UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

NAVIGATION AND CONTROL OF A QUADCOPTER FOR REAL-TIME IMAGE PROCESSING APPLICATIONS

by

Vikesh Govender

Submitted in partial fulfillment of the requirements for the degree
Master of Engineering (Computer Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

June 2022

SUMMARY

NAVIGATION AND CONTROL OF A QUADCOPTER FOR REAL-TIME IMAGE PROCESSING APPLICATIONS

by

Vikesh Govender

Supervisor(s):	Prof H.C. Myburgh
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Master of Engineering (Computer Engineering)
Keywords:	Stability controller, position controller, altitude controller, navigation, image processing, quadcopter, estimation filter.

In recent years, autonomous quadcopters have been gaining a lot of popularity in the research field to perform various image processing applications such as surveillance, package delivery, search and rescue missions, etcetera. However, the most important components of any quadcopter application are the control and navigation of the quadcopter; without these components mentioned, it is difficult to achieve most quadcopter applications. This especially applies to image processing applications, because without an adequate control system the images from a camera will experience disturbances that will hinder the performance of the image processing system. The navigation system is also an important component as it gives the quadcopter the ability to transition between different positions, thus making the quadcopter autonomous.

In several quadcopter-related research work, the research mainly focuses on the image processing application and generally provides an inadequate description of the navigation and control of the quadcopter. While image processing is an important and complex component, it can prove to be unusable if the navigation and control of the quadcopter are of inferior quality. This dissertation focuses on closing the gap and providing a detailed description of the design and implementation of the navigation and control for a quadcopter that can be used for real-time image processing applications.

A stability controller, position controller and altitude controller were developed. A PID controller

was used for the stability controller to ensure that the quadcopter would be stable during flight. A PD controller was used for the position controller and a PID controller was used for the altitude controller. A Kalman filter was used to reduce the noise from the sensors and also to improve data acquisition, since the quadcopter had a fast refresh rate of 3 ms. The quadcopter required a fast refresh rate to prevent it from experiencing an unstable flight. For the position estimation, the Kalman filter also served as a data fusion algorithm to combine data from the accelerometer and the GPS latitude and longitude coordinates.

Furthermore, the dissertation found that the three-blade propellers provided better overall stability; however, it experienced a slightly longer response time. It can be said that the three-blade propellers will perform better for image processing applications, as cameras are largely affected by fast movements and vibrations on the frame. A battery-compensation algorithm was also used, as the quadcopter's motors were affected as the battery voltage was decreasing. Earplugs were installed to reduce the vibrations that were transmitted from the frame to the flight controller board, which was particularly helpful, as the accelerometer can be affected by vibrations. The Caddx Ratel camera was used as the vision sensor for the quadcopter. The Caddx Ratel camera was able to provide the YOLOv2 algorithm with an adequate image stream for further image processing for real-time applications, such as person identification and tracking.

LIST OF ABBREVIATIONS

ANN	Artifical Neural Network
BRIEF	Binary Robust Independent Elementary Features
CNN	Convolutional Neural Network
EKF	Extended Kalman Filter
FAST	Features from Accelerate Segment Test
FOV	Field Of View
FPV	First Person View
GPS	Global Positioning System
GPU	Graphic Processing Unit
HOG	Histogram of Gradients
IMU	Inertial Measurement Unit
IoU	Intersection of Union
KDE	Kernel Density Estimation
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean Squared Error
NMEA	National Marine Electronics Association
NN	Neural Network
ORB	Oriented FAST and Rotated BRIEF
PID	Proportional Integral and Derivative term
PSNR	Peak Signal Noise Ratio
PWM	Pulse Width Modulation
RPN	Region Proposal Network
SIFT	Scale-Invariant Feature Transform
SSIM	Structural SIMilarity
SURF	Speeded-Up Robust Features
UKF	Unscented Kalman Filter
YOLO	You Only Look Once

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 PROBLEM STATEMENT

1.1.1 Context of the problem

In recent years, autonomous quadcopters have been gaining popularity to perform various image processing applications, such as surveillance, package delivery, search and rescue missions, etcetera. In several quadcopter-related research works, the research mainly focuses on the image processing application and generally provides an inadequate description of the navigation and control of the quadcopter. While image processing is an important and complex component, it can prove to be unusable if the navigation and control of the quadcopter are of inferior quality [1].

The navigation and stability of a quadcopter are two important aspects of any quadcopter application. Navigation is the ability for the quadcopter to transition to a certain position or estimate the quadcopter's current position. The stability controller is required to balance the quadcopter's frame when it is in the air, as well as to ensure that transitions are smooth. A well-designed stability controller also allows the quadcopter to behave in a predictable and controlled manner [2]. In most quadcopter applications, navigation and stability are important aspects to take into account, without which it is difficult to actualise an application that employs a quadcopter [1]. This dissertation demonstrates the process of designing and implementing a stability and navigation controller that can be used for real-time image processing applications.

1.1.2 Research gap

The main objective of this research is to implement a navigation and stability controller that can be used in various applications, such as surveillance and package delivery. This dissertation presents the research and development of a stability and navigation controller that can be used for real-time image processing applications for quadcopters.

1.2 RESEARCH OBJECTIVE AND QUESTIONS

The main objective of this research is to design a navigation and control system for a quadcopter that can be used for real-time image processing applications. As previously mentioned, navigation and control of a quadcopter are important in any quadcopter application. Without a stability controller the quadcopter will be difficult to manipulate in the air and the quadcopter will become unpredictable [2]. There are various stability controllers. However, each stability controller has advantages and disadvantages that need to be considered. An appropriate navigation algorithm has to be used to estimate the quadcopter's current and new position. The navigation algorithm is also responsible for instructing the quadcopter's relevant motors to make a transition to the desired position. If the stability controller or navigation algorithm does not perform well, the images from the quadcopter can be affected and can often complicate the performance of any image processing application [1].

Different sensors have different data acquisition rates, e.g. the GPS and accelerometer. Estimation algorithms can be used to deal with the different data acquisition rates by fusing data from faster sensors and slower sensors together [3]. Various different estimation algorithms exist. However, the estimation algorithm for this application needs to balance speed and accuracy.

Many different propellers exist. Each design has advantages and disadvantages that need to be considered. The appropriate propellers should to be used on the quadcopter to benefit the quadcopter for real-time image processing applications. The following research questions are to be answered in this proposed research:

1. Can a stability controller be designed to ensure a stable platform for accurate image acquisition?
2. Can accurate navigation be performed?
3. How can sensors with different data acquisition rates be combined for reliable stability control and navigation?

4. Is it possible to acquire reliable and not distorted images on this platform?
5. Is it possible to perform real-time image processing on this platform?

1.3 APPROACH

The proposed system consists of two main systems, namely: the controller system and the image processing system. The proposed system uses real-time image processing to demonstrate the quadcopter's controller capabilities. The controller will be developed on an embedded platform that will be mounted onto the quadcopter and the image processing software will be implemented on a high-level programming language, such as Python.

The controller can be divided into a stability and a navigation controller. The stability controller will have to be developed and tested intensely before the navigation controller is being developed. The stability controller will undergo intensive testing to ensure that the quadcopter is stable and predictable. The stability controller is an important part of the proposed system. If the stability controller is unstable, it will prove to be difficult to get reliable images for the image processing applications and it will be difficult to navigate to a new position. The navigation controller's main function is to estimate the current and new position of the quadcopter, as well as to navigate the quadcopter to the required position. The navigation controller must perform accurately to ensure that the quadcopter can navigate to the desired position and can perform the required image processing for the chosen application.

The image processing can be implemented once the controller has been implemented and tested intensively. The image processing system will use a pre-trained Machine Learning (ML) network. The pre-trained model will be imported and any extra image processing algorithms will be implemented (e.g., colour classification and distance estimation) in Python. Finally, once each system has been tested, relevant results will be obtained from the implemented system and will be compared to similar research studies to identify the performance of the proposed system.

1.4 RESEARCH GOALS AND CONTRIBUTION

The main goal of this research is to provide a stability controller and navigation algorithm to be used in quadcopters, being used for real-time image processing applications. The proposed system will provide a structured approach in designing and implementing a stability and navigation controller that utilises sensor fusion and estimation filters to achieve a quality flight controller for quadcopters with real-time image processing capabilities.

1.5 RESEARCH OUTPUTS

An article is being prepared for publication. V. Govender and H. C. Myburgh, “Navigation and control of a quadcopter for real-time image processing applications,” to be submitted to MDPI Sensors.

1.6 OVERVIEW OF STUDY

Chapter 2 provides a full literature study on the different quadcopter platforms, stability controllers, image processing techniques and estimation algorithms that are possible candidates in designing and implementing a navigation and stability controller for a quadcopter.

Chapter 3 provides reasoning for choosing hardware and software components to achieve a fully-functional quadcopter that can achieve the research goals.

Chapter 4 of the dissertation deals with the results regarding the stability and navigation algorithms obtained from the quadcopter being implemented. The results are thoroughly scrutinised in Chapter 4 of the dissertation to answer the proposed research questions and to discuss the performance of the developed system opposed to previous research papers. Image processing results for a person-following application is also presented in this chapter to prove that the proposed navigation and control system provides sufficient functionality for real-time image processing applications.

The final chapter of the dissertation provides a summary of the findings and also provides a brief discussion of further improvements to the system.

CHAPTER 2 LITERATURE STUDY

2.1 CHAPTER OVERVIEW

This chapter documents the research having been conducted to determine a suitable control and navigation system for a quadcopter that can be used in real-time image processing applications. The literature survey is divided into five main components. Section 2.2 discusses the various quadcopter configurations and the axes that will be used in further modelling of the quadcopter. In Section 2.3, popular controllers being used for stability, altitude and position are discussed, along with a list of possible sensors that can be used for a flight controller. In Section 2.4, estimation filters to improve the accuracy of sensor data are discussed.

In Section 2.5, the different types of cameras that can be used on a quadcopter are discussed. Finally, in Section 2.6, the different types of image processing techniques, such as ORB, Canny edge detection and various deep learning methods are discussed.

2.2 QUADCOPTER PLATFORM

This section describes the two main types of quadcopter configurations and the advantages and disadvantages of both configurations. The basic definition of movement is also clarified in this section and is used in further sections of the quadcopter modelling.

2.2.1 Quadcopter configuration

Before proceeding to the control of the quadcopter, it is important to define and model the quadcopter. The quadcopter can be defined as a + configuration or an x configuration [4]. The + configuration of a quadcopter can be seen in Figure 2.1(a) and the x configuration can be seen in Figure 2.1(b).

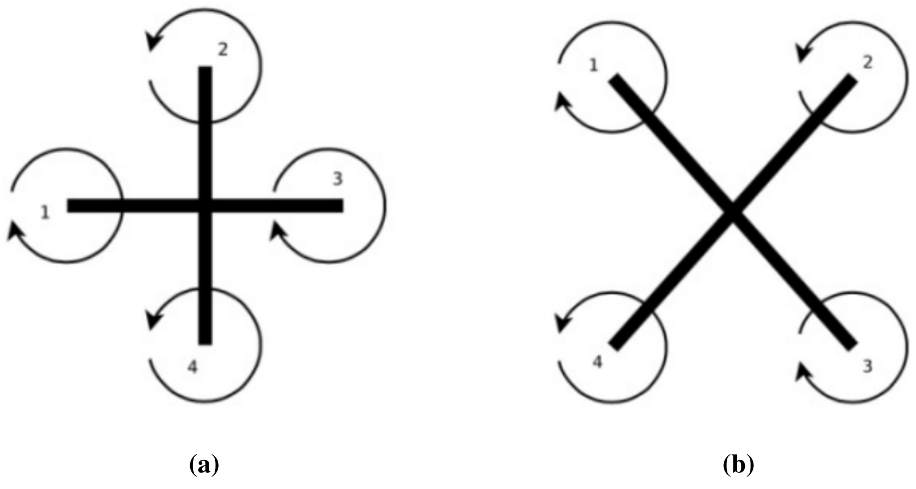


Figure 2.1. A top view of the + configuration and x configuration of a quadcopter are represented. (a) + configuration (Taken from [4], © 2016 IEEE.), (b) x configuration (Taken from [4], © 2016 IEEE.).

