrangement which means that each cell in the battery has a capacity of 2000 mAh.

7.1.3 Voltage

Three different voltages are worth defining for a LiPo battery. These voltages are described in Table 7.1.

Charged	The voltage of a fully charged LiPo cell is 4.2 V. Exceeding this voltage could cause permanent damage to the cell.
Nominal	Value used by manufacturers to describe the voltage of a cell which is 3.7 V.
Discharged	3 V is the minimum voltage a LiPo cell can discharge to before causing permanent damage.

Table 7.1: Voltages associated with LiPo Batteries

Referring to Figure 7.1, this battery has a nominal voltage of $3.7 \times 4 = 14.8V$ and a full battery voltage of $4.2 \times 4 = 16.8V$.

7.1.4 C Rating

The C rating of a LiPo battery is a constant which when multiplied by the mAh rating of a battery gives an indication of the maximum current that can safely be drawn by the battery during discharge and charging. Three different C constants representing three unique current ratings are given to each battery. These are summarised in Table 7.2 along with the calculations showing the corresponding maximum currents for the battery used in this project.

C Rating	**Description**
Constant discharge C_d	Indicates the maximum current that can be drawn continuously. $C_d \cdot Q_{max} = 120 \cdot 2 = 240$A
Burst discharge C_b	Indicates the maximum current that can be drawn for a small period of time (10 seconds). $C_b \cdot Q_{max} = 240 \cdot 2 = 480$A
Charge C_c	Maximum current that can be applied to the battery during charging. A higher C_c value means faster charging, however it is not recommended to exceed $1C_c$ as this decreases the lifetime of the battery. $C_c \cdot Q_{max} = 3 \cdot 2 = 6$A

Table 7.2: C ratings of LiPo batteries

7.1.5 Cell Balancing

When a battery is used, the individual cells often discharge at different rates due to non-idealities in each cell. This causes the battery cells to become unbalanced. This effect

increases as the battery gets older. During charging, it is important to ensure that each cell is charged back to the same voltage independently of other cells, this is called balanced charging. Unbalanced charging causes some cells to become overcharged while others remain undercharged. This results in the battery becoming more unbalanced and can result in an unsafe permanent failure of the battery.

Each battery has a balance plug and a discharge plug as indicated by (1) and (2) in Figure 7.1 respectively. The balance plug contains 5 wires which connect to each cells positive terminal while one is used to connect to the most negative terminal of the battery. The balance plug is used during charging to control each cells voltage individually, thereby ensuring that each cell remains balanced.

7.2 Recharging Approach

Section 2.2 presented three different methods used for recharging multirotor batteries. These were: direct contact approach (DCA), wireless charging using WPT and a battery swapping approach. This thesis takes a closer look at the first two approaches and dismisses the battery swapping approach as the mechanical complexity of this system is out of the scope of this thesis. This approach does however show promise due to the near instantaneous recharge times and could be investigated further in a different project. DCA and WPT are now compared in the context of this project. From this analysis the best approach is chosen. The projects charging requirements are firstly listed as follows:

- Fast recharge time.
- Robust. Successful recharging must be repeatable.
- Efficient. Minimise power loss in limited power scenarios (Solar powered).
- Simplistic. A simple system has less margin for error and is easier to maintain.
- Suited to an outdoor environment.

7.2.1 Wireless Power Transfer (WPT)

Resonant inductive coupling is a type of WPT that is used to transfer electrical power from a transmitting coil located on the charging station to a receiving coil located on the UAV via resonating magnetic fields. Power can be transferred this way at short distances, although, the system is most efficient when there is minimum separation between the transmitting and receiving coils. This means that in order to have acceptable recharge times a mechanism is required to position the UAV directly above the transmitting coils located on the charging platform. Available off-the-shelf WPT components are capable of achieving a maximum efficiency of 90% if no separation distance exists between the coils, however this efficiency drops quickly when placed further apart and thus accurate placement is key to ensuring acceptable efficiency.

The available components have a respective maximum output voltage and current of 5V and 2A per receiving coil. In order to charge a 4S battery such as the one used in this project, four receiving coils would need to be connected in series and placed on the UAV in order to generate the required voltage and current needed for recharge. Additional hardware required on the UAV is a voltage regulator which ensures that the correct voltage is supplied to the batteries at all times. Also required onboard the UAV is a balanced charging circuit which manages the charging process. Adding all this additional hardware on a small UAV such as the one used for this project would impact the mass of the UAV significantly. This would

also impact the inertia of the UAV which could potentially require adjustment to UAV's controllers. WPT is thus better suited to larger UAVs who have onboard space for additional hardware and also have the payload carrying capacity. Some UAVs have large batteries which require large amounts of current for optimal charging. Supplying this current via WPT requires large coils and becomes increasingly inefficient making this solution less practical.

The advantage of using WPT is its suitability to outdoor environments. The transmitting coils can be enclosed in a protective casing which isolates the electrical system from external factors such as dust and rain.

7.2.2 Direct Contact Approach (DCA)

A direct contact approach makes use of electrical conductors located on the charging station and UAV. Large amounts of electrical energy can be efficiently transferred from the charging station to the UAV via these conductors and thus this approach can achieve faster recharge times when compared to WPT. In a direct contact approach the UAV's placement accuracy on the charging platform are also key to ensuring successful recharging. In both WPT and DCA a mechanism is required to position the UAV after an autolanding and thus neither approaches are advantageous in this regard.

Similarly to the approach used for WPT, the charging circuitry could be placed directly on the UAV. This would result in the same disadvantages as discussed earlier, however this would simplify the charging stations contact pad design by reducing the required conductors to two as only the full battery voltage needs to be applied to the conductors. Alternatively, majority of the charging circuitry could be placed on the charging station which excludes the need for any major modification to the UAV. This approach would require a more complicated contact pad design as four separate voltages (one for each cell of the battery) need to be applied to the UAV using five conductors opposed to two. Minimising additional onboard hardware is prioritised as this is a requirement for smaller UAVs. This direct contact approach is thus chosen as the recharging method for this project.

In order to make this approach better suited to an outdoor scenario the contact surfaces would need to be protected against rain and dust using a protective enclosure.

7.3 Charging Station Prototype Design

This section presents a 3D modelled prototype of the physical charging station system. Each individual component's role is discussed.

Figure 7.2 presents a modelled prototype of the charging station. (1) is a protective enclosure designed to house the charging platform (5). The platform is able to slide in and out of the charging box using a retractable rail system (4). Doors (2) are able to open and close using motorized brackets (3). Included on top of the charging station is a large ArUco marker (6) which is used to improve the general localisation accuracy of the UAV. This method is discussed in detail in Chapter 8.

When a UAV is ready to land the charging platform slides out of the charging box. Once the UAV has successfully landed it can be retracted back into the box to charge safely while protected from natural elements. UAVs can also be automatically stored this way in the event of bad weather or when not in use.

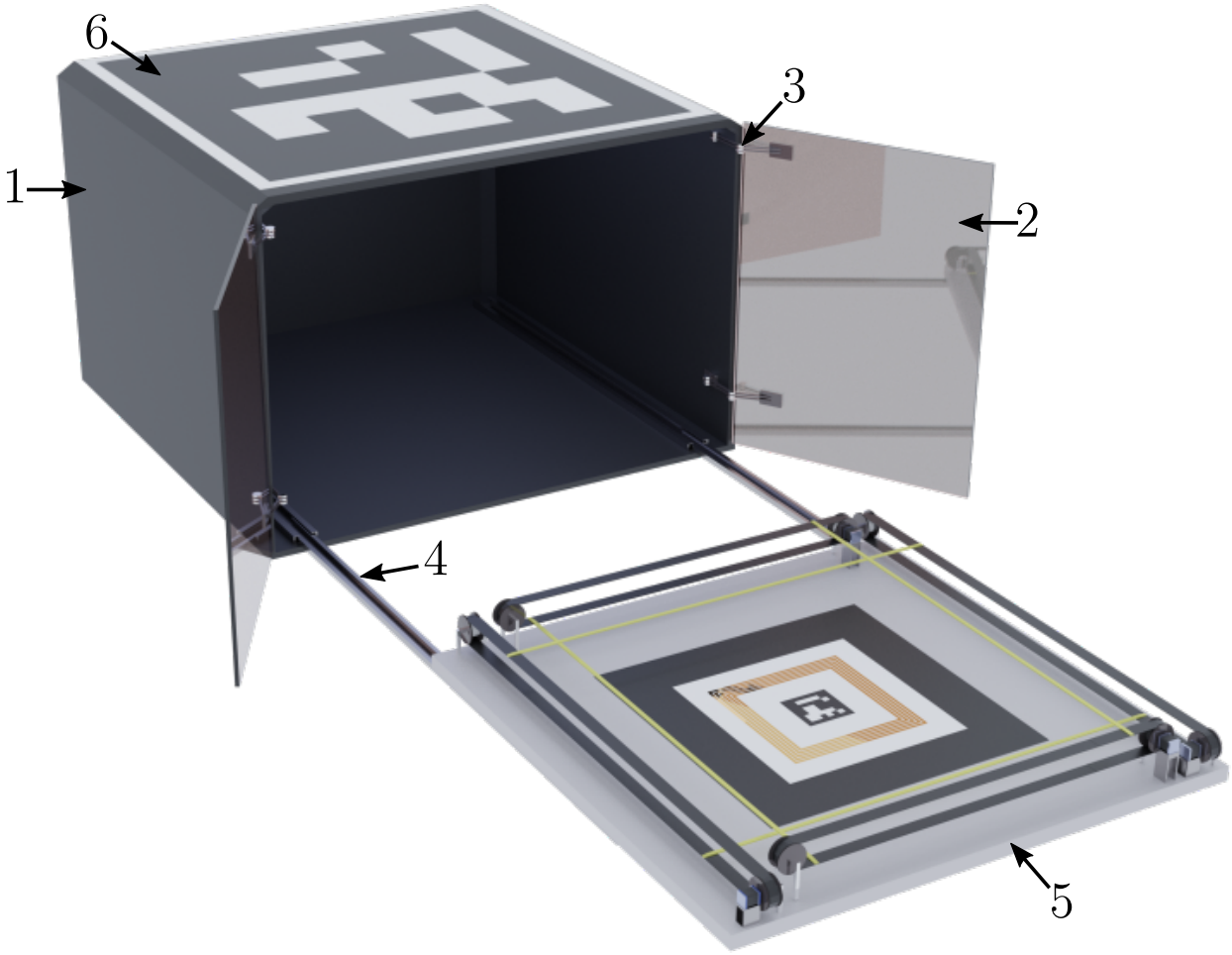


Figure 7.2: Charging station prototype

This prototype does not fully model the mechanical complexity of this system but instead provides a clear conceptual idea of how this system would work and look.