The main difference between the configurations is that the + configuration only has to use one motor to influence the roll, pitch or yaw motion of the quadcopter, whereas the x configuration uses two motors [4]. The x configuration can produce more thrust on either side of the quadcopter compared to the + configuration, since the x configuration has two motors on each side of the quadcopter [4]. The x configuration is the most popular configuration, as it provides better stability and also allows the quadcopter to be more acrobatic in the air [4]. The x configuration is also the preferred method, as it allows a camera to be easily mounted facing straight, as there are no propellers limiting the camera's line of vision [4]; however, this can be easily avoided in the + configuration if a landing gear is attached to the quadcopter and the camera is mounted to the bottom of the quadcopter. The direction in which the motors will be spinning should be noted. The opposing motors spin in the same direction and the adjacent motors spin in the opposite directions; this keeps the quadcopter in equilibrium, provided that all the motors are producing the same thrust [4].

The basic definition of the quadcopter's movements is briefly described to prevent confusion in the rest of the paper. Roll is a movement that causes the quadcopter to move to the left or to the right. Pitch is the movement that causes the quadcopter to move forward or backwards. Yaw is the movement that causes the quadcopter to rotate clockwise or anti-clockwise. The angular motions are illustrated in Figure 2.2 [5]. To increase or decrease the altitude, all the motors are required to produce the same thrust. To increase altitude, the motors need to produce a thrust greater than the gravitational force acting on the quadcopter, and to decrease the altitude the thrust is required to be less than the gravitational force acting on the quadcopter.

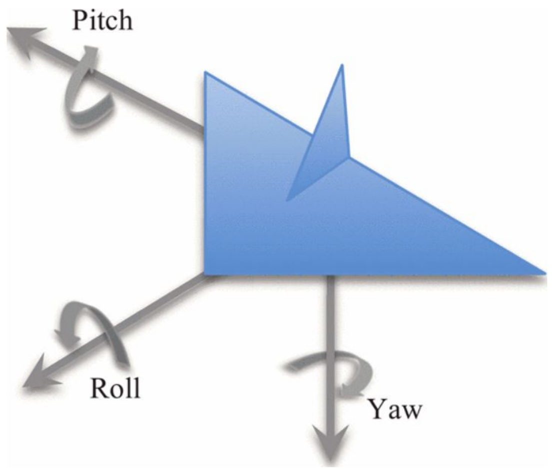


Figure 2.2. Roll, pitch and yaw of a quadcopter (Taken from [5], © 2014 IEEE.).

Typically, an Inertial Measurement Unit (IMU) sensor is used to measure the pitch and roll of a quadcopter. An IMU sensor consists of a gyroscope and an accelerometer. The gyroscope measures the rate of change in angular velocity and the accelerometer measures the acceleration forces acting on the quadcopter. The gyroscope is susceptible to drift [6] and the accelerometer is sensitive to vibrations [7]; therefore, data from the gyroscope and the accelerometer are often combined to improve results in pitch and roll.

2.3 FLIGHT CONTROLLER

This section deals with a detailed understanding of the stability controller, altitude controller and the position controller. Each controller is essential to ensure that the quadcopter can navigate in a 3D environment in a predictable and stable manner. Mathematical descriptions accompanied by reasoning are important to understand the working principles of each controller, as well as the contribution that each controller makes to the overall flight controller.

2.3.1 Stability controllers

A stability controller for the quadcopter is important to account for factors, such as manufacturing defects in motors, external forces (e.g. wind), propeller imbalances and the non-symmetrical weight distribution of a quadcopter. The stability controllers being evaluated, include the Sliding Mode (SM) controller [8, 9] and a Proportional Integral Derivative (PID) controller [10, 11, 12]. The SM and PID controllers are chosen to be evaluated as they are the most popular methods used for stability controllers for quadcopters.

2.3.1.1 SM controller

The Sliding Mode (SM) controller is a robust stability controller that accounted for non-linearities [9]. The SM controller consists of two components, namely the continuous and the discontinuous components [9]. The discontinuous component is represented by μ_D and the continuous component is represented by μ_C. The result of the continuous and discontinuous components are [9]

$$u(t) = \mu_C(t) + \mu_D(t). \tag{2.1}$$

The continuous component is represented by [9]

$$\mu_C(t) = \left(\tfrac{d}{dt} + \lambda\right)^n e(t)dt, \tag{2.2}$$

where λ represents the gain parameter, n represents the order and $e(t)$ represents the error between the reference value and the current value. When n = 1, the continuous component is reduced to,

$$\mu_C(t) = \dot{e} + \lambda e. \tag{2.3}$$

The discontinuous component is represented by [9]

$$\mu_D(t) = K_D sign(\mu_C(t)), \tag{2.4}$$

where K_D is the gain parameter for the discontinuous component and *sign* represents a non-linear function. Typically, a sigmoid function [9] is used as the non-linear function, therefore resulting in,

$$sign\left(\mu_C(t)\right) \cong \tfrac{\mu_C(t)}{|\mu_C(t)|+\delta}, \tag{2.5}$$

and the discontinuous term is expanded to [9]

$$\mu_D(t) = K_D \tfrac{\mu_C(t)}{|\mu_C(t)|+\delta}. \tag{2.6}$$

Finally, the thrust, roll, pitch and yaw are represented by $U1$, $U2$, $U3$ and $U4$, respectively [9],

$$U_1 = (\ddot{z}_d + \lambda(\dot{z}_d - \dot{z}) + g)\tfrac{m}{\cos\theta\cos\phi} + K_z \tfrac{s_z}{|s_z|+\delta}, \tag{2.7}$$

$$U_2 = \left[\ddot{\phi}_d - \dot{\psi}\dot{\theta}\left(\tfrac{I_y - I_z}{I_x}\right) - \dot{\psi}\dot{\theta} + \tfrac{J_r\dot{\theta}\Omega_r}{I_x} + \lambda\left(\dot{\phi}_d - \dot{\phi}\right)\right] I_x + K_{D\phi}\tfrac{s_\phi}{|s_\phi|+\delta}, \tag{2.8}$$

$$U_3 = \left[\ddot{\theta}_d - \dot{\psi}\dot{\phi}\left(\tfrac{I_z - I_x}{I_y}\right) + \dot{\psi}\dot{\phi} - \tfrac{J_r\dot{\phi}\Omega_r}{I_y} + \lambda\left(\dot{\theta}_d - \dot{\theta}\right)\right] I_y + K_{D\theta}\tfrac{s_\theta}{|s_\theta|+\delta}, \tag{2.9}$$

$$U_4 = \left[\ddot{\psi}_d - \dot{\phi}\dot{\theta}\left(\tfrac{I_x - I_y}{I_z}\right) - \dot{\phi}\dot{\theta} + \lambda\left(\dot{\psi}_d - \dot{\psi}\right)\right] I_z + K_{D\psi}\tfrac{s_\psi}{|s_\psi|+\delta}. \tag{2.10}$$

2.3.1.2 PID controller

The PID controller is the most popular controller used to stabilise an aerial vehicle. The PID controller consists of three components, namely the proportional component, the integral component and the derivative component [10, 12].

The proportional component calculates the error between the reference value and the current value, and scales the error according to the K_p term. The proportional component is applied to the system in the opposite direction of the error. The proportional component is calculated by [12]

$$u_t = K_p(y(t) - x_t), \tag{2.11}$$

where K_p is a proportional gain parameter, $y(t)$ is the reference value of the quadcopter at time t, and x_t is the current value of the quadcopter at time t.

By using only the proportional component, the system often experiences rapid changes; therefore, a derivative component is often added to smooth the response of the system [12]. The derivative component is responsible for reducing the rapid response from the system [12]. However, it should be noted that the derivative term will provide no significance in the case of a constant error, as the derivative component measures the rate of change in error [12]. This type of stability controller is known as a PD controller

$$u_t = K_p(y(t) - x_t) + K_D\frac{\partial(y(t) - x_t)}{\partial t}, \tag{2.12}$$

where K_D is a derivative gain parameter.

In some applications, the PD controller is sufficient as a stability controller. However, the PD controller cannot reduce the error to zero and, therefore, the integral component is added [12]. The integral component calculates the sum of errors over a period of time. The PID controller is calculated by [10, 12]

$$u_t = K_p(y(t) - x_t) + K_I\int(y(t) - x_t)dt + K_D\frac{\partial(y(t) - x_t)}{\partial t}, \tag{2.13}$$

where K_I is an integral gain parameter and the Laplace transform is [12]

$$U(s) = (K_p + K_D s + \frac{K_i}{s})(Y(s) - X(s)). \tag{2.14}$$

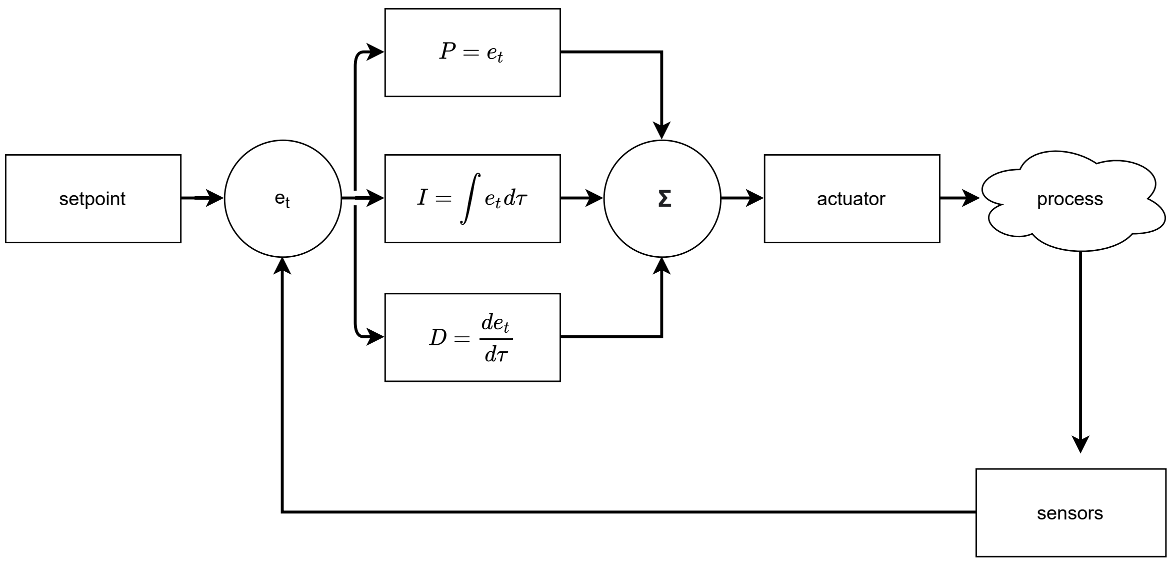


Figure 2.3. Block diagram for PID controller (Taken from [5], © 2014 IEEE.).

It can be noted that if the stability controller is not tuned correctly the quadcopter could react slowly to change and the image processing application cannot be fully realised. On the other hand, if the stability controller reacts in a abrupt manner the quadcopter will not be able to obtain quality images for further processing [2]. Therefore, it is of utmost importance that the stability controller is tuned correctly.

The SM controller [9] and the PID controller [10, 11] was compared by evaluating rise time, settling time and overshoot. It was found that both controllers produced satisfactory results. Due to popularity, the PID controller is slightly advantageous.

2.3.2 Altitude controller

Ultrasonic or laser sensors are often used to measure distance; however, in the case of the quadcopter, the sensors mentioned suffer a change in height if there is an obstacle on the ground, e.g. if the quadcopter flies over a box [13]. Due to the issue mentioned, an alternative altitude sensor that can be used, is a barometric sensor. By making use of the pressure measurements, it is possible to calculate the altitude of the quadcopter. By using this method, it is possible to keep the barometric sensor far from the ESCs to reduce electromagnetic interference and eliminate the issue of uneven ground. The altitude of the drone can be calculated by [11]

$$h = 44.33 \times 10^3 \left[1 - \left[\frac{P}{P_0} \right]^{\left(\frac{1}{5255 \times 10^{-3}}\right)} \right], \tag{2.15}$$

where h represents the altitude, P represents the current pressure and P_0 represents the ground pressure.

2.3.3 Position controller

The position controller is used to track the position of the quadcopter and to enable the quadcopter to navigate. An accelerometer can be used to estimate the distance travelled by the quadcopter in its respective directions; however, accelerometers are sensitive to vibration. To minimise the error, GPS data and the accelerometer data are combined to develop an accurate position estimation algorithm [3].

GPS modules use the National Marine Electronics Association (NMEA) Protocol. The NMEA protocol is a data protocol being used for communication transfer. The NMEA communication protocol was developed for marine electronics. Nowadays, the NMEA communication protocol is also used for other electronic devices, such as GPS modules. Table 2.1 shows the message IDs and the respective description for each message ID.

Table 2.1. NMEA message ID [14]

NMEA Packet message ID	
–RMC	Time, latitude, longitude, date, speed, magnetic variation.
–VTG	Course over ground, speed over ground
–GGA	Time, number of satellites, altitude, global latitude and longitude
–GSA	Provides details for each satellite (maximum of 12 satellites)
–GSV	Provides elevation angle, azimuth angle, signal to noise ratio
–GLL	Time, geographic latitude and longitude

2.3.4 Propellers

One of the important factors when designing a quadcopter is the type of propellers that will be used on the quadcopter. There exists a variate of choices regarding propellers. Each propeller is associated with a set of specifications, such as, diameter. pitch and number of blades.

Diameter is the full length of he propeller [15]. The pitch is the measurement of how far a propeller will travel in a single rotation [15]. An example of a propeller specification would be 10 * 4.5 inches, the 10 inches indicates the diameter and the 4.5 inches indicates the pitch of the blade [15]. The number of blades on a propeller is also indicated, such as, two-blade or three-blade propeller. In this dissertation only discusses how two-blade and three-blade propellers affect the quadcopter.

Each propeller there exists advantages and disadvantages. The two-blade propellers are efficient propellers and require smaller power motors compared to the three-blade propellers, as the three-bade propellers produce more thrust [15]. The two-blade propellers are more affected by the wind and opposed to the three-blade propellers [15]. The three-blade propellers are generally better for applications that favour stability as opposed to speed [15]. It can also be noted that the three-blade propeller could potentially mean an increase in cost, as the more powerful motors will be needed [15].

2.4 ESTIMATION FILTERS

The estimation filter can also be used to predict sensor values when data acquisition is too slow. In the case of the proposed project, the estimation filter will be used for position estimation. By making use of an estimation filter, the overall system responsiveness and accuracy will improve. The following estimation filters are the most widely-used filters.

A Kalman filter is a Bayes filter that is used for linear distributions. The Kalman filter assumes a linear Gaussian distribution [16] and is a popular state estimator. Variation of the Kalman filter exists to deal with non-linear distributions, such as the extended Kalman filter and the unscented Kalman filter [16].

The Extended Kalman filter (EKF) is a Bayes filter and an optimal estimator. The EKF shares most of the properties of the Kalman filter; however, it uses a first-order Taylor expansion to linearise a non-linear model [16].

The Unscented Kalman filter (UKF) is used for non-linear systems and is a variation of the Kalman filter. The UKF makes use of the unscented transform.

A Particle filter is a recursive Bayes filter. The broad idea of the Particle filter is to make use of several points, called particles, that are used to depict a distribution [16]. Each particle represents a possible state, while the state estimation can be improved with the increase of particles, there is a trade-off for computational time [16]. The Particle filter is useful, as it can work with unknown distributions. The motion model and the observation model remain the same in both the Kalman filter and the Particle filter [16].

2.5 VISION SENSOR

Cameras are used as vision sensors. The most common methods to use a camera is monocular [17, 18, 19, 20, 21] and stereo vision [22, 23, 24, 21]. The main difference between the monocular and stereo vision approach is that monocular vision uses a single camera, while stereo vision uses two cameras. The cameras used for either approach are often low pixel because high pixel cameras produce a large amount of data to process; thereby increasing the computation time of the image processing. In the stereo vision approach two cameras are placed at different angles, viewing the same scene [25]. Stereo vision allows for depth information by computing a disparity map, using both images from the cameras [25]. Stereo vision is often used to detect features in a scene, as well as to estimate distance between the cameras and the objects in a scene.

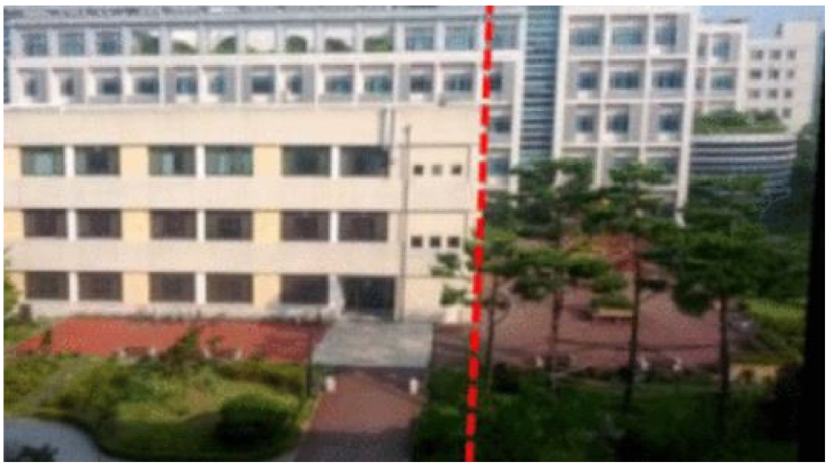
Two types of cameras are often considered for drone use:

- Charge-Coupled Device (CCD) cameras,
- Complementary Metal–Oxide–Semiconductor (CMOS) cameras.

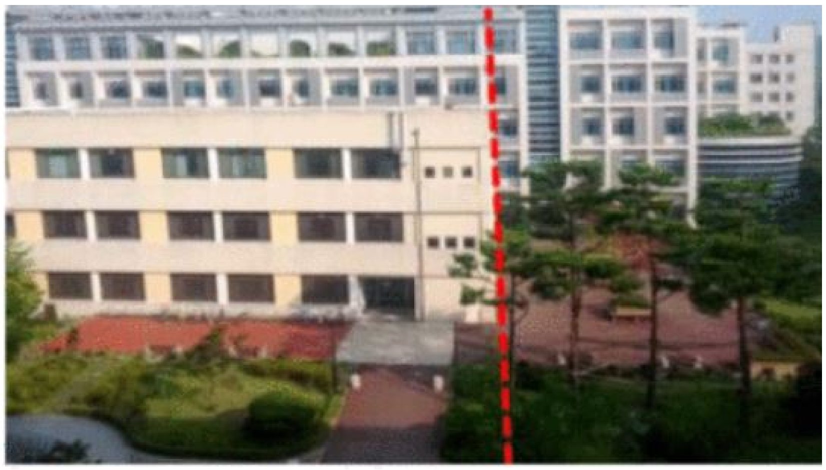
CCD cameras use an image CCD image sensor [26]. The image sensor consists of photoelectrons that convert the intensity of light into electric charge. The downside of CCD cameras is that the electric charge is collected serially; therefore, image acquisition is limited [26]. The CMOS camera has a similar method for image acquisition. However unlike the CCD camera, the CMOS collects the charge in a parallel manner, which ultimately improves the image acquisition time [26].

2.5.1 Jello effect

Jello effect is the phenomenon that occurs when a camera experiences high frequency vibration [27]. This phenomenon causes the camera's image to be distorted. The jello effect is caused by fast movements in the air and high frequency vibration that are transmitted from the quadcopter's frame to the camera [27]. Figure 2.4(a) represents the original image and Figure 2.4(b) represents the distorted image. The red dotted line clearly indicates the distortion between the two images.



(a)



(b)

Figure 2.4. Comparison between original image and distorted image. (a) Original image (Taken from [28], © 2015 IEEE.), (b) Distorted image (Taken from [28], © 2015 IEEE.).

To mitigate the effects of jello, smoother quadcopter movements are required, as well as a vibration damping system to isolate high frequency vibration between the quadcopter frame and camera.

The Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR) are popular methods to measure image quality [29, 30]. MSE calculates the cumulative square intensity difference between the reference image and the image to be evaluated [29, 30]. PSNR uses the MSE to measure the peak error [29, 30]. Another popular method is the Structural SIMilarity (SSIM) that measures the similarity in an image by taking into account lamination, contrast and similarity [29, 30].

2.6 TARGET DETECTION

Image processing is often used for target detection. There are two main steps to develop a target detection system: Pre-processing and target detection.

In the pre-processing phase, the image undergoes various processing to reduce noise in the image and to remove unwanted features in the image [31]. Pre-processing is particularly important because it intensifies key features in an image and improves the accuracy of the image processing [31]. Pre-processing techniques include dilution, erosion, maximum and minimum filter, smoothing filter etcetera. [31]. In the target detection phase, the target of interest is extracted from the pre-processed image. The target detection involves algorithms, such as template matching [22], Neural Networks (NN) [32, 33, 12], Histogram of Gradients (HOG) [24].

In work related to pre-processing, the dilation and erosion techniques have been used. Dilation and erosion are known as a morphological technique that can be used to influence the size of an image [31, 21]. The main difference between the erosion and dilation technique is that the erosion technique reduces the size of a target by eliminating pixels around the target's edges and the dilation technique increases the size of a target by increasing the number of pixels around the target's edges [31, 21].