Figure 7.3 provides a closer look at the charging platform. Once the UAV has successfully landed on the platform, it requires accurate repositioning to ensure electrical contact between the charging pads (10) and the conductors on the UAV. To accomplish this four positioning rods (9) move inwards and push the UAV into the required position. These positioning rods are attached to four rubber belts (7) which rotate around eight pulleys (8). Four of these pulleys are precisely actuated using four stepper motors (6). Two pairs of belts are placed at different heights in order to avoid contact between the positioning rods. Also included on the charging platform are four pressure pads which are placed under each corner of the charging station. These are used to detect a UAV landing on the platform.

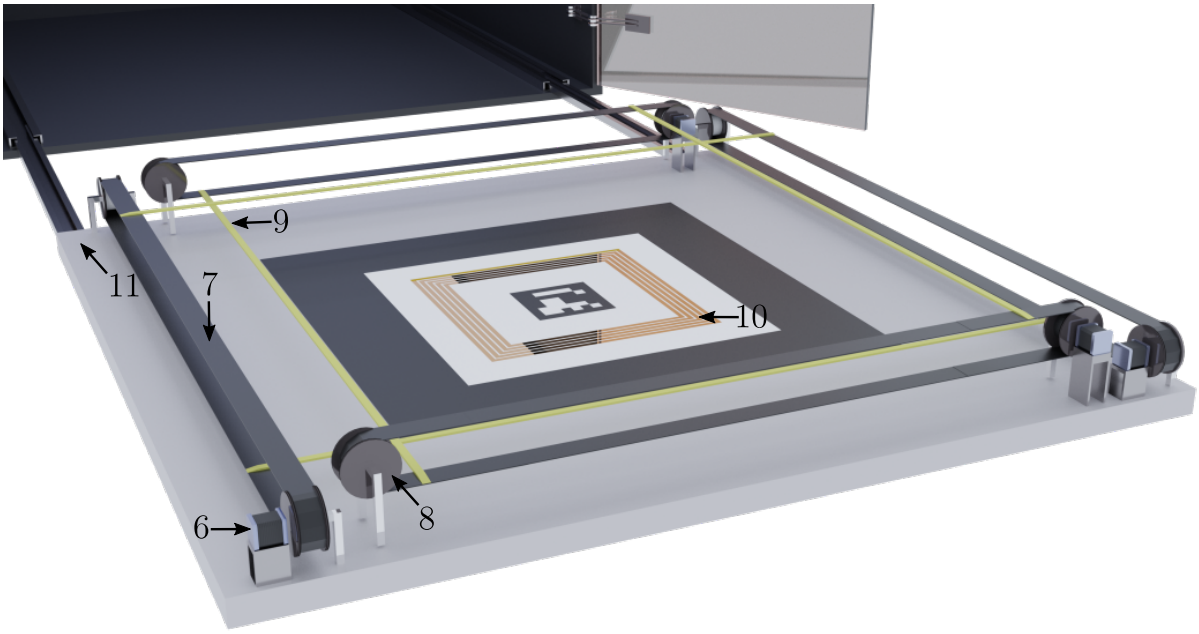


Figure 7.3: Charging station prototype

Figure 7.4 provides a dissected view of the custom landing gear that can be fitted to any UAV. In this design sharp electrodes (16) are kept pressed down on the charging pads (10) under pressure, using a compression spring (18), which is able to freely slide inside a rigid cylindrical sleeve (17,15). The sleeve is fixed mounted to the UAV's landing gear (12) using a protective casing (14). This casing and the cylindrical sleeve have been cross sectioned to reveal the inner parts and would otherwise enclose the springs and electrodes. Wires (13) connected to each electrode are routed up towards the charge management system (CMS) discussed later in Section 7.4.3.

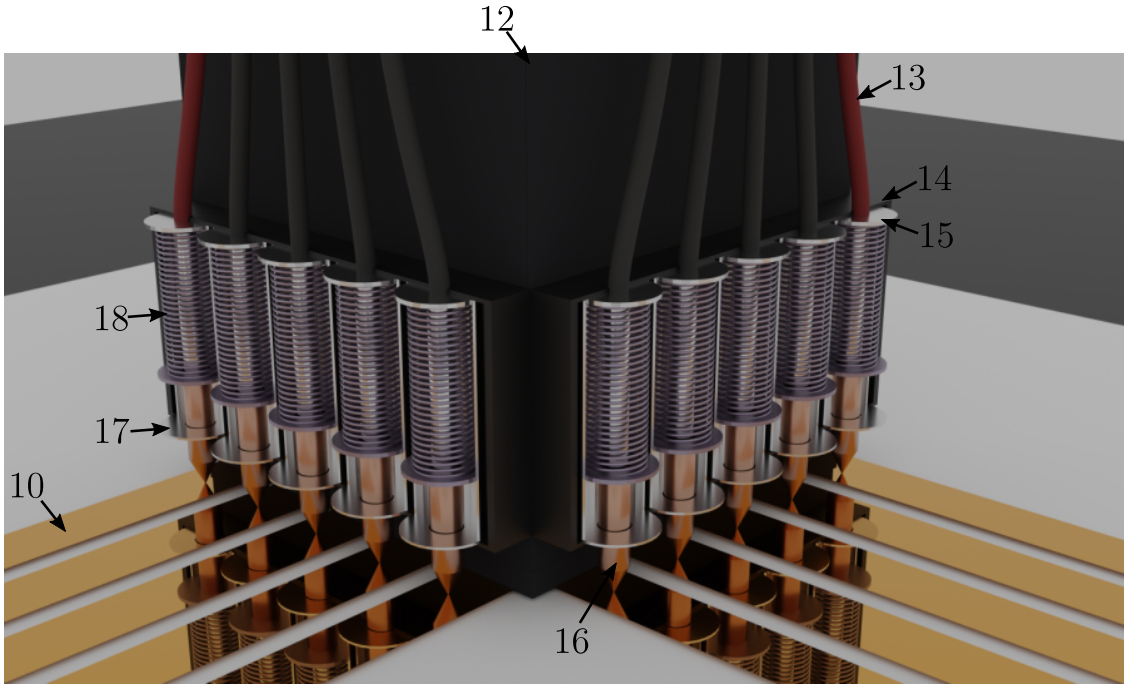


Figure 7.4: Dissected view of the landing gear prototype

This landing gear design can be fitted to each leg of the UAV which would result in a total of eight electrodes making contact with each charging pad. This maximises the chance of making a successful connection and increases maximum current transfer. This landing gear could be fitted to any UAV airframe using different mountings which could be designed for a wide array of existing UAV airframes thereby making this a transferable solution.

7.4 Recharging Circuitry

This section provides an overview of all custom circuitry needed for the recharging process. This includes a detailed overview of the electronics required onboard the recharging station as well as a detailed overview of the charge management system (CMS) located on the UAV.

7.4.1 Recharge Station Electrical System

The requirements for the recharging stations electrical systems is listed as follows:

- Must have on/off toggling system for charging pads.
- Must be able to detect when UAV has landed.
- Must be able to control all the charging stations stepper motors.
- Must be deployable in remote locations.

From the requirements a system block diagram is presented in Figure 7.5. A microcontroller is used to manage and integrate the three main electrical systems of the recharge station, namely: motors, pressure sensors and the charging circuit. The microcontroller sends the required signals to the stepper motor driver circuits to actuate the four motors. The microcontroller can also turn the charging circuitry on and off using a relay switch. Power is delivered to the system via 220V AC mains rectified to 24V DC using a power supply or alternatively a solar panel and battery can be used in remote locations.

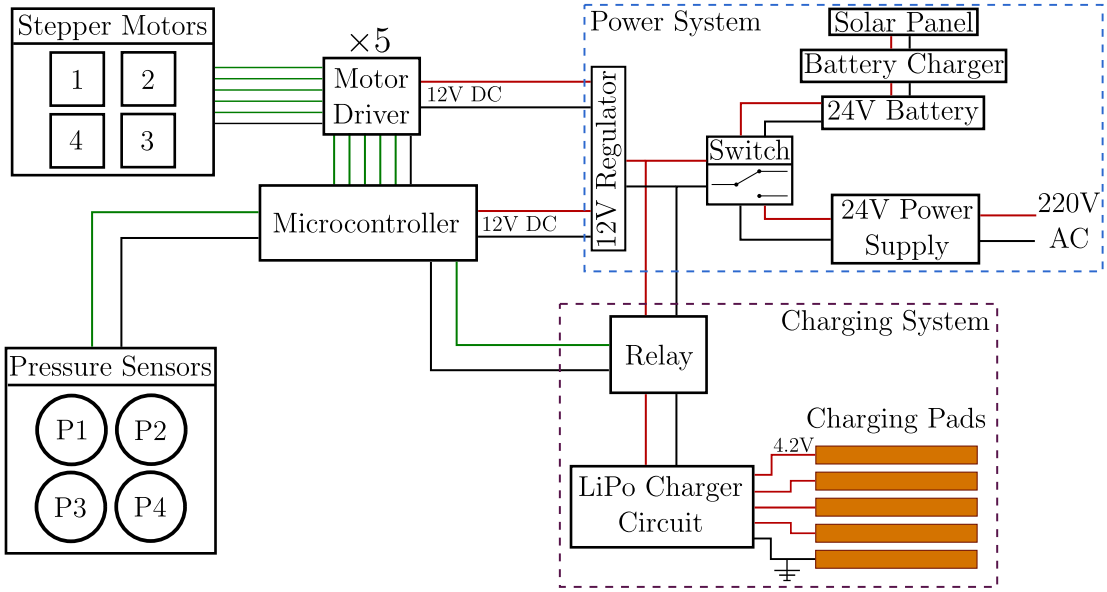


Figure 7.5: Recharge station electrical system block diagram

7.4.2 LiPo Charger Circuit

A block diagram of the LiPo charger circuit is shown in Figure 7.6 below.