In earlier work related to target detection, background extraction methods were used to obtain the foreground target in an image [21, 34]. By using a background extraction method, a model of the background is required [34]. Kernel Density Estimation (KDE) is also used for target detection; however, due to its poor performance with outliers, a Robust KDE was developed and proved to successfully deal with outlier [23]. Using colour alone for target detection often fails, due to illumination issues [35]. Recent methods have used colour and depth estimation to account for illumination [35]. However, the issue with depth-sensing becomes clear when multiple targets with the same depth are visible in a frame [35]. Feature detection algorithms such as Scale-Invariant Feature Transform (SIFT) [22, 19, 36], Speeded-Up Robust Features (SURF) [19] and Oriented FAST and Rotated BRIEF (ORB) [22], are commonly used for target detection. In recent studies, ORB has been found to perform better than SIFT and SURF [22, 19]. Other target detection algorithms used foreground blob detection and Histogram of Gradients (HOG). It has been found that the HOG produces higher accuracy than foreground blob detection. However, foreground blob detection is more applicable for real-time operation [24]. Canny edge detection can also be combined with colour detection, in order to improve target detection [20].

A Convolution Neural Network (CNN) is also a viable option for target detection [19]; however, an issue with the CNN is that it requires a large number of training inputs to provide accurate results [19].

A Siamese Network is often used, as opposed to the CNN, because the Siamese Network requires a small number of samples to train the network; therefore being an ideal option if there is a lack of training data [33]. The Faster Region CNN (R-CNN) is a variation of a CNN and is one of the leading deep learning methods for target detection, as it provides reasonable accuracy for real-time target detection [37, 38]. The You Only Look Once (YOLO) method is an alternative deep learning method that can perform in real-time and provide adequate accuracy [39, 40].

A few of the popular target detection techniques are further discussed.

2.6.1 ORB

The Oriented FAST and Rotated BRIEF (ORB) algorithm is a common algorithm being used for real-time feature detection that is a combination of a Features from Accelerate Segment Test (FAST) algorithm and the Binary Robust Independent Elementary Features (BRIEF) algorithm [41].

BRIEF is a binary descriptor that compares the intensities of the pixels of two images [41]. FAST is an algorithm that is used for corner detection [41]. Figure 2.5 is used to assist in the explanation of the FAST algorithm. A pixel is regarded to be a corner pixel if the intensity of the pixel is greater or less than other pixels surrounding the pixel of interest [41]. In the case of pixel p, the intensity of sixteen other pixels is compared to pixel p [41]. The number of pixels evaluated around the pixel of interest can be set, thus allowing the FAST algorithm to be computationally efficient and suitable for real-time applications [41]. An issue with the FAST algorithm is that it does not provide the orientation. However, a rotational matrix can be used to get the orientation before the BRIEF algorithm is applied [41].

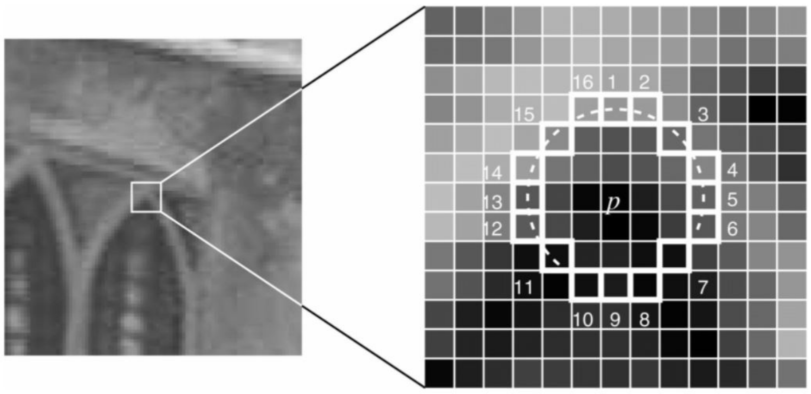


Figure 2.5. Feature detection with FAST (Taken from [42], © 2008 IEEE.).

SIFT and SURF are other techniques being used for corner detection; however, these algorithms do not perform well in real-time applications [22].

2.6.2 Canny edge detection

Canny edge detection is an algorithm that is applied to a grayscale image to emphasise edges in a picture. Canny edge detection consists of five stages. For the first step, a Gaussian blur is applied to the image to reduce noise [43].

The second step is to calculate the gradient [43]. By calculating the gradient, the intensity and the direction of the edge can be found [43]. The gradient for each pixel is calculated, using a Sobel operator [43]. To find the gradient in the horizontal plane, the kernel function (K_{GX}) is applied to the horizontal and vertical plane [43]. The magnitude and direction of the gradient are then calculated.

In the third step, the local maximum gradient pixels are kept, while the rest are eliminated. This is done by evaluating each pixel, as well as neighbouring pixels around each pixel, and checking if the current pixel has the highest gradient amongst the neighbouring pixels [43]. When the current pixel has the highest gradient amongst the neighbouring pixels, then the pixel is kept. When the current pixel does not have the highest gradient amongst the neighbouring pixels, then the pixel is set to zero [43].

The fourth step is to determine which pixels are relevant and which pixels do not provide any relevance. This is done by using a high threshold and a low threshold [43]. Pixels that have a higher value than the high threshold are placed in a strong group, while pixels that have a lower value than the low threshold, are eliminated and pixels that are between the low threshold and the high threshold are placed in a weak group [43].

The final step is constructing the output for the Canny edge detection algorithm. All strong pixels are kept, while weak pixels are only kept if they are connected to strong pixels [43].

2.6.3 Deep learning methods

A number of deep learning methods exists; however, only a few of the popular deep learning methods for the real-time image processing application will be discussed further.

2.6.3.1 Artificial neural network

An Artificial Neural Network (ANN) is a collection of units that are linked together [12]. An ANN mimics the neurons in the brain. The units in an ANN are often referred to as a neurons [12]. Figure 2.6 shows a neuron in an ANN [12, 44].

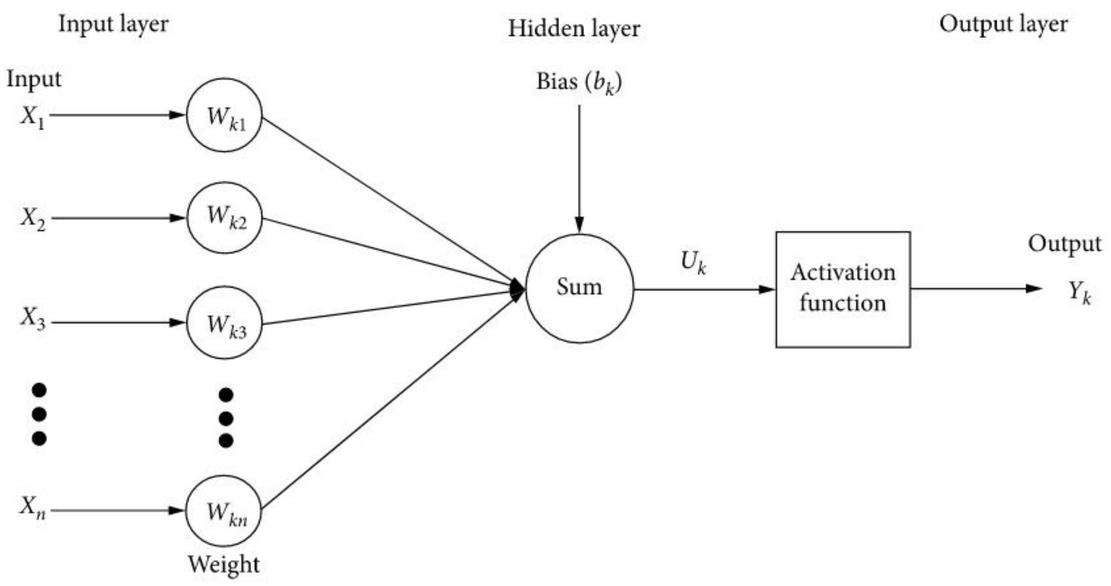


Figure 2.6. Model of a neuron (Taken from [44], with permission.).

At the neuron, multiple inputs (X_n) are scaled, using a corresponding weight of W_{kn} and added together. The sum of the scaled inputs are calculated as [44]

$$U_k = \sum_{j=0}^{n} W_{kj} X_j + b_k. \tag{2.16}$$

Once the sum of the scaled inputs has been computed (U_k), the sum of the scaled inputs is passed through an activation function. An activation function is used to scale U_k between the limits of the activation function being used [12, 44]. Depending on the activation function being used, it can introduce non-linearities to the ANN [12, 44]. The most common activation functions include the sigmoid function, hyperbolic tangent function, unit function ,etcetera. [12]. The output from the neuron is given as [44]

$$Y_k = g(U_k), \tag{2.17}$$

where g represents the activation function.

In Figure 2.7 it should be noted that the layers between the input layer and the output layer are known as the hidden layers [12]. Figure 2.7 shows a feed-forward ANN and is referred to as a feed-forward ANN because the arrows are all pointed in one direction [12, 45].

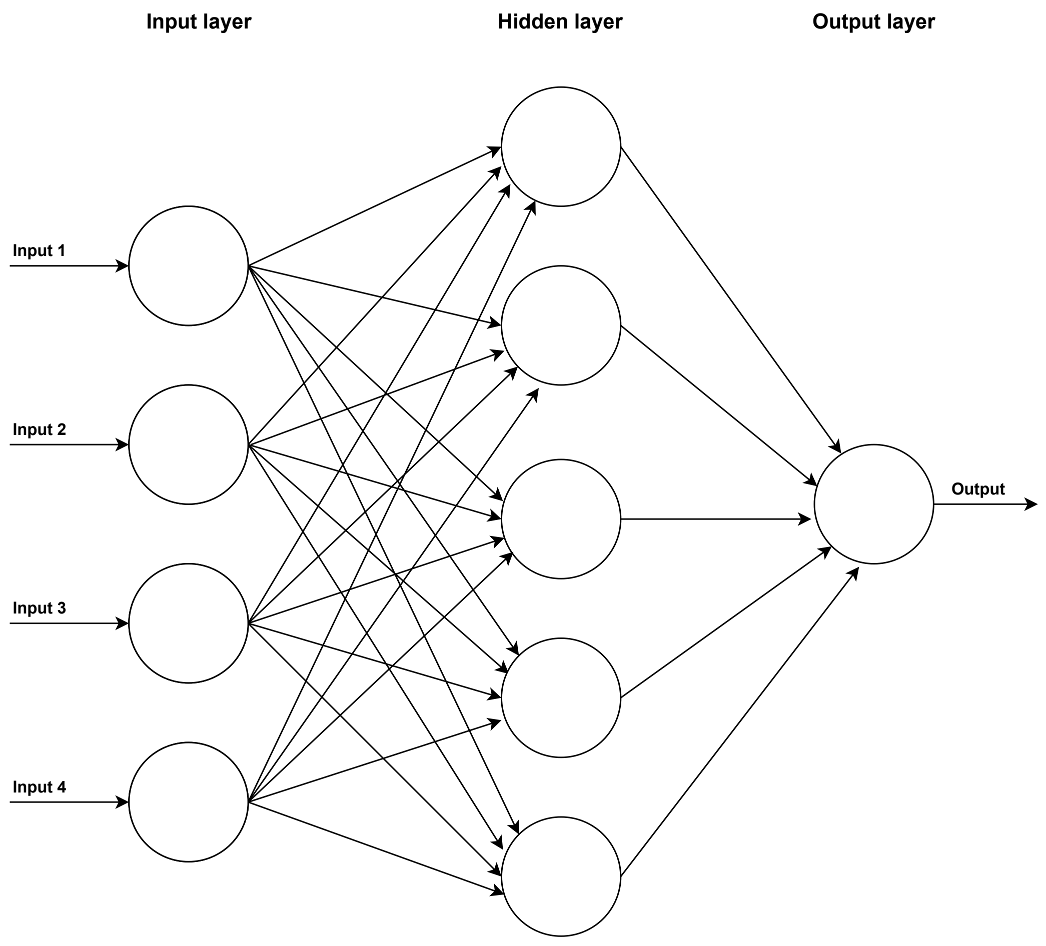


Figure 2.7. A neural network (From [45], © 2017 IEEE.).

An ANN that has arrows pointing forward and backwards, exists. The backwards operation is known as back-propagation. Back-propagation is the process of training the ANN to update the weights (W_{kj}) being used to scale the inputs [12]. To train a network, data consisting of known outputs are used, which is known as the training set. The back-propagation process can be mathematically described as [12]

$$W_{kj} = W_{kj} + \alpha a_j \Delta_k, \tag{2.18}$$

where α is the learning rate and Δ is the error between the predicted output and the actual output.

The ANN indeed performs reasonably well; however, it requires a large amount of training data to provide accurate results [12]. The ANN also suffers from overfitting if the training data does not contain a fair representation of all data [12].

A Siamese network deals with the problem of requiring large training data sets. A Siamese network differs from a traditional ANN, by training the Siamese network with a small amount of available data, and instead of trying to classify the input data, the Siamese network calculates the similarity between the input data and the training data [33].

2.6.3.2 Convolutional neural networks

Another type of deep learning method being used for target detection is a Convolutional neural networks (CNN). The CNN architecture consists of three types of layers: convolutional layer, pooling layer and fully-connected layer.

The convolutional layers are filters that are convolved with the input to the convolutional layer. A convolutional layer is then followed by an activation function, e.g. ReLU, sigmoid etcetera. The output being generated from the convolutional layer is known as a feature map [46].

The pooling layer is used to reduce the size of the convolutional layers, in order to reduce the computational time. The two methods used for pooling are max pooling and average pooling. The pooling layer segments the input, based on stride size and filter size. The max or average value in each segment is then retained and placed in an output matrix of the pooling layer [46]. Max pooling is the most common method for pooling.

The last type of layer is the fully-connected layer. The fully-connected layer is a neural network where every neuron is connected to the features of the preceding layer. The fully-connected layer is then passed through to a function, such as a softmax function, as the final output from the CNN [46].

Initially, a sliding window was used for target detection. To use the sliding window method, an nxn window size was predefined, indicating the number of pixels that was passed through the CNN at a time. The window would then slide across the input image, based on a stride size. The problem with the CNN and the sliding window method is that it was computationally expensive; therefore it could not be used for real-time target detection. Another problem with the sliding window was that it was difficult to choose the correct size of the window.

2.6.3.3 Faster region convolutional neural network

The R-CNN is a method used to reduce the computation time. The main issue with the CNN and the sliding window approach was that some of the windows being processed by the CNN were not necessary, as the windows did not contain anything useful to the relevant application. For example, if the application was to detect cars in an image, then some of the sliding windows would contain portions of the sky. The architecture for a Faster R-CNN is shown in Figure 2.8.

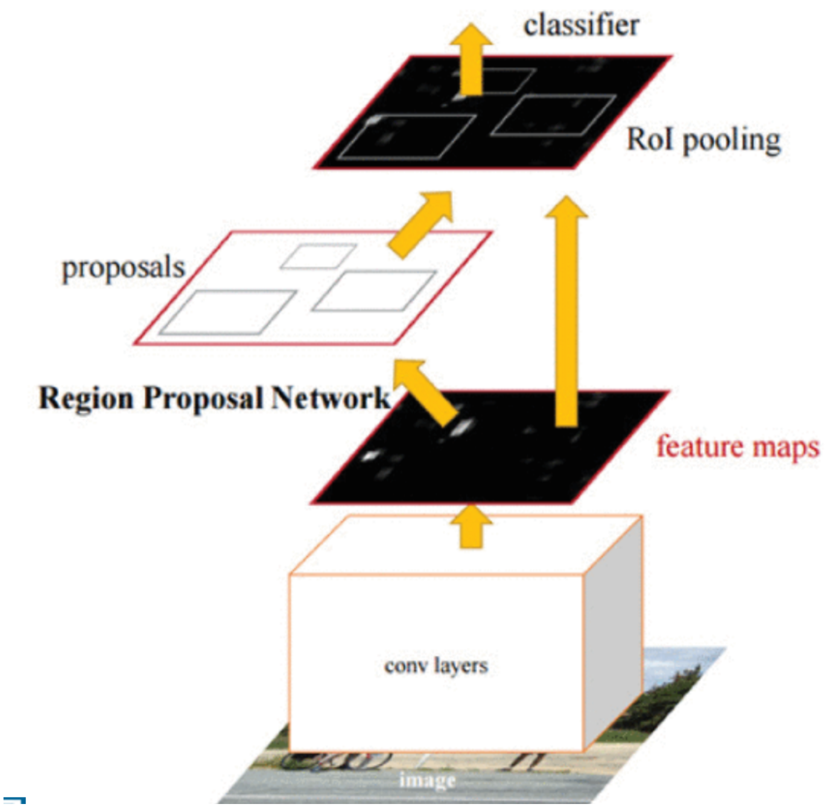


Figure 2.8. R-CNN architecture. (Taken from [38], © 2018 IEEE.).

The Faster R-CNN consists of the CNN and the Region Proposal Network (RPN). The CNN remains the same as disussed in the CNN section; however, instead of using the sliding window method, the Faster R-CNN has a seperate network (RPN) to propose regions in an image that contain mostly relevant portions of an image. The proposed regions and the feature map from the convolutional layers are combined in the RoI pooling and are then passed through the classifier for target detection

[37, 38, 47, 48].

2.6.3.4 You Only Look Once (YOLO)

The YOLO algorithm is currently one of the state of the art real-time target detection algorithms. The YOLO algorithm divides the input image into an SxS grid cell, as shown in Figure 2.9. The SxS grid is also the output from the CNN being used in the YOLO algorithm [39, 40, 49, 50].

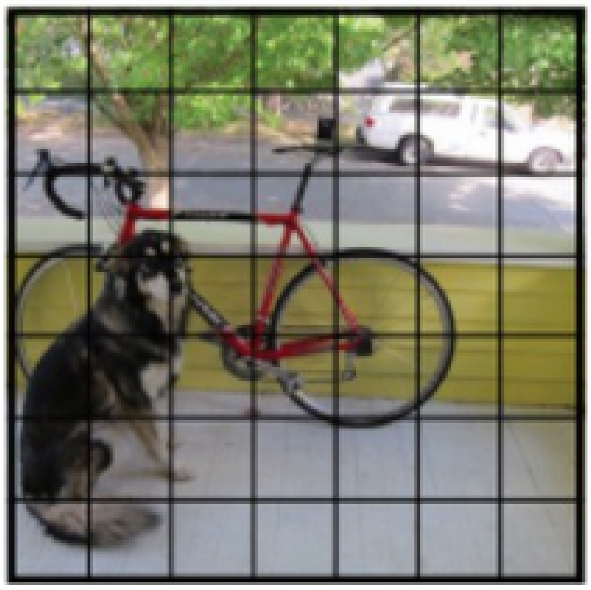


Figure 2.9. Image divided into an SxS grid (Taken from [51], © 2016 IEEE.).

The YOLO algorithm uses a CNN to predict the bounding box around the target, the confidence level that there is a target in a given grid cell, as well as what type of target is found in a given grid cell. Figure 2.10 shows the bounding boxes found by the CNN [39, 40, 49, 50].

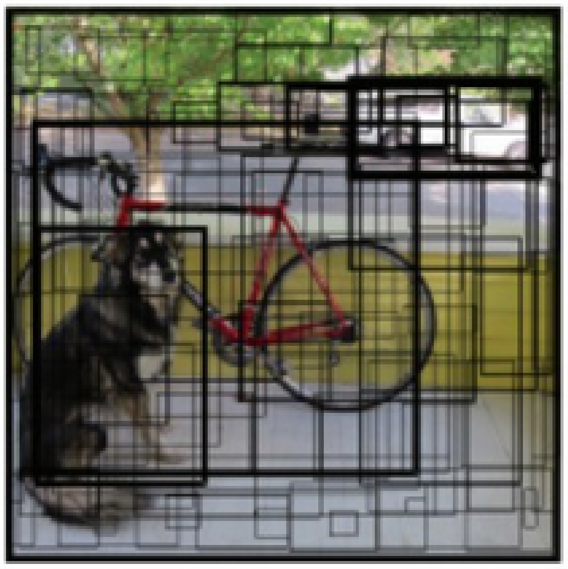


Figure 2.10. Result from the CNN (Taken from [51], © 2016 IEEE.).

It can be noted from Figure 2.10 that there is a lot of bounding boxes drawn on the image. To remove some of the overlapping bounding boxes and unnecessary bounding boxes, a non-suppression method is used. The non-suppression method compares each overlapping bounding box and calculates the Intersection of Union (IoU), which is calculated as

$$IoU = \frac{I}{A}, \tag{2.19}$$

where I represents the area of intersection of the two bounding boxes and A represents the sum of the areas of each bounding box.

The non-suppression algorithm first discards all bounding boxes with a confidence level lower than a predefined threshold (p_c). The non-suppression algorithm then picks the bounding boxes with the highest confidence levels and then discards any boxes with a IoU value of less than the predefined threshold (c) [39, 40, 49, 50]. The result after the non-suppresion algorithm has been applied is shown in Figure 2.11.

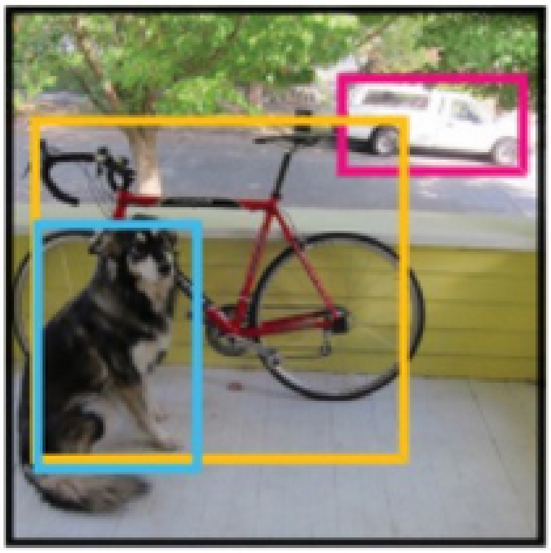


Figure 2.11. CNN output after undergoing non-suppression (Taken from [51], © 2016 IEEE.).

The YOLO algorithm can also be used to detect people in an image to meet the requirements of the proposed project. To detect the colour of the person's shirt and pants, a colour classification algorithm is required. The K-Means and the MeanShift algorithm produced successful results for extracting colour from an image [36]. MeanShift is advantageous compared to the K-Means clustering algorithm because it does not require the number of clusters upon start-up, however, MeanShift is computationally expensive compared to K-Means clustering [52].

2.7 CHAPTER SUMMARY

This chapter discussed different quadcopter configurations along with each configuration's advantages and disadvantages. Furthermore, the flight controller was discussed extensively, as well as estimation filters that can be used to improve stability and accuracy of the position, altitude and stability controllers. The vision sensors were discussed and key factors, such as the jello effect were also discussed. Finally, the possible target detection and colour classification algorithms for real-time image processing were discussed. The algorithms that were discussed in this chapter are not the only algorithms that exist that can solve the various challenges for this project; however, they are the most viable algorithms that will achieve the desired goals.