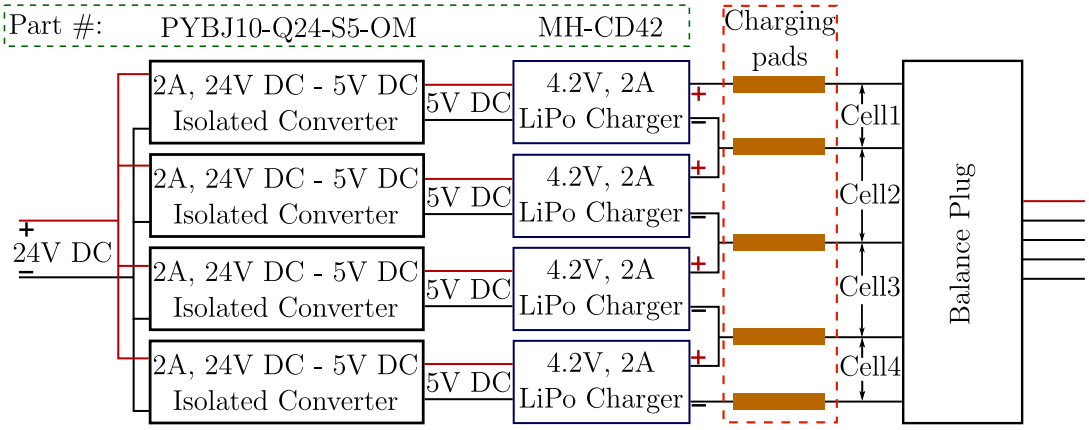


Figure 7.6: 4S LiPo balanced charging circuit

In this circuit four electrically isolated DC-DC converters are used to convert the 24V input to 5V output required by the LiPo chargers. These converters also provide ground isolation between the input and output so that the LiPo chargers may be connected in series. A 2A supply is required for each LiPo charger in order to charge each cell at $1C_c$. Although each cell can be charged faster by applying higher current this will result in long term damage after each charge and decrease battery life span.

Four MH-CD42 LiPo charging modules are connected in series. In this configuration each cell is charged individually thereby balancing the battery. These modules are capable of charging at a maximum current of 2.1A, making it an ideal charger for this project. This module also has important battery protection features such as: over-current protection, over-voltage protection, short-circuit protection, over-temperature protection. This protects the battery from potentially dangerous conditions during charging.

7.4.3 Charge Management System (CMS)

The charge management system (CMS) is a small and lightweight custom PCB that can be fitted to any UAV thereby enabling recharging via this recharging station design. The CMS connects the custom electrodes fitted to the UAV to the UAV's battery system. The functions and requirements of the CMS are listed below:

- Acts as intermediary circuit connecting the UAV's battery and power distribution board (PDB).
- Disconnects battery from UAV's PDB during charging.
- Connects UAV's charging conductors to the batteries balancing plug to enable charging.
- Perform battery voltage measurement for sensing charge completion and fault detection.
- Must be small and lightweight appropriate for any UAV.

Once the UAV has been moved into position on the charging station it is important to verify that the onboard conducting electrodes have successfully made contact with the charging pads by ensuring that the correct voltage is being applied before connecting the charging circuit to the UAV battery. To do this a microcontroller measures the electrode voltage using a voltage measurement circuit while disconnecting the electrodes from the battery using the

MOSFET switches M1-M5 as shown in Figure 7.7. If the voltages are not correct then the positioning process is reattempted.

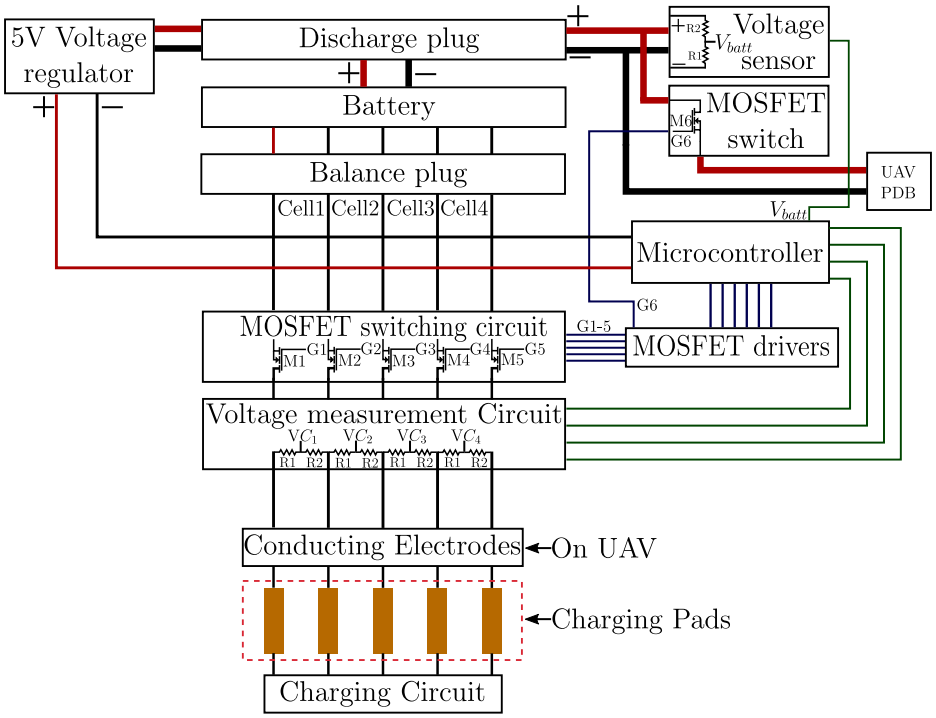


Figure 7.7: Block diagram of the charge management system

If the voltages are correct then the MOSFET M6 is turned off thereby disconnecting the battery from the UAV's PDB. M1-M5 are turned on to connect the charging circuit to the battery via the balance plug. The electrode voltage is consistently monitored to detect any faults that may occur during charging.

The total battery voltage is measured to determine when the battery is fully charged. At this point the battery is disconnected from the charging circuit and reconnected to the UAV's PDB. The charging circuit will automatically also stop applying voltage once it detects that the battery is fully charged as a secondary safety precaution.

A 5V regulator is connected to the battery which provides power to the system. Also included are MOSFET drivers which drive the gates of each MOSFET. A microcontrollers logic pin can normally not supply enough current to charge the MOSFETS input capacitance to turn it on fully. Back-current is also induced by the MOSFET gate capacitance during discharge which can damage a microcontroller. The driving circuits are designed to handle this back-current.

7.5 Charging Results

This section provides charging performance results of the custom charging circuitry shown in Figure 7.6. The supply and charging current in this experiment was limited to 2A in order to charge the 2000mAh battery at $1C_c$. The theoretical total charge time from completely empty to completely full is 1 hour. This charge time could be lowered significantly if the battery were to be charged with higher current although this would decrease the battery life time. If charge time was a priority and regular battery replacement was not considered an issue then the same circuit components with higher current ratings could be used.

Using the charging circuit, the total charge time of the 4S battery was measured to be 65 minutes. This is close to the estimated value. It is reasonable to assume that a UAV would never be charged from a fully depleted battery state and thus in practise this charge time would be lowered by roughly 25-35%.

7.6 Summary

This chapter gave a detailed overview of a proposed recharging station. LiPo battery concepts and terminology was discussed first before choosing a charging approach. It was determined that the recharging circuit would need to provide cell balancing as this greatly reduces battery related risks and prolongs the batteries life span.

The chapter then discussed two main charging approaches which were considered, namely: wireless power transfer approach and a direct contact approach. It was decided to use a direct contact approach as this method has the fastest charging time, requires the least amount of additional onboard hardware and is much more efficient than WPT at transferring power. In multi-cell batteries, large amounts of current is required to charge each cell. The coils in a high current WPT system are quite large which makes this solution only practical for larger UAVs. A direct contact approach scales much better in this regard.

3D modelled prototypes of the proposed charging station and custom landing gear was presented along with descriptions of the physical systems individual components. A detailed overview of three different electrical systems used to charge the UAV was presented next. The first system is responsible for managing the electronics onboard the charging platform, the second was the charging circuit used to provide cell balanced charging to the UAV battery and the last is a charge management system that is fitted to the UAV. The charge management system connects the battery to the UAV's power distribution board and charging pads while managing the connections between these interfaces.

Finally charging results of the charging circuit is presented. Figure 7.8 presents a flow diagram summarising the proposed charging procedure.

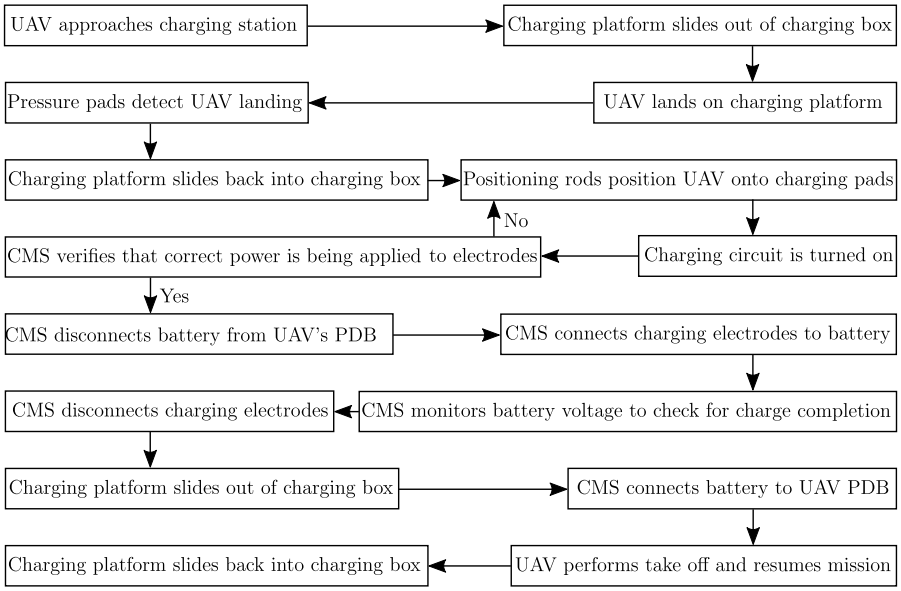


Figure 7.8: Flow diagram of charging procedure

Chapter 8

Improved Localisation

Some UAV applications are highly dependant on localisation accuracy for optimal operation. The localisation accuracy provided by traditional GNSS is not sufficient for these applications. Differential GNSS is one solution to improving accuracy, this technology is however expensive and requires more hardware external to the UAV. This chapter aims to use the onboard camera to improve the UAV's localisation accuracy. The solution aims to be low cost by excluding the need for addition hardware.

In this project highly recognizable recharging stations are placed in the operating environment of the UAV. The UAV uses an onboard camera and OBC to detect these landmarks in real-time. The charging stations contain visual features that can be used to obtain an accurate estimate of the UAV's position relative to the inertial frame. This chapter aims to use these targets with a Simultaneous Localisation and Mapping (SLAM) algorithm to produce and use an accurate map of the charging station locations to localise the UAV in real-time. This solution is one method by which the functionality and use case of the recharging stations can be expanded.

8.1 Proposed Strategy

The proposed solution strategy involves placing at least one, however preferably multiple recharging stations in the UAV's operating environment. In some scenarios it would not make sense to have multiple recharging stations within the same area, either only one UAV is being used that requires recharging, or the operating area is so small that one recharging station is sufficient to service the whole area. In this case it would be beneficial to have only one recharging station aided by multiple other low cost auxiliary visual targets that contain the same recognizable features and visual design as the recharging stations. These auxiliary targets assist by providing more opportunity for the UAV to localise with a mapped target in the environment.