CHAPTER 3 METHODS

3.1 CHAPTER OVERVIEW

This chapter deals with the design and implementation of the quadcopter platform and image processing system. This chapter is divided in six main sections. The platform and mass calculations of the quadcopter is presented in Section 3.2. In Section 3.3, the design and implementation of the subsystems for the flight controller, such as the stability controller, altitude controller and position controller are presented. Furthermore, the magnetometer calculations and the battery compensation algorithm are presented in Section 3.3. In Section 3.4 the Bayes filter and the Kalman Filter are presented that will be used as the estimation filter to improve sensor data.

Section 3.5 presents the vision sensor chosen for the image processing application. The real-time image processing technique being used, are described in detail in Section 3.6. Finally, in Section 3.7, the various communications between the quadcopter and base station are described in detail.

3.2 QUADCOPTER PLATFORM OVERVIEW

The method used to achieve the proposed system is divided into the stability controller for the quadcopter and the target detection algorithm. Figure 3.1 shows the high-level representation of the system.

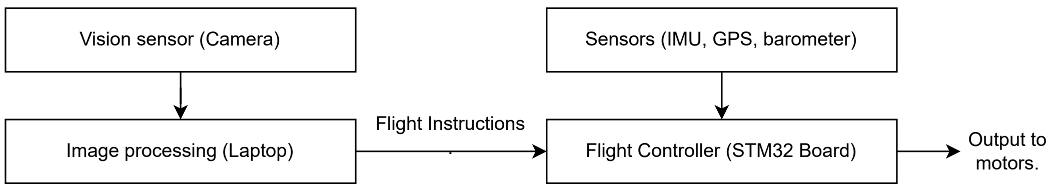

Vision sensor (Camera)
Sensors (IMU, GPS, barometer)
Image processing (Laptop)
Flight Instructions
Flight Controller (STM32 Board)
Output to motors.

Figure 3.1. High-level representation of the proposed system.

A base station (laptop) was responsible for handling the image processing required for the proposed system. The base station received wireless input from the cameras to perform target detection. Based on the image processing, flight instructions were sent to the flight controller. The flight controller was responsible for controlling the motors and stabilising the quadcopter. The flight controller received inputs from the sensors (IMU, GPS and barometer), as well as the flight instructions received by the image processing. A list of components on the quadcopter and masses are recorded in Table 3.1.

Table 3.1. List of components used on the quadcopter.

Components	Mass(g)
Component	**Mass (g)**
DJI quadcopter frame	282
DJI 920KV motor	64 x 4
DJI ESC	15 x 4
Power distribution board	10
2200mAh LiPo battery	173
STM32F10	10
GPS (U-blox M8N + HMC588L)	45
IMU (MPU6050)	15
Barometer (MS5611)	1.2
RC receiver	32
8 x 4.5" propellers	9 x 4
Caddx Ratel FPV camera	8
Eachine transmitter (TS5828L)	30
3D prints	16.4
Estimated total mass	1067.5 + miscellaneous
Actual total mass	1150

For the quadcopter to perform the desired motion, the quadcopter configuration must be defined and each motor should be labelled. Figure 3.2 represents the quadcopter in an x configuration and the motors are labelled from 1 to 4.

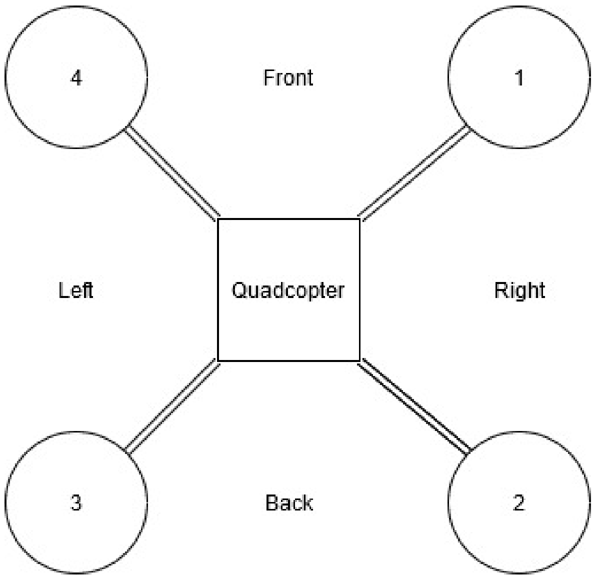
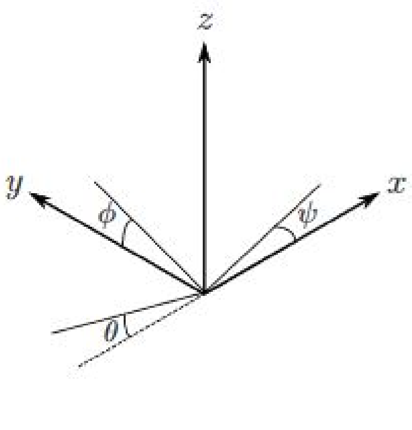


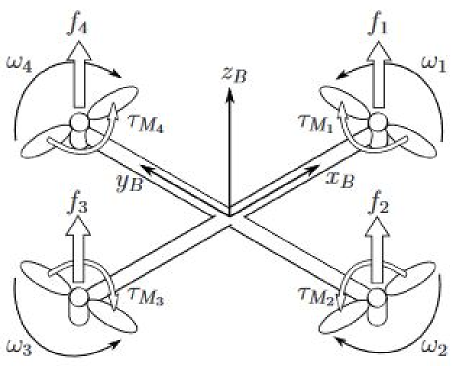
Figure 3.2. Quadcopter x configuration.

3.2.1 Quadcopter modelling

The fixed frame of the quadcopter is shown in Figure 3.3(a). It is important to establish a point of reference to measure the linear translation and the rotation of the quadcopter [10, 53, 54]. The force diagram of the quadcopter is shown in Figure 3.3(b). The force diagram for the quadcopter represents the forces and the moments, acting on the quadcopter.



(a)



(b)

Figure 3.3. Fixed frame and the force diagram for the quadcopter. (a) Fixed frame (Taken from [55], © 2018 IEEE.), (b) Force diagram (From [55], © 2018 IEEE.).

The fixed frame is described, using the Cartesian representation. The fixed frame is represented by the symbol $\boldsymbol{\xi}$ [10, 53]

$$\boldsymbol{\xi} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{3.1}$$

The rotation of the quadcopter is measured in roll, pitch and yaw. The roll, pitch and yaw are represented by ϕ, θ and ψ, respectively. The vector $\boldsymbol{\eta}$ represents the rotation of the quadcopter [10, 53, 54]

$$\boldsymbol{\eta} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}. \tag{3.2}$$

The linear velocities in the x, y and z direction are represented by $v_{x,Q}$, $v_{y,Q}$ and $v_{z,Q}$, respectively [10, 53]. The vector $\boldsymbol{V}_Q$ represents the linear velocities

$$\boldsymbol{V}_Q = \begin{bmatrix} v_{x,Q} \\ v_{y,Q} \\ v_{z,Q} \end{bmatrix}. \tag{3.3}$$

The angular velocities ($\boldsymbol{v}$) for the quadcopter are represented by p, q and r [10, 53, 54]

$$\boldsymbol{v} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{3.4}$$

For simplicity, the *cos*, *sin* and *tan* functions are represented by C, S and T symbols, respectively, and the angle is shown in subscript [10, 53, 54]. The rotational matrix being used to describe the motion of the quadcopter, is represented by $\boldsymbol{R}$,

$$\boldsymbol{R} = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}. \tag{3.5}$$

The quadcopter is assumed to be symmetrical; therefore, the diagonal components of the moment of inertia are only present, resulting in the rest of the components equating to zero [10, 53]. The moment of inertia is calculated as,

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}. \tag{3.6}$$

The force produced by each motor is represented by f_i [10, 53],

$$f_i = k\omega_i^2, \tag{3.7}$$

where k represents the lift constant and ω represents the angular velocity of the motor. The created torque from a signle motor is represented by τ [10, 53],

$$\tau_{Mi} = b\omega_i^2 + I_M\dot{\omega}_i, \tag{3.8}$$

where b represents the drag constant and I_M represents the inertia moment being produced by the motor.

The thrust force produced by all the motors, needs to overcome the force of gravitational force that acts on the quadcopter [53]. The vector $\boldsymbol{T}_Q$ represents the thrust force in the Cartesian representation,

$$\boldsymbol{T}_Q = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^4 k\omega_i^2 \end{bmatrix}. \tag{3.9}$$

The torque is represented by τ, using the fixed frame representation [53, 54],

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_\phi \\ \tau_\omega \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk(-\omega_2^2 + \omega_4^2) \\ lk(-\omega_1^2 + \omega_3^2) \\ \sum_{i=1}^4 \tau_{Mi} \end{bmatrix}, \tag{3.10}$$

where l represents the length between the motor and the centre of mass of the quadcopter.

The Newton-Euler equations were used to describe the dynamics of the quadcopter, therefore, to use the Newton-Euler equations, the quadcopter is assumed to be a rigid-body [10, 53, 54]. The quadcopter's forces include the force required to accelerate the quadcopter and the centrifugal force acting on the quadcopter, which are required to equal the gravitational force and the total thrust force [53, 54]. The Newton-Euler general equation is,

$$m\dot{\boldsymbol{V}}_B + \boldsymbol{v} \times (m\boldsymbol{V}_Q) = \boldsymbol{R}^T\boldsymbol{G} + \boldsymbol{T}_Q. \tag{3.11}$$

In the fixed frame, a centrifugal force does not exist, therefore, the equation is reduced to [53, 54],

$$m\ddot{\boldsymbol{\xi}} = \boldsymbol{G} + \boldsymbol{R}\boldsymbol{T}_Q, \tag{3.12}$$

and in its expanded form, it is [53],

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \frac{\sum_{i=1}^4 k\omega_i^2}{m} \begin{bmatrix} C_\psi S\theta C_\phi + S_\psi S_\phi \\ S_\psi S\theta C_\phi - C_\psi S_\phi \\ C_\theta C_\phi \end{bmatrix}. \tag{3.13}$$

In the force diagram, the centrigual force is present. The angular acceleration of the quadcopter and the gyroscopic forces are also present [53, 54]. Therefore, the Newton-Euler equation becomes,

$$I\dot{\boldsymbol{v}} + \boldsymbol{v} \times (I\boldsymbol{v}) + \boldsymbol{\Gamma} = \boldsymbol{\tau}, \tag{3.14}$$

and in its expanded form, it is [53],

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = I^{-1}\left(-\begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_{xx}p \\ I_{yy}q \\ I_{zz}r \end{bmatrix} - I_r \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega_r + \tau \right). \tag{3.15}$$

3.3 ESTIMATION FILTER

In the case of the proposed project, the estimation filter will be used for position estimation. By using an estimation filter, the overall system responsiveness and accuracy will improve.