Once these visual markers are in place, a mapping algorithm is used to map each target to the local inertial frame. This map is used by a localisation algorithm to estimate the inertial frame location of the UAV. The mapping process can either be done automatically using a SLAM algorithm or it can be done manually. If manual mapping is used the problem reduces to a pure localisation problem as further mapping is not required. Both manual and automatic mapping approaches are discussed in Sections 8.2.1 and 8.2.2.

Improving localisation using an external vision (EV) system consists out of two main subsystems. The first is the SLAM algorithm that estimates the UAV's local position using the

onboard camera and the second is the state estimator that fuses the EV estimate with other onboard sensors to obtain the position states of the UAV. The SLAM algorithm is dependant on real-time image processing which is implemented on the OBC. The SLAM algorithm is run as a ROS node which publishes data to a MAVROS topic. The state estimator subscribes to this topic to receive the data. PX4 enables external position fusion from ROS or other MAVLink enabled systems. To do this the correct topic name as well as the time delay of the EV measurement relative to the equivalent time registered by the IMU clock needs to be specified.

Figure 8.1 show three different actions or scenarios that make up the proposed improved localisation system.

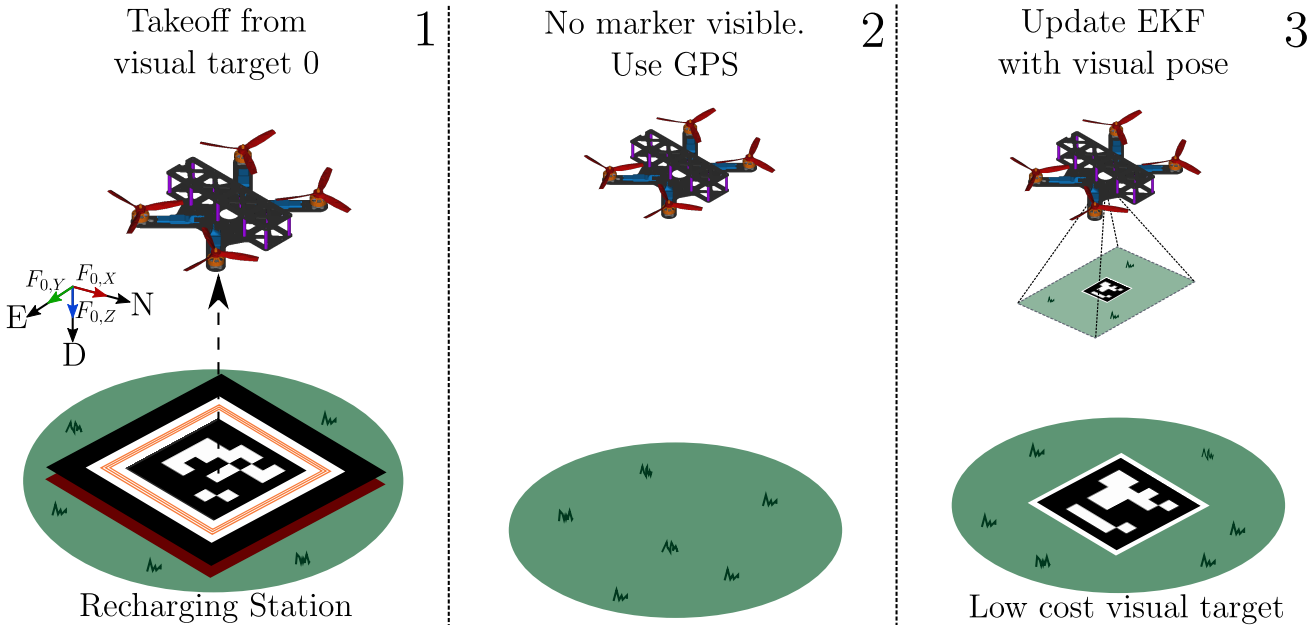


Figure 8.1: Fiducial SLAM system

In action 1, the UAV takes off from visual target 0 ($fid0$). This is the first marker detected during a flight. This marker is manually mapped to location with coordinates $I = \{0, 0, 0\}$ and orientation $\alpha_I = \{0, 0, \psi_m\}$, where ψ_m is the yaw offset between the markers rotation around its defined z axis and the world frames North direction. This results in the inertial frame being aligned with the SLAM maps coordinate frame. If this first marker is not manually mapped, then the map origin is set to the UAV body's location upon the first markers detection. This results in an undesired offset between the map frame and the inertial frame.

In action 2, the UAV is in a location where no markers are detected. This scenario is inevitable unless the marker density within the environment is sufficiently high. This scenario places a limitation on the effectiveness of the proposed SLAM implementation, however several solutions are presented in Section 8.2.2. The estimated position of the UAV will still be more accurate if a marker was recently detected opposed to no markers being detected at all.

In action 3, the UAV detects a low cost auxiliary marker, these are used to increase the density of markers within the UAV's environment. This marker can however be replaced with another recharging station if desired. The pose of the camera relative to the map/inertial frame origin is calculated. This visual pose estimate is published to a MAVLink topic where PX4's state estimator (EKF2) updates the UAV's local position state.

8.2 Mapping

The fiducial marker based SLAM package `Fiducial SLAM` [66] automatically creates a map recording the individual marker locations and their variance in the inertial frame. This process can however also be done manually. The practicality of each method in the context of this project is investigated in the following subsections.

8.2.1 SLAM Mapping

Suppose a scenario where two fiducial/ArUco markers namely: $fid0$ and $fid1$ are placed at fixed locations in an environment. In this context $fid0$ has initially been mapped to the inertial frame origin. The map specifying each unique ArUco's 6 DoF pose is updated by observing pairs of markers. A new fiducial is added to the map by observing it together with an already mapped fiducial.

Given that the inertial frame 6 DoF pose of $fid0$ is already known. This is represented as a transform, T_map_fid0 from the map frame origin to the marker's coordinate system. Then $fid1$'s pose is added to the map using:

$$T\_map\_fid1 = T\_map\_fid0 \times T\_cam\_fid1 \times T\_cam\_fid0^{-1} \tag{8.1}$$

where $T\_cam\_fid_i$ is the transform of fiducial marker i's coordinate system to the camera's coordinate system. This transform is calculated by solving a set of linear equations upon detection as described by Section 6.5.1. This method is used to build up a map of all markers as more marker pairs are observed.

Each marker observation has a variance or uncertainty value attached. Each markers pose in the map is estimated and updated using a combination of multiple observations. Each observation is weighted according to its variance. The new pose of a marker is updated using a Kalman gain K. The Kalman gain represents how good a new measurement is compared to the existing one. The Kalman gain provides a weighting of how far towards the new estimated position to move. The Kalman gain is calculated using two variances $var1$ and $var2$ as follows:

$$K = \frac{var1}{var1 + var2}. \tag{8.2}$$

From this the new position of the marker is continuously updated with new observations using:

$$\mathbf{P}'_1 = \mathbf{P}_1 + K(\mathbf{P}_2 - \mathbf{P}_1), \tag{8.3}$$

where $\mathbf{P}_2$ is the estimated transform of a new observation using Equation 8.1. $\mathbf{P}_1$ represents the current pose of the marker and $\mathbf{P}'_1$ represents the updated pose.
From inspecting the codebase, $var2$ is calculated according to how well the UAV is upright on the ground such that their is minimal tilt between the marker and camera. This is more formally expressed as:

$$var2 = s1 + s2 + s3, \tag{8.4}$$

where,

$$s1 = |\mathbf{P}_I| P_\theta^2, \tag{8.5}$$

here $\mathbf{P}_I$ represents the estimated inertial frame pose of the UAV relative to the map origin. P_θ represents the estimated roll angle of the UAV.

$$s2 = |\mathbf{P}_I| P_\phi^2, \tag{8.6}$$

where P_ϕ represents the pitch angle of the UAV and,

$$s3 = \left(\frac{P_{I,Z}}{P_{cf,Z}}\right)^2 (P_{cf,X}^2 + P_{cf,Y}^2). \tag{8.7}$$

$P_{I,Z}$ represents the z component of the UAV's inertial frame pose and P_{cf} represents the pose of the camera relative to the marker.

$var1$ is calculated by combining two Gaussian probability densities at the new Kalman estimated mean using the mean and variance of both distributions. The first distribution, $N(\mu_2, \sigma_2)$ has variance $\sigma_2 = var2$ and assumes a 1-dimensional space between $\mathbf{P}_1$ and $\mathbf{P}_2$, from this the mean μ_2 of the first distribution is calculated as $\mu_2 = |\mathbf{P}_2 - \mathbf{P}_1|$.

The second distribution, $N(\mu_1, \sigma_1)$ has variance $\sigma_1 = var1$ which represents the variance of the new Kalman filtered estimate and has mean $\mu_1 = 0$. The probability density at the new mean μ_{new} of both distributions is calculated using:

$$N(\mu_{new}, \mu_i, \sigma_i) = \frac{1}{\sigma_i \sqrt{2\pi}} ^{-\frac{(\mu_{new}-\mu_i)^2}{2\sigma_i}}, \tag{8.8}$$

where $\mu_{new} = |K(\mathbf{P}_2 - \mathbf{P}_1)|$ and σ_i and μ_i represents the variance and mean of both distributions denoted by the subscript i. From this the two probability densities are combined using the sum in quadrature of both probabilities as follows:

$$N(\mu_{new}, \sigma_1) = \sqrt{N(\mu_{new}, \mu_1, \sigma_1)^2 + N(\mu_{new}, \mu_2, \sigma_2)^2}. \tag{8.9}$$

Now $var1$ is calculated using:

$$var1 = \left(\frac{1}{N(\mu_{new}, \sigma_1)\sqrt{2\pi}}\right)^2. \tag{8.10}$$

$var1$ is additionally bound by a maximum and minimum value in order to prevent blow up.

8.2.2 Mapping Limitations

In a typical scenario the recharge stations will be spread out over a large area. This makes observing new marker pairs difficult due to the limitations of the camera's field of view. The UAV would need to ascend to high altitude to observe a marker pair located far apart. At high altitude pose estimation becomes unreliable as the markers appear small and indistinguishable. This causes large errors in the pose estimation of the markers which reduces the efficiency of this approach. Some solutions to this problem are listed as follows:

- Increased camera resolution and field of view using a different camera.
- Increased marker size.
- Increase marker density. Placing several low cost auxiliary markers closer to one another removes the need to fly at high altitude to observe marker pairs.
- Manual mapping. Manually mapping the charging station locations prior to flight reduces the problem to pure localisation.

Increased Camera Resolution and Field of View

Increased camera resolution allows for more feature preservation on small targets due to the increased number of pixels on the image. Increasing the cameras field of view would lower the required altitude needed to observe a new marker pair. However both solutions are only only effective up to a certain distance and does not provide a solution to all scenarios. More processing power is required to process images with larger resolution and thus could impact performance. These changes can however still be implemented to increase the overall effectiveness.

Increased Marker Size

The pose estimation error when the UAV is at high altitude can be minimised by increasing the physical marker size. As the marker is located on top of the charging station as shown previously in Figure 7.2, the size of the marker is limited to the size of the charging box unless the marker is allowed to extend over the box. The increase in marker size on its own does not provide an effective solution to all scenarios as it is only effective up to a certain altitude.

The increase in marker size does also make the solution less practical as they would require larger open space which is not possible in all scenarios. The effectiveness of this solution can be increased when combined with the previously mentioned solutions: increased camera resolution and field of view.

Increased Marker Density

By adding more low cost auxiliary markers to the environment, the UAV would not need to fly to such extreme altitudes to observe new marker pairs. These markers do not necessarily need to have recharging capability and are thus low cost. The UAV would however still need to fly to considerable altitude depending on the density of the markers.

A simulation experiment was setup where the feasibility of this approach was investigated. In the experiment two markers were placed at varying distances from each other. Given that one of the markers was mapped accurately the experiment determined how accurately a second marker could be mapped from the first. The distance between the markers represents the marker density in the environment. The UAV had to ascend to varying distances to observe the marker pair. Once at the required altitude the UAV was left hovering for 10 seconds so that multiple observations could be gathered, thereby increasing mapping accuracy. Table 8.1 presents the SLAM simulation results comparing marker density to SLAM mapping accuracy with the required altitude as the determining factor to mapping accuracy. In this experiment a realistic marker size of 1x1 m was used.

Marker Seperation Distance (m)	**Required Altitude (m)**	**Mapping Accuracy (%)**
5	5	99.90
10	10	98.85
20	15	97.44
30	20	93.93
50	34	85

Table 8.1: Mapping accuracy vs marker density

The results show that the markers could be accurately mapped at large separation distances. This experiment included a realistic amount of camera noise on the images for a more realistic result.

During the last two tests, specifically at 20 m and 34 m altitude the individual observations fluctuated considerably. It would thus not be possible to do localisation at these altitudes but are sufficient for mapping provided enough time is allowed for multiple observations to be obtained. These results show that mapping with increased marker density is a viable solution. Combining this solution with increased camera resolution. field of view and marker size can further improve the effectiveness of this solution.

Manual Mapping

In some scenarios none of the previously mentioned automatic mapping solutions might be possible. In that case the markers can be mapped manually. One possible method of doing this involves taking multiple GNSS measurements of a marker's location. Finding the origin of all the measurements and transforming these to the local NED frame. This allows the markers location to be estimated using traditional GNSS.

This process can be automated by recording the current GNSS location of a charging station each time the UAV performs an autolanding. Over time the recharging station can be mapped automatically using the UAV's GNSS.

8.3 Localisation

Given that all markers have been successfully mapped, the UAV's 6 DoF pose relative to the map origin can be determined from the transform of the map coordinate system to the fiducial's coordinate system, $T\_map\_fid_i$ and the determined transform of the fiducial's coordinate system to the camera's coordinate system, $T\_fid_i\_cam$. The UAV's pose, T_map_UAV is calculated using:

$$T\_map\_UAV = T\_map\_fid_i \times T\_fid_i\_cam \tag{8.11}$$

Once the world 6 DoF pose of the UAV is estimated by the ROS based SLAM algorithm, PX4's EKF2 is used to fuse this data with its current sensors including GNSS to achieve a more accurate inertial frame position estimate.
Figure 8.2 illustrates the localisation process. It shows how the SLAM output message is passed to PX4's EKF2 using the publish and subscribe architecture. Figure 8.2 also illustrates the map that is generated. The green markers are the mapped markers that are currently observable whereas the red marker is not. The blue lines demonstrate the links between markers. A link connects the marker pairs that were observed simultaneously to either add a new marker to the map or to update the pose estimate of a marker.

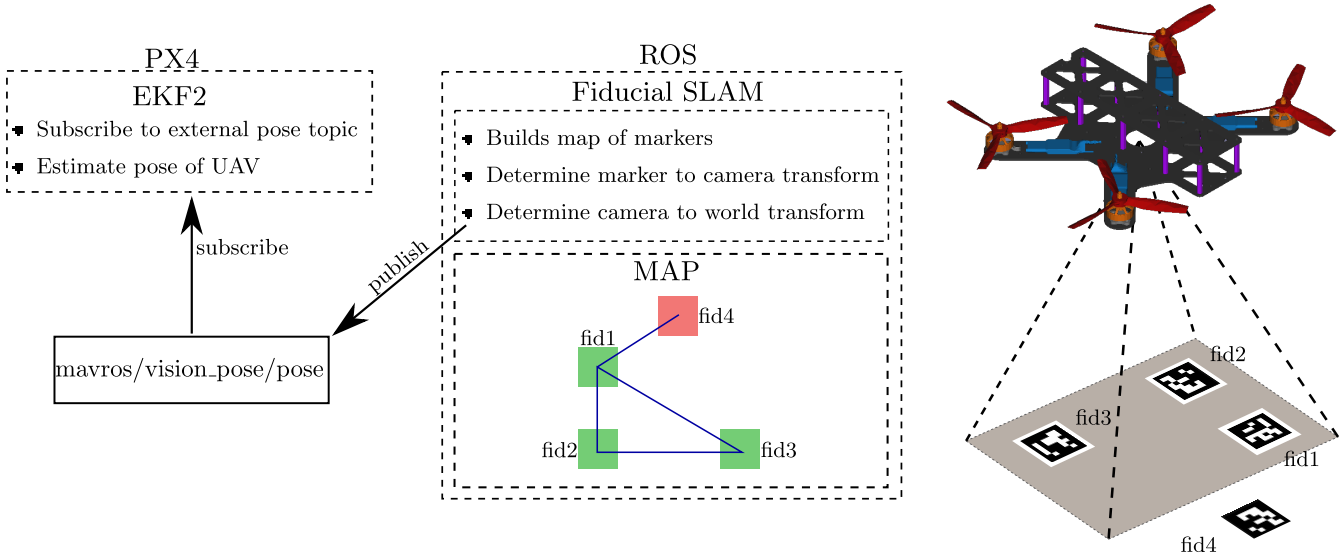


Figure 8.2: Localisation process flow diagram

8.4 PX4 State Estimator

The 6 DoF pose estimate from the SLAM system needs to be fused with the current state measurements of the UAV. PX4's state estimator is used to do this. An overview of the state estimator is firstly provided before the integration of the SLAM measurements are discussed. Figure 8.3 provides a block diagram of PX4's state estimator.

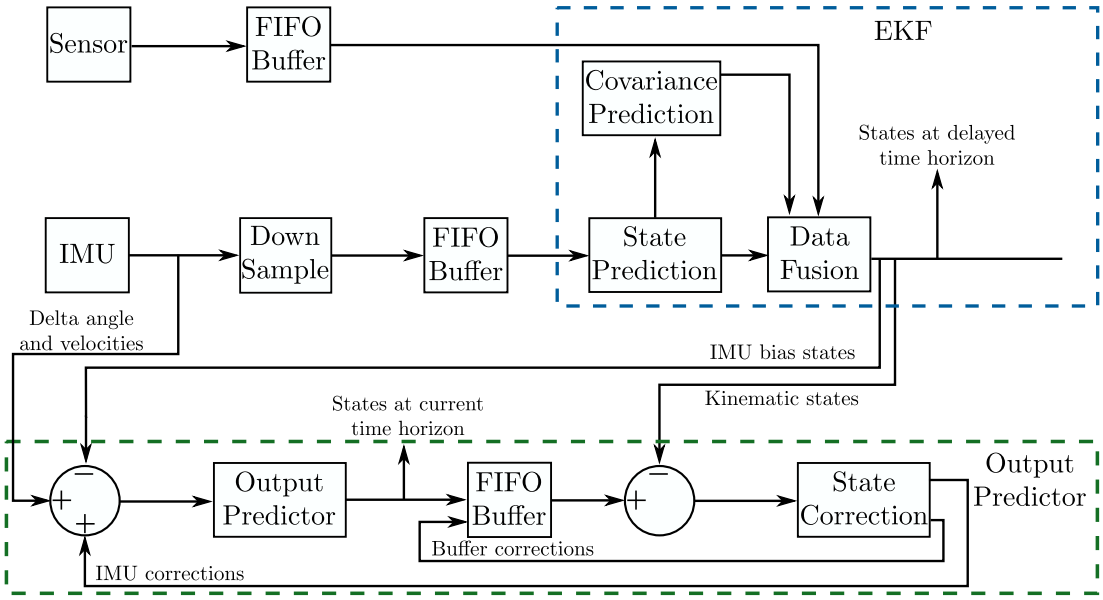


Figure 8.3: PX4 state estimator block diagram

PX4's state estimator is a strapdown inertial navigation system (INS) extended Kalman filter (EKF) with a delayed time horizon. The delayed time horizon architecture is used to account for the different sampling times of the UAV's various sensors. The INS primarily makes use of the IMU's accelerometer and gyroscope to determine the position, velocity and orientation of the UAV using a state prediction process. These measurements are combined with with external sensors such as GNSS, barometer and external vision which help in correcting

errors in the state prediction process. PX4's INS state estimator is based off of work done by [70, 71, 72, 73].

PX4's state estimator contains 24 states, namely:

$$\mathbf{x} = \begin{bmatrix} \text{Orientation}(q_0, q_1, q_2, q_3) \\ \text{Velocity}(V_N, V_E, V_D) \\ \text{Position}(N, E, D) \\ \text{Gyro angle bias}(g_{x,b}, g_{y,b}, g_{z,b}) \\ \text{Accelerometer bias}(a_{x,b}, a_{y,b}, a_{z,b}) \\ \text{Earth magnetic field vector}(m_N, m_E, m_D) \\ \text{Wind velocity}(w_N, w_E) \end{bmatrix}. \tag{8.12}$$

The only states that are updated by the EV system in this project is the inertial frame position. The discrete time state transition and output functions are given as:

$$\begin{aligned} \mathbf{x}_k &= g(u_k, \mathbf{x}_{k-1} + \varepsilon_k) + \varepsilon_{static} \\ \mathbf{z}_k &= h(\mathbf{x}_k) + \delta_k \end{aligned} \tag{8.13}$$

where $\mathbf{x}_k$ is the discrete time state vector and $\mathbf{z}_k$ is the discrete time output vector where k represents the discrete time step. g is the non-linear state transition function and h is the non-linear output function. ε_k is the process noise of the IMU, ε_{static} is static systematic noise and δ_k represent the measurement noise. The relevant blocks of Figure 8.3 are discussed in the following subsections.