3.3.1 Bayes filter

The Bayes filter is one of the main building blocks for developing an estimation filter. It is beneficial to understand the probability theory of the Bayes filter before moving on to estimation filters, such as the Kalman filter and the Particle filter. The belief state is defined as [16]

$$bel(\boldsymbol{x}_t) = p(\boldsymbol{x}_t | \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}), \tag{3.16}$$

where $\boldsymbol{x}$ is the current position, $\boldsymbol{z}$ is the observation and $\boldsymbol{u}$ the motion control. The Bayes rule is applied to the belief state [16]

$$bel(\boldsymbol{x}_t) = np(\boldsymbol{z}_t | \boldsymbol{x}_t, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) \times p(\boldsymbol{x}_t | \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}). \tag{3.17}$$

By using the Markov assumption, the belief state becomes [16]

$$bel(\boldsymbol{x}_t) = np(\boldsymbol{z}_t | \boldsymbol{x}_t) \times p(\boldsymbol{x}_t | \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}). \tag{3.18}$$

The law of total probability is then applied to transform the belief state to [16]

$$bel(\boldsymbol{x}_t) = np(\boldsymbol{z}_t | \boldsymbol{x}_t) \times \int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) p(\boldsymbol{x}_{t-1} | \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) d\boldsymbol{x}_{t-1}. \tag{3.19}$$

The Markov assumption is once again applied to transform the belief state into [16]

$$bel(\boldsymbol{x}_t) = np(\boldsymbol{z}_t | \boldsymbol{x}_t) \times \int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_t) p(\boldsymbol{x}_{t-1} | \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) d\boldsymbol{x}_{t-1}. \tag{3.20}$$

The Bayes filter can be broken into two steps, namely: prediction step and correction step. The prediction step can be written as [16]

$$\bar{bel}(\boldsymbol{x}_t) = \int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{u}_t)p(\boldsymbol{x}_{t-1}|z_{1:t-1},\boldsymbol{u}_{1:t-1})d\boldsymbol{x}_{t-1}, \tag{3.21}$$

and the correction step is written as [16]

$$bel(\boldsymbol{x}_t) = np(z_t|\boldsymbol{x}_t)\bar{bel}(\boldsymbol{x}_t). \tag{3.22}$$

Two models are used to describe the position estimation, namely the motion model and the observation model. The motion model describes the belief state with regards to the previous state and the external inputs (e.g. RC receiver). The motion model can be mathematically described by [16]

$$P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{u}_t). \tag{3.23}$$

The observation model uses measurements (e.g. IMU, GPS, etcetera.) to update the belief state. The observation model can be mathematically described by [16]

$$P(z_t|\boldsymbol{x}_t). \tag{3.24}$$

3.3.2 Kalman filter

A Kalman filter is a Bayes filter that is used for linear distributions. The Kalman filter assumes a linear Gaussian distribution [16] and is a popular state estimator. The motion model ($\mathbf{X}_{t+1}$) for the Kalman filter can be calculated by [16]

$$\mathbf{X}_{t+1} = \mathbf{A}_t\mathbf{X}_t + \mathbf{B}_t\mathbf{u}_t + \boldsymbol{\varepsilon}_t, \tag{3.25}$$

where $\mathbf{A}_t$ is a square matrix that represents a state with the absence of external inputs, $\mathbf{B}_t$ is a square matrix that represents a state due to external inputs and $\boldsymbol{\varepsilon}_t$ represents additional noise. The observation model ($\mathbf{Z}_t$) is calculated by [16]

$$\mathbf{Z}_t = \mathbf{C}_t\mathbf{X}_t + \boldsymbol{\delta}_t, \tag{3.26}$$

where $\mathbf{C}_t$ is a projection matrix and $\boldsymbol{\delta}_t$ represents the input measurement noise from sensors or image processing. Observation noise is represented by $\mathbf{R}_t$. The covariance matrix ($\mathbf{P}_t$) is calculated by [16]

$$\mathbf{P}_t = \mathbf{A}_t\mathbf{P}_t\mathbf{A}_t^T + \mathbf{R}_t. \tag{3.27}$$

The next step is to calculate the Kalman gain. The Kalman gain indicates if the measured value or the predicted measurement represents the actual value more accurately [16]. The Kalman gain is a value between zero and one. A Kalman gain of one indicates that the measured value depicts the actual value more accurately [16]. The Kalman gain ($\mathbf{K}_t$) can be calculated by [16]

$$\mathbf{K}_t = \mathbf{P}_t \mathbf{C}_t^T [\mathbf{C}_t \mathbf{P}_t C_t^T + \mathbf{Q}_t]^{-1}. \tag{3.28}$$

The Kalman gain is then used to update the estimated state by [16]

$$\mathbf{X}_{t+1} = \mathbf{X}_t + \mathbf{K}_t(\mathbf{Z}_t - \mathbf{C}_t \mathbf{X}_t). \tag{3.29}$$

The covariance matrix ($\mathbf{P}_t$) is also updated, using the Kalman gain [16]

$$\mathbf{P}_{t+1} = (\mathbf{I}\ \text{-}\ \mathbf{K}_t \mathbf{C}_t)\mathbf{P}_t. \tag{3.30}$$

In the case of the quadcopter, the basic equation of motion is used in one-dimension is

$$x_t = x_{t-1} + v_t + \frac{1}{2} a_t \Delta T^2 \tag{3.31}$$

where, x_t represents the new position, x_{t-1} represents the previous position, v_t represent the measured velocity and a_t represents the measured acceleration. ΔT is 3 ms since the quadcopter takes a sample every 3 ms. The same equation of motion will be used to estimate the position of the quadcopter in the x and y axis. Therefore **A**, **B** and **C** equates to

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \tag{3.32}$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{2}\Delta T \\ \Delta T \end{bmatrix} \tag{3.33}$$

Since only the position was required the **C** becomes

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{3.34}$$

The noise terms $\boldsymbol{\varepsilon}$ and $\boldsymbol{\delta}$ was assumed to be zero. The final motion model is

$$\mathbf{X}_{t+1} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \mathbf{X}_t + \begin{bmatrix} \frac{1}{2}\Delta T \\ \Delta T \end{bmatrix} \tag{3.35}$$

and observation model is

$$\mathbf{Z}_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{X}_t \tag{3.36}$$

3.4 FLIGHT CONTROLLER

This section describes the stability controller for the quadcopter and mathematically describes the functions being used, based on the roll, pitch and yaw angles that are provided from the IMU sensor. The altitude controller is described, based on the barometeric pressure sensor, and the position controller is described according to the GPS and magnetometer. The altitude and position controller also describe the involvement of the estimation filter. A number of flowcharts are featured in this section to assist with the explanations.

3.4.1 Stability controller

To stabilise the frame of the quadcopter, a controller was required. The stability controller consists of three parts, namely a proportional component, an integral component and the derivative component, often known as a Proportional, Integral and Derivative (PID) controller [5, 10, 11, 12, 56]. The proportional component consists of a proportional constant (K_p) and a calculated error ($e(t)$). In the case of the quadcopter, the proportional constant (K_p) was a tunable parameter and the calculated error ($e(t)$) was the difference between the IMU measured angle (roll, pitch and yaw) and the desired angle of the quadcopter [11]. The proportional component was calculated as [12]

$$Controller_P = K_p e(t). \tag{3.37}$$

The derivative component was used for a smooth response and was often used to prevent rapid changes in motion. The derivative component consists of a derivative constant (K_D) and the derivative of the error ($e(t)$). In the case of the quadcopter, the derivative constant (K_D) was a tunable parameter and the derivative of the error ($e(t)$) was the difference between the previous IMU angle and the current IMU angle [5, 11]. The derivative component was calculated as [12]

$$Controller_D = K_D \frac{\partial e(t)}{\partial t}. \tag{3.38}$$

In some cases, a PD controller is sufficient; however, to increase stability an integral component was added. The integral component consists of an integral constant (K_I) and the integral of the error ($e(t)$). In the case of the quadcopter, the integral constant (K_I) was a tunable parameter and the integral of the error ($e(t)$) is the accumulation of errors over a period of time [5, 11]. The integral component was calculated as [12]

$$Controller_I = K_I \int e(t)dt. \tag{3.39}$$

Therefore, the PID controller can be written as in its simplest form

$$PID = Controller_P + Controller_I + Controller_D. \tag{3.40}$$

Equation (3.40) can be expanded and can be written as [11, 12]

$$PID = K_p e(t) + K_I \int e(t)dt + K_D \frac{\partial e(t)}{\partial t}. \tag{3.41}$$

Figure 3.4 shows a flowchart for the stability controller. The stability controller first retrieved the gyroscope and accelerometer data from the IMU module. Each component of the PID controller was then calculated in the next three steps. The PID controller was then computed for the roll, pitch and yaw axes.

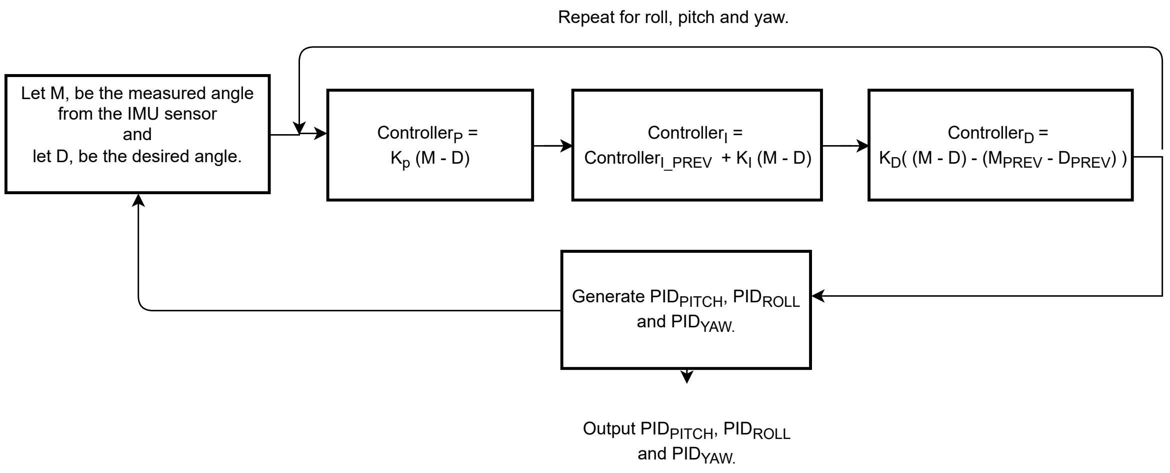

Repeat for roll, pitch and yaw.
Let M, be the measured angle from the IMU sensor and let D, be the desired angle.
Controller$_P$ = K$_p$ (M - D)
Controller$_I$ = Controller$_{I\_PREV}$ + K$_I$ (M - D)
Controller$_D$ = K$_D$((M - D) - (M$_{PREV}$ - D$_{PREV}$))
Generate PID$_{PITCH}$, PID$_{ROLL}$ and PID$_{YAW}$.
Output PID$_{PITCH}$, PID$_{ROLL}$ and PID$_{YAW}$.

Figure 3.4. PID algorithm for stability controller.

3.4.2 Altitude controller

The stability controller was used to stabilise the frame of the quadcopter; however, the stability controller did not control the altitude of the quadcopter. The altitude controller made use of a barometer sensor to measure the altitude of the quadcopter [11, 57]. A barometer sensor measures pressure and the temperature to compute the altitude of the quadcopter. The MS5611 barometric sensor was chosen because it operated between 3.3 - 5V [58] and had a resolution of 10 cm [58]. The MS5611 specifications produced high precision, which was favourable for this application. Table 3.2 shows the factory calibration variables that was read from the barometer sensor.

Table 3.2. Factory calibration of the barometer sensor [58].