8.4.1 Sensor Fusion

The first 10 states of Equation 8.12, namely: orientation, velocity and position are determined by integrating IMU measurements. [67] shows how this is accomplished. External non-IMU sensors such as barometer, GNSS and EV can be fused to update the estimated states. This is done by firstly calculating the innovation variance using the covariance matrix, $\mathbf{\Sigma}_k$ as:

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{\Sigma}_k \mathbf{H}_k^T + \mathbf{R}_k, \tag{8.14}$$

where $\mathbf{H}_k$ is the Jacobian of the measurement function and $\mathbf{R}_k$ represents the measurement noise.

A Kalman gain, $\mathbf{K}_k$ determines by how much non-IMU sensors should be used to update the state measurements. The Kalman gain is calculated using:

$$\mathbf{K}_k = \mathbf{\Sigma}_k \mathbf{H}_k \mathbf{S}_k^{-1}. \tag{8.15}$$

The states are then updated with the Kalman gain using:

$$\mathbf{x}'_k = \mathbf{x}_k - \mathbf{K}_k(\mathbf{x}_k - z_k) \tag{8.16}$$

where z_k represents a new sensor measurement. The covariance matrix is then updated using the latest Kalman gain.

8.4.2 Output Predictor

The various sensors used by PX4 all have different sampling rates. This poses a problem of how to combine current sensor measurements with delayed measurements. A traditional EKF solves this problem by fusing delayed states with delayed measurements and the current

covariance matrix. This however leads to instability and inaccuracy.

PX4's EKF2 improves this by introducing a delayed time horizon which fuses sensor measurements, states and covariance matrix from the same point in time. The delayed states are then predicted forward into the current time horizon using a complementary filter. This approach is based of of work done by [69].

8.4.3 External Vision Fusion

The delayed time horizon architecture allows for any combination of sensors to be added to PX4's state estimator. This enables EV systems such as marker based SLAM. The relative time delay between the vision system and the equivalent UAV's IMU measurements need to be specified in order for the EKF to incorporate the EV measurements into the delayed time horizon state prediction. The EKF also needs to subscribe to the correct topic and thus this needs to be specified [68].

8.5 Results

The SLAM implementation was firstly developed and tested in a SITL and HITL Gazebo simulation. Here the SLAM output was compared to a ground truth measurement to test the correctness of the SLAM output. This simulation was also used to test the practicality of the solution by making it as realistic as possible. The SLAM system was then incorporated into PX4's state estimator in a SITL simulation before performing practical testing.

8.5.1 Fiducial SLAM Verification

The aim of this simulation experiment was to to test the correctness and accuracy of the SLAM system by comparing it to a ground truth estimate. The simulated experiment was designed to be as realistic as possible by including camera noise while requiring the UAV to automatically map the markers prior to the test. In this test three markers are placed 5m apart. The UAV takes off from the first marker mapped to the map frame origin. This aligns the map frame origin with the inertial frame origin. The UAV then climbs to an altitude of 10 m and makes a single slow pass over the other two markers. This maps the remaining two markers before obtaining the results shown in Figure 8.4.

Figure 8.4 compares the recorded SLAM output to a ground truth measurement. The results of this experiment show the accuracy of the SLAM system given that the markers were previously mapped as described above. The results show the UAV flying from $fid0$ to $fid1$ to $fid2$ and then back again at an altitude of 6 m. At this altitude it is not possible to observe a marker pair, this prevents the map from being updated with new observations. This was done to test the accuracy of the mapping method described above as this is how markers will be mapped in a practical scenario.

The results show that the SLAM system is accurate compared to the ground truth estimate. In some cases the measurements deviate slightly, this is due to mapping inaccuracy but can be improved with more observations. The periods where no SLAM data is received occur when the UAV is transitioning to a new marker and no markers are fully visible. These instances are shown as the periods where the SLAM output measurement does not change, such as t = 5-10 s.

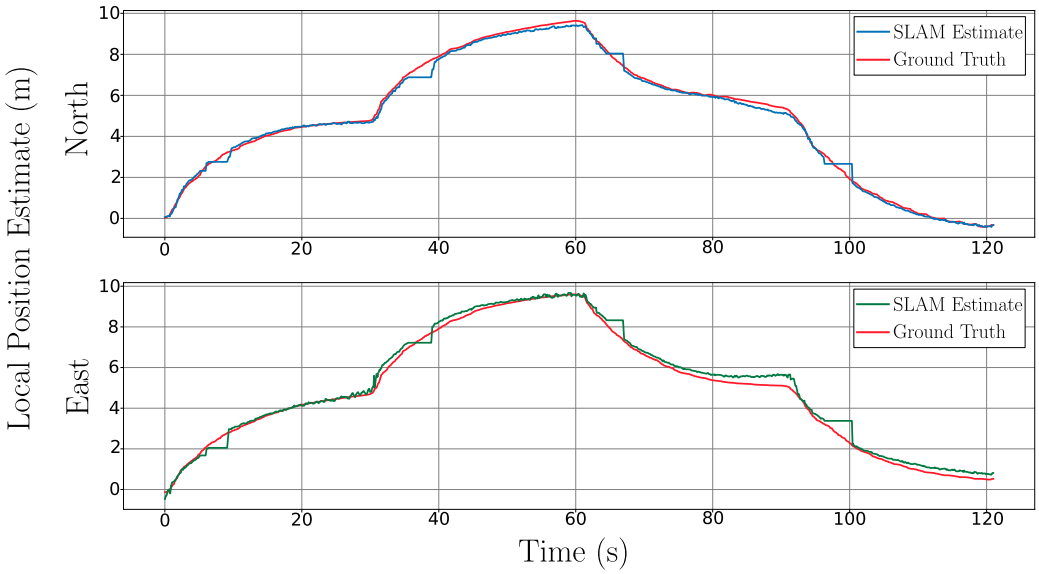


Figure 8.4: Comparing SLAM output to ground truth

8.5.2 Practical SLAM with EKF Fusion

A practical test was setup where the aim was to test the position estimation performance of PX4's EKF2 with external vision fusion. The Jetson Nano with the onboard camera was responsible for executing the SLAM software. The SLAM output was published to the flight controller where it was combined with onboard GNSS and IMU measurements to estimate the position of the UAV. The goal was to improve the position estimate of the UAV by fusing EV data. The results therefore present a comparison between using only GNSS (Figure 8.7) and GNSS with EV fusion (Figure 8.6) to show the improvement.

In this practical experiment no ground truth measurement was available. The UAV had to be kept stationary in an unarmed state in order to make a valid comparison between the SLAM output and the inaccuracy of GNSS. A mapped marker was placed in the view of the onboard camera in order to obtain an accurate SLAM estimate of the UAV's inertial frame position. A diagram illustrating the experimental setup is shown in Figure 8.5.

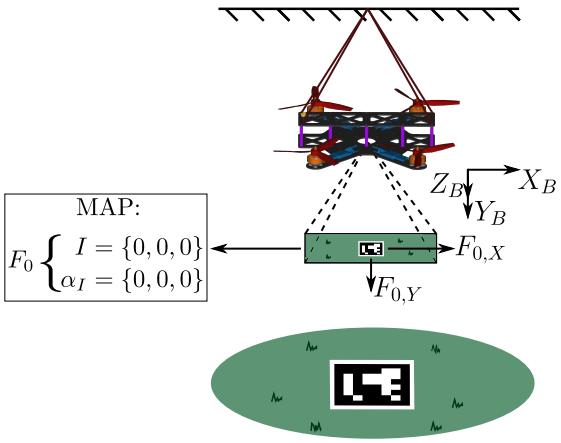


Figure 8.5: Practical SLAM experiment setup

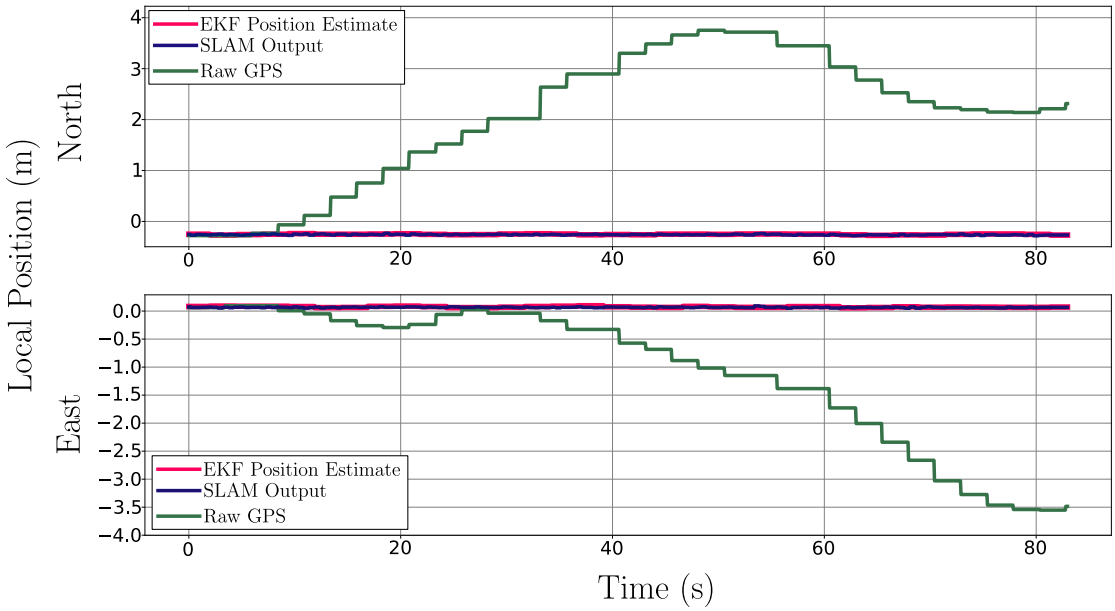


Figure 8.6: Improved localisation using external vision

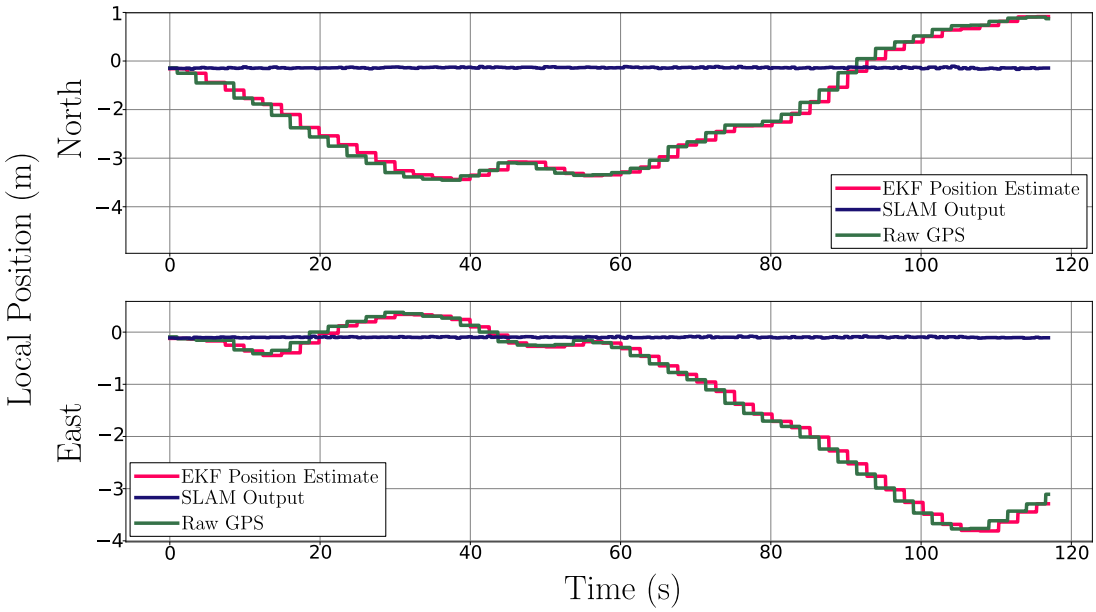


Figure 8.7: Default localisation method using only GNSS

Referring to Figures 8.6 and 8.7 it is observed that the SLAM output remains stationary as expected, the GNSS measurements however have a low frequency drift causing large inaccuracy. In Figure 8.6 it can be observed that the estimated position of the UAV closely follows the EV output. In Figure 8.7 the SLAM output was not published to the flight controller, this results in the position estimate to follow inaccurate GNSS. This test confirms that the localisation accuracy can be improved using an EV system such as marker based SLAM, provided a marker is clearly visible and that it has been accurately mapped.

8.6 Summary

This chapter investigated whether the recharging stations presented in Chapter 7 could be used to also improve the general localisation accuracy of the UAV. No additional hardware was required to accomplish this, making it inexpensive compared to other methods such as differential GNSS. A marker based SLAM algorithm was used to create an accurate map of the markers located on the charging stations relative to the inertial frame origin. The output of this system was published to the flight controllers state estimator which fuses EV data with IMU and onboard GNSS data to create a more accurate local position estimate of the UAV.

The chapter also discussed the method used to map all fiducial targets within the environment. It was determined that automatically mapping new recharging stations would be difficult due to the practical distance between stations. This limitation was discussed, and several solutions to this problem was provided. The most promising solutions were to either manually map the charging stations or to increase the density of markers by placing more low cost auxiliary markers in the operating environment. It was determined that these markers could still be accurately mapped with a marker separation distance of 30 m provided the UAV is able to gather enough observations.

The chapter provides a brief description of PX4's EKF2 which was used to fuse the SLAM output with the other sensors of the UAV to obtain a more accurate local position estimate. Both simulation and practical results are presented. These results showed that the SLAM algorithm provides an accurate estimate of the UAV's local position and that when used by the UAV's state estimator the local position estimate of the UAV closely follows the SLAM output, thereby improving localisation accuracy.

Chapter 9

Conclusion

This thesis addressed the problem of achieving automated self recharging of multirotor UAVs. The proposed recharging solution had to be suitable for most industrialized UAVs which would enable them to self recharge with minimal change to the UAV's physical hardware. A method of accurately landing on the charging station using an inexpensive monocular camera is also addressed. Additionally, this thesis investigated methods by which the charging stations could be used to improve the general localisation accuracy of the UAV.

9.1 Proposed Solution

9.1.1 UAV Development

A custom UAV was commissioned in order to develop and practically demonstrate the proposed solution. PX4 open source flight control software running on a Pixhawk mini was chosen as the avionics of the UAV. PX4's controllers, together with a derived model of the UAVs dynamics was implemented in MATLAB/Simulink in order to design custom controller gains suitable for the purpose of this project. A model of the UAV was also implemented in Gazebo simulator where the custom controller response was verified in both SITL and HITL simulations. Thereafter, practical flight tests were conducted to verify that the practical flight response was acceptable and matched the designed response. These flights were successful and confirmed that the simulation accurately modelled the practical system.

9.1.2 Autolanding

The proposed solution involves using the UAV's GNSS to fly to the known GNSS location of the charging station. At this point the UAV makes use of an onboard computer which processes onboard camera images to identify a custom marker placed on the charging station. The estimated pose of the landing platform is published to the flight controller where a custom controller tracks the landing platform using visual position feedback. An autolanding state machine is responsible for guiding the UAV from high altitude to low altitude while providing continuous fault detection and landing repair features.

Simulation results showed that to guarantee landing accuracy, the UAV needs to get as close as possible to the landing target while maintaining visual feedback control. This minimum altitude constraint is determined by the field of view of the camera and the marker size. The designed marker thus included a large outer marker for high altitude localisation and a smaller inner ArUco marker for the final landing stages. With this marker design, an average practical landing accuracy of 4 cm was achieved.

9.1.3 Charging Station

Three charging methods for UAVs was identified from literature, namely: direct contact approach, wireless power transfer and a battery swap approach. This thesis proposed a direct contact based recharging approach. The design includes a charging platform that is able to freely retract in and out of a protective enclosure. This protects not only the charging platform but also the UAV during charging or when not in use.
The charging platform design includes a mechanism that is used to accurately move the UAV to the required charging location on the platform. Custom landing gear and a charge management circuit are the only additional parts that would need to be fitted to a UAV to enable self recharge. The UAV's battery and power distribution board both connect to the charge management circuit which manages the connections between the charging circuit, UAV and UAV's battery. The charging circuit will provide balanced charging which would ensure safety and longer battery life time. The charging station could also be deployed in remote locations by making use of a solar panel and battery for power supply.

A practical charging time of 65 minutes was achieved using the proposed balanced charging circuit. A faster charging time could be achieved by increasing the supply current of the charging circuit however this decreases the overall battery life time.

9.1.4 Improved Localisation

To improve the existing GNSS based localisation system of the UAV this thesis proposes a method by which the charging stations can be used as visual landmarks within the environment. If the exact location of these landmarks are known then the UAV is able to estimate its own position within the same reference frame. A marker based SLAM approach is used to map and simultaneously localise the locations of the markers located on the charging stations. A Kalman filter approach is used to estimate the marker positions using a combination of multiple observations. The SLAM output is then published to PX4's state estimator where it is fused with IMU measurements to obtain a more accurate inertial frame pose estimate of the UAV.

A simulation was created that tested the mapping accuracy of the proposed solution. Results showed that the markers could be mapped with 85% accuracy at an altitude of 34 m and 93% accuracy was achieved at 20 m altitude. Thereafter, a practical experiment was conducted which aimed to test position estimation performance of PX4's EKF when combined with the visual SLAM algorithm. The experiment showed that the localisation accuracy of the UAV could be greatly improved to centimetre level accuracy compared to a GNSS inaccuracy of up to 4m.

9.2 Future Work

In order to improve the system further and allow for continuation of this project, possible future work is:

- Night time landing. The current marker design and landing strategy only allows for daytime landings. This places a large limit on when autonomous UAVs can be operational. The existing marker could be augmented through the inclusion of infrared LEDs placed in a non symmetrical pattern. The UAV could make use of an infrared camera such as the PixArt IR camera to identify the landing marker in night time conditions while still remaining cost effective.

- Build practical charging station. In this project the charging station was not fully built. To continue the project the full charging station, including the landing gear could be manufactured and tested.
- Attempt full practical charging process from landing, charging and takeoff.
- Multiple charging station network management system. With multiple charging stations spread over a large area an onboard charging management system needs to be developed that would allow UAVs to use the network of charging stations to optimise flight time and battery capacity. Multiple UAVs would need to communicate to cooperatively manage charging resources in an optimal way.
- Perform large scale practical marker based SLAM on an airborne UAV with a ground truth measurement such as differential GNSS or motion capture (MoCap) system. This way the localisation accuracy improvement can be more accurately quantified in a more realistic practical scenario.

Bibliography

[1] United States Patent and Trademark Office, Patent no: US9387928B1, July 12 2016, [Online]

[2] L. Li, L. Sun, and J. Jin, "Survey of advances in control algorithms of quadrotor unmanned aerial vehicle," International Conference on Communication Technology Proceed- ings, ICCT,vol.2016-February,pp. 107–111,2016. Available at: https://ieeexplore.ieee.org/document/7399803

[3] P.A Beardsley, M Eriksson, J Alonsomora and J Rehder, " Robust and autonomous docking and recharging of quadrotors". [Online]. Available at: https://patents.google.com/patent/US9573701B2/en.

[4] R Clarke, "Autonomous Multi-Rotor Aerial Vehicle with Landing and Charging System". [Online]. Available at: https://patents.google.com/patent/US20170139409A1/en.

[5] "Skycharge High Power Drone Charging Pad And Infrastructure" [Online]. Available at: https://skycharge.de/

[6] A.B Junaid, A Konoiko, Y Zweiri, M.N Sahinkaya and L Seneviratne. "Autonomous Wireless Self-Charging for Multi-Rotor Unmanned Aerial Vehicles" 2017. [Online]. Available at: https://www.mdpi.com/1996-1073/10/6/803

[7] A.B Junaid, Y Lee, Y Kim. "Design and implementation of autonomous wireless charging station for rotary-wing UAVs". [Online]. Available at: https://www.sciencedirect.com/science/article/pii/S1270963816301547

[8] A Kurs, A Karalis, R Moffatt, J.D Joannopoulos, P Fisher and M Solja ˇci´c. "Wireless power transfer via strongly coupled magnetic resonances. [Online]. Available at: https://science.sciencemag.org/content/317/5834/83/tab-pdf

[9] M. Lu, M. Bagheri, A. P. James and T. Phung, "Wireless Charging Techniques for UAVs: A Review, Reconceptualization, and Extension," in IEEE Access, vol. 6, pp. 29865-29884, 2018, doi: 10.1109/ACCESS.2018.2841376.

[10] L Zhi-ning, L Xia-qing, Y Lu-jian, D Leo and Z Hong-wei, "An Autonomous Dock and Battery Swapping System for Multirotor UAV".

[11] T Toksoz, J Redding, M Michini, B Michini and J.P How, "Automated Battery Swap and Recharge to Enable Persistent UAV Missions".

[12] December 5 2017, GPS.gov, GPS Accuracy, Available at: https://www.gps.gov/systems/gps/performance/accuracy/, accessed on October 23 2019

[13] A.D Swart, "Monocular Vision Assisted Autonomous Landing of a Helicopter on a Moving Deck". [Online]. Available at: http://scholar.sun.ac.za/handle/10019.1/80134

[14] S. Saripalli, J. F. Montgomery and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), Washington, DC, USA, 2002, pp. 2799-2804 vol.3, doi: 10.1109/ROBOT.2002.1013656.

[15] R. Ginkel, I. Meerman, T. Mulder and J. Peters: "Autonomous Landing of a Quadcopter on a Predefined Marker".

[16] S. Yang, S. Scherer, A Zell, "An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle"

[17] K.E Wenzel, P. Rosset, A. Zell., "Low-Cost Visual Tracking of a Landing Place and Hovering Flight Control with a Microcontroller", Journal of Intelligent and Robotic Systems, 2009, Vol. 57(1-4), pp. 297-311

[18] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle," Journal of intelligent & robotic systems, vol. 61, no. 1, pp. 221–238, 2011

[19] R. Polvara, S. Sharma, J. Wan, A. Manning and R. Sutton, "Towards autonomous landing on a moving vessel through fiducial markers," 2017 European Conference on Mobile Robots (ECMR), Paris, 2017, pp. 1-6, doi: 10.1109/ECMR.2017.8098671.

[20] M.F Sani, G Karimian, "Automatic Navigation and Landing of an Indoor AR. Drone Quadrotor Using ArUco Marker and Inertial Sensors".

[21] B. Nelson, J. D. Preez, T. van Niekerk, R. Phillips and R. Stopforth, "Autonomous Landing of a Multirotor Aircraft on a Docking Station," 2020 International SAUPEC/RobMech/PRASA Conference, Cape Town, South Africa, 2020, pp. 1-6, doi: 10.1109/SAUPEC/RobMech/PRASA48453.2020.9041028.

[22] A. Marut, K. Wojtowicz and K. Falkowski, "ArUco markers pose estimation in UAV landing aid system," 2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace), Torino, Italy, 2019, pp. 261-266, doi: 10.1109/MetroAeroSpace.2019.8869572.

[23] OpenCV, 'Detection of ArUco Markers', [Online], Available at: https://docs.opencv.org/3.4.10/d5/dae/tutorial_aruco_detection.html

[24] B. Pervan, F.C Chan, D. Gebre-Egziabher, S. Pullen, P. Enge and G. Colby: "Performance Analysis of Carrier-Phase DGPS Navigation for Shipboard Landing of Aircraft". Journal of The Institute of Navigation, 2003. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.194.5925&rep=rep1&type=pdf

[25] DefenseMediaNetwork: 'Details of the X-47Bs First Autonomous Carrier Landing' [ONLINE]. 2014. Available at: http://www.defensemedianetwork.com/stories/ details-of-the-x-47bs-first-automated-carrier-landing/

[26] C. Lin, W. Wang, S. Liu, C. Hsu and C. Chien, "Heterogeneous Implementation of a Novel Indirect Visual Odometry System," in IEEE Access, vol. 7, pp. 34631-34644, 2019, doi: 10.1109/ACCESS.2019.2904142.

[27] T. Tykkälä, C. Audras and A. I. Comport, "Direct Iterative Closest Point for real-time visual odometry," 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, 2011, pp. 2050-2056, doi: 10.1109/ICCVW.2011.6130500.

[28] J. Engel, V. Koltun and D. Cremers. “Direct Sparse Odometry”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018.

[29] A. Barsan. “DYNSLAM: Robust Dense Mapping For Large-Scale Dynamic Environments”. [Online]. Available at: https://siegedog.com/dynslam/

[30] T. Yang, P. Li, H. Zhang, J Li and Z. Li. “Monocular Vision SLAM-Based UAV Autonomous Landing in Emergencies and Unknown Environments”. [Online], Available at: https://www.mdpi.com/2079-9292/7/5/73

[31] I. Alzugaray, L. Teixeira and M. Chli, ”Short-term UAV path-planning with monocular-inertial SLAM in the loop,” 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 2739-2746, doi: 10.1109/ICRA.2017.7989319.

[32] A. Handa, T. Whelan, J. McDonald and A. J. Davison, ”A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM,” 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 1524-1531, doi: 10.1109/ICRA.2014.6907054.

[33] R. Mur-Artal, J.M.M. Montiel and J.D. Tardos. “ORB-SLAM: a Versatile and Accurate Monocular SLAM System”, 2015, IEEE Transactions on Robotics.

[34] L. Jayatilleke and N. Zhang, ”Landmark-based localization for Unmanned Aerial Vehicles,” 2013 IEEE International Systems Conference (SysCon), Orlando, FL, 2013, pp. 448-451, doi: 10.1109/SysCon.2013.6549921

[35] W. Roozing and A. H. Göktoğan, ”Low-cost vision-based 6-DOF MAV localization using IR beacons,” 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Wollongong, NSW, 2013, pp. 1003-1009, doi: 10.1109/AIM.2013.6584225.

[36] T.T. Mac, C. Copot, R.D. Keyser, C.M. Ionescu. “The development of an autonomous navigation system with optimal control of an UAV in partly unknown indoor environment”. Mechatronics, 49 (2018), pp. 187-196

[37] Hyon Lim and Young Sam Lee, ”Real-time single camera SLAM using fiducial markers,” 2009 ICCAS-SICE, Fukuoka, 2009, pp. 177-182.

[38] DJI Developer, Software Environment Setup Guide, [Online], Available at: https://developer.dji.com/onboard-sdk/documentation/development-workflow/environment-setup.html#linux-with-ros

[39] J.J. Rademeyer, J.A.A Engelbreght and H.A Engelbreght, “Vision-Based Flight Control for a Quadrotor UAV”, [Online], Available at: https://scholar.sun.ac.za/handle/10019.1/108083

[40] J. Blom, C.E. van Daalen and P.G. Wiid. “Accurate Localisation of a Multi-rotor Using Monocular Vision”. [Online], Available at: https://scholar.sun.ac.za/handle/10019.1/103810

[41] Drone Trest, Complete List of Flight Controller Firmware Projects, [Online], Available at: https://blog.dronetrest.com/

[42] PX4 Developer Guide, Companion Computer For Pixhawk Series, [Online], Available at: https://dev.px4.io/master/en/companion_computer/pixhawk_companion.html

[43] ROS.org , What is ROS, [Online], Available at: http://wiki.ros.org/ROS/Introduction

[44] Q. Lin, X. Liu and Z. Zhang, "Mobile Robot Self-LocalizationUsing Visual Odometry Based on Ceiling Vision," 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 2019, pp. 1435-1439, doi: 10.1109/SSCI44817.2019.9003092.

[45] R. Muñoz-Salinas, M. J.Marín-Jimenez and R. Medina-Carnicerab, "SPM-SLAM: Simultaneous localization and mapping with squared planar markers". [Online], Available: https://www-sciencedirect-com.ez.sun.ac.za/science/article/pii/S0031320318303224

[46] Wingtra, Surveying & GIS, [Online], Available at: https://wingtra.com/drone-mapping-applications/surveying-gis/

[47] Airshepard, [Online], Available at: https://airshepherd.org/

[48] Airborne Drones, Surveillance and security drone, [Online], Available at: https://www.airbornedrones.co/surveillance-and-security/

[49] Amazon, Amazon Prime Air, [Online], Available at: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011

[50] Lange,S., Sunderhauf,N., Protzel,P.: "A Vision Based Onboard Approach for Landing and Position Control of an Autonomous Multirotor UAV in GPS-Denied Environments", Journal of Intelligent Robotic Systems (2019), vol. 95, pp. 645-664

[51] Swart, A., Dr Peddle, I.K., "Monocular Vision Assisted Autonomous Landing of a Helicopter on a Moving Deck", Masters thesis, Stellenbosch University, Stellenbosch.

[52] Yang, S., Scherer, S., Zell, A.: "An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle", J. Intell. Robot. Syst. vol. 69, pp. 499–515

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

[54] Goblin Hobbies, "Xnova RM2206-2300KV specifications", [Online]. Available at: https://www.xnovamotors.com/xnova-rm2206-2300kv-specifications/

[55] P.G Ioppo, "The Design, Modelling and Control of an Autonomous Tethered Multirotor UAV" 2017. [Online]. Available at: https://scholar.sun.ac.za/handle/10019.1/101095

[56] A.P. Erasmus, "Stabilization of a Rotary Wing Unmanned Aerial Vehicle with an Unknown Suspended Payload" 2019. [Online]. Available at: https://scholar.sun.ac.za/handle/10019.1/107832

[57] W. Johnson, "Helicopter Theory," 1994

[58] J. Blakelock, Automatic Control of Aircraft and Missiles. Wiley, 1991. [Online]. Available at: https://books.google.co.in/books?id=ubcczZUDCsMC

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

[60] H. Schaub, "Kinematics: Describing the motions of spacecraft," 2018. [Online]. Available at: https://www.coursera.org/learn/spacecraft-dynamics-kinematics

[61] P. D. S. Moller, "Automated Landing of a Quadrotor Unmanned Aerial Vehicle on a translating Platform", 2015. [Online]. Available: https://scholar.sun.ac.za/handle/10019. 1/98014

[62] H. Parwana, M.Kothari, "Quaternions and Attitude Representation", 2017. [Online]. Available at: https://www.researchgate.net/publication/319349983_Quaternions_and_Attitude_Representation

[63] D. Brescianini, M. Hehn, and R. D'Andrea, "Nonlinear quadrocopter attitude control," 2013.

[64] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadro- tors," Proceedings - IEEE International Conference on Robotics and Automation,pp. 2520– 2525, 2011.

[65] J. Vaughan and R. Agrawal, "Aruco Detect ROS Package",[Online], Available at: http://wiki.ros.org/aruco_detect

[66] J. Vaughan and R. Agrawal, "Fiducial SLAM ROS Package",[Online], Available at: http://wiki.ros.org/fiducial_slam

[67] P. Priseborough "PX4 EKF2 Documentation", [Online], Available at: https://github.com/PX4/ecl/tree/master/EKF/documentation

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

[69] A. Khosravian, J Trumpf, R Mahony and T Hamel, "Recursive attitude estimation in the presence of multi-rate and multi-delay vector measurements". [Online]. Available at: https://ieeexplore.ieee.org/document/7171825.

[70] JE. Bortz, "A New Mathematical Formulation for Strapdown Inertial Navigation". [Online]. Available at: https://ieeexplore.ieee.org/document/4103660.

[71] PG. Savage, "Strapdown Inertial Navigation Integration Algorithm Design Part 1: Attitude Algorithms". [Online]. Available at: https://arc.aiaa.org/doi/10.2514/2.4228.

[72] PG. Savage, "Strapdown Inertial Navigation Integration Algorithm Design Part 2: Velocity and Position Algorithms". [Online]. Available at: https://arc.aiaa.org/doi/10.2514/2.4242.

[73] PG. Savage, "Computational Elements for Strapdown Systems". [Online]. Available at: https://www.semanticscholar.org/paper/Computational-Elements-for-Strapdown-Systems-Savage-Plain/da66a66f44eb7d95ced9e700a13af3bfc29a93e6.