Variables	**Description**
C1	Pressure sensitivity
C2	Pressure offset
C3	Temperature coefficient of pressure sensitivity
C4	Temperature coefficient of pressure offset
C5	Reference temperature
C6	Temperature coefficient of the temperature

The variables found in Table 3.2 were further used to calculate the pressure at a given altitude. The temperature reading and the pressure reading were represented by D1 and D2, respectively. The difference between the digital temperature value and the reference temperature was calculated by [58]

$$dT = D2 - C5. \tag{3.42}$$

The actual temperature was calculated by [58]

$$TEMP = 2000 + dT\frac{C6}{2^8}. \tag{3.43}$$

The offset at the actual temperature was then calculated by [58]

$$OFF = 2^{16} \times C2 + \frac{C4 \times dT}{2^7}. \tag{3.44}$$

The sensitivity at the actual temperature was calculated by [58]

$$SENS = 2^{15}C1 + \frac{C3 \times dT}{2^8}. \tag{3.45}$$

Finally, the actual pressure was calculated by [58]

$$P = \frac{\frac{D1 \times SENS}{2^{21}} - OFF}{2^{15}}. \tag{3.46}$$

Equations (3.42)-(3.46) can be found in the MS5611 barometer sensor data sheet [58]. An issue with the MS5611 barometric sensor was that it was sensitive to light; therefore, to mitigate this issue, an enclosure was required. The designed enclosure had to prevent light from reaching the sensor, as well as to have adequate airflow for the barometric sensor to measure the air pressure.

For the quadcopter to react in a timely manner the refresh rate of the quadcopter was required to be 3 ms. However, the MS5611 sensor takes approximately 9.04 ms [58] to convert the pressure or temperature data after the data had been requested. This meant that to get temperature and pressure readings it would have taken approximately 18.08 ms and the altitude could only be calculated approximately every 5 cycles. It was noticed that the temperature did not change significantly as the altitude increased; therefore, to increase the refresh rate from the barometer, a temperature reading was taken only every 0.5 s. This increased the refresh rate dramatically and the actual altitude could be computed approximately every 3 cycles. In every cycle the Kalman filter was used to predict a new pressure value and to reduce noise from the barometer readings. In addition to off-loading some of the controller calculations, the altitude controller calculated the pressure the quadcopter required to fly, based on the desired height. By rearranging (2.15), the pressure can be calculated as

$$P = P_0 \left[\left[1 - \frac{h}{44.33 \times 10^3}\right]^{(5255 \times 10^{-3})} \right]. \tag{3.47}$$

Once the altitude had been calculated, the altitude value was passed through a PID controller to allow the quadcopter to maintain altitude [5, 10, 59]. The motors were manipulated by using (3.48)-(3.51) [11, 10], with reference to Figure 3.2.

$$Motor_1 = PWM - PID_{PITCH} + PID_{ROLL} - PID_{YAW} + PID_{ALTITUDE} \tag{3.48}$$

$$Motor_2 = PWM + PID_{PITCH} + PID_{ROLL} + PID_{YAW} + PID_{ALTITUDE} \tag{3.49}$$

$$Motor_3 = PWM + PID_{PITCH} + PID_{ROLL} - PID_{YAW} + PID_{ALTITUDE} \tag{3.50}$$

$$Motor_4 = PWM - PID_{PITCH} - PID_{ROLL} + PID_{YAW} + PID_{ALTITUDE} \tag{3.51}$$

Figure 3.5 shows a flowchart of the quadcopter's altitude controller.

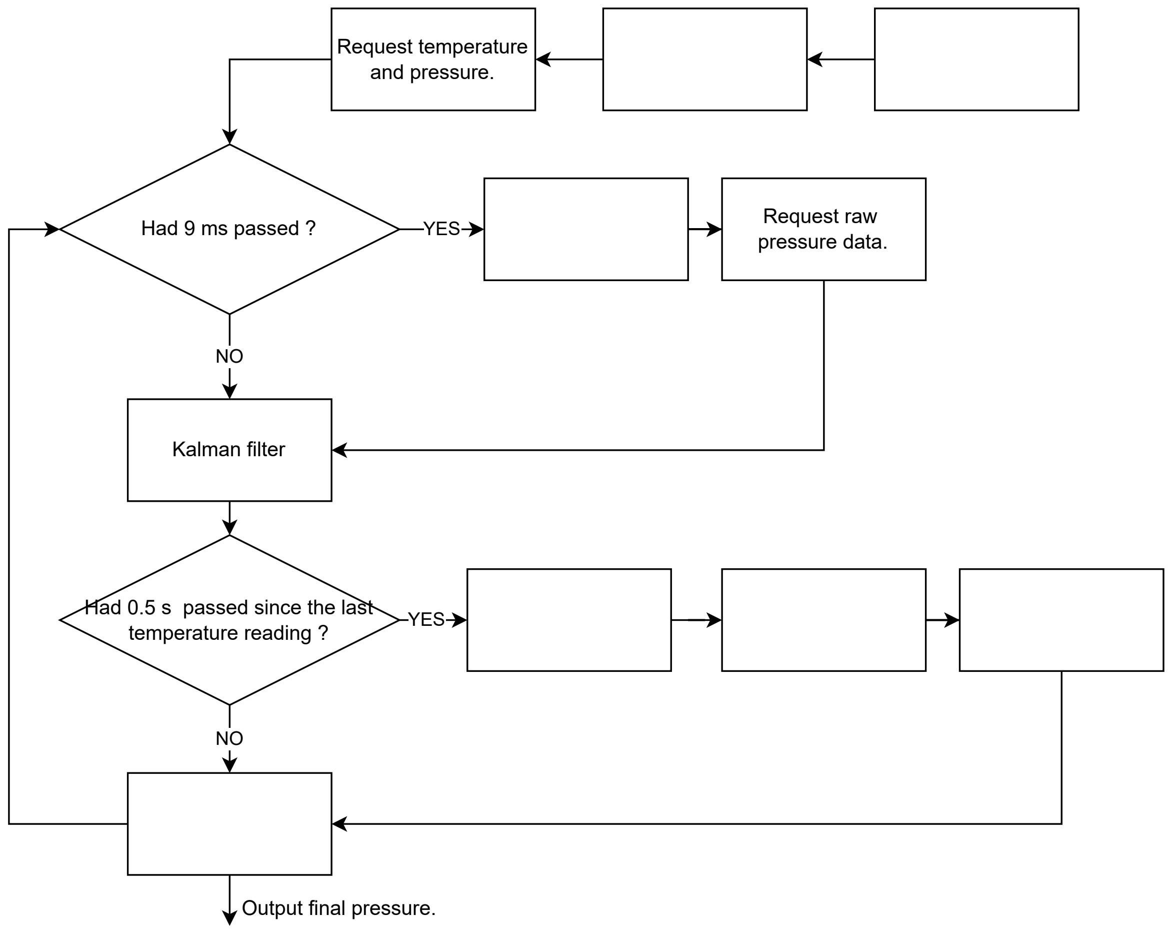

Request temperature and pressure.
Had 9 ms passed ?
YES
Request raw pressure data.
NO
Kalman filter
Had 0.5 s passed since the last temperature reading ?
YES
NO
Output final pressure.

Figure 3.5. Altitude controller.

3.4.3 Position controller

The data from the GPS module and the accelerometer were combined to produce an accurate distance estimation [60, 61, 62]. The data were combined by using the Kalman filter. The acceleration data was used in the prediction step (3.25) and the latitude and longitude from the GPS module were used in the update step (3.29) [60, 61, 62]. Table 3.3 shows the NMEA message structure being used by the GPS module.

Table 3.3. NMEA message structure [14].

NMEA message structure	
$	The NMEA start character.
Talker ID	GP and GN if the message ID is RMC, GSA or GLL. GL if the message ID is GSV. GP if the message ID is other.
NMEA message ID	NMEA message ID.
Data field	Delimited by a comma ’,’
*	End character of data field
Checksum	A hexadecimal number calculated by exclusive OR of all characters between ’$’ and ’*’
<CR><LF>	End character of NMEA message

The u-blox NEO M8N GPS module has a default baud rate of 9600 bps and a default refresh rate of 1 Hz [63]. The refresh rate of the GPS module was increased to 10 Hz and the baud rate was set to 57600 bps. Furthermore, the NMEA packet –GSV message ID was removed to reduce the size of the NMEA output message. For the positon controller the latitude and longitude are required. The number of satellites are also required to determine the accuracy of the GPS coordinates. Table 3.4 shows the example of the GPS NMEA message output.

Table 3.4. Example of NMEA message structure [14].

NMEA message structure.
$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M, , , ,0000*18
$GPGLL,3723.2475,N,12158.3416,W,161229.487,A,A*41
$GPGSA,A,3,07,02,26,27,09,04,15, , , , , ,1.8,1.0,1.5*33
$GPRMC,161229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598, ,*1
$GPVTG,309.62,T, ,M,0.13,N,0.2,K,A*23

Figure 3.6 shows the flowchart for the position controller.

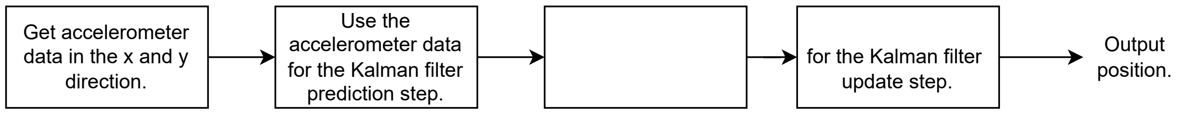

Get accelerometer data in the x and y direction.
Use the accelerometer data for the Kalman filter prediction step.
for the Kalman filter update step.
Output position.

Figure 3.6. Position estimation controller.

3.4.3.1 Magnetometer

To determine that the direction the quadcopter is facing, a magnetometer is required. The magnetometer is used to detect the magnetic field on earth. The magnetometer can measure the strength of the magnetic field in a three-dimensional space [64]. The magnetometer outputs an electrical voltage that is related to the strength of the magnetic field. Figure 3.7 shows a two-dimensional representation of how the magnetic north is calculated. The force of the magnetic field in the x direction and the y direction are represented by MX and MY, respectively. If Y = 0 is assumed to be the front of the quadcopter, then β represents the angle between the quadcopter and the magnetic north.

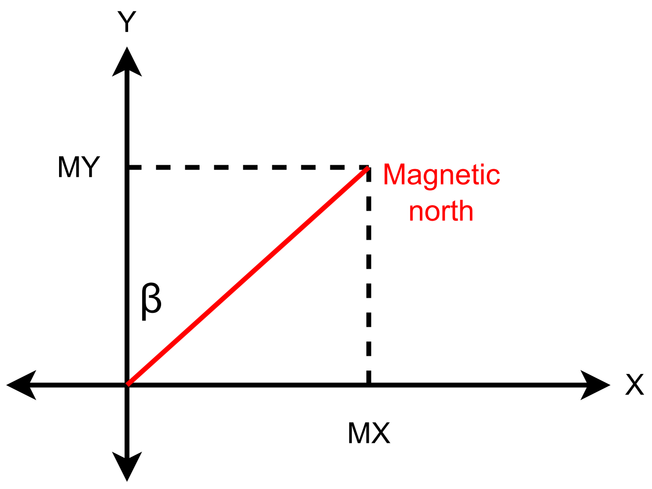


Figure 3.7. Two dimensional representation of calculating the magnetic north.

The angle between the quadcopter and the magnetic north can be calculated by

$$\beta = \arctan\left(\frac{MY}{MX}\right). \tag{3.52}$$

The issue with the magnetometer sensor is that it outputs an electrical voltage related to the magnetic north; however, the GPS module output coordinates with respect to the geographic north. Figure 3.8 shows the magnetic north in red and the geographic north in blue.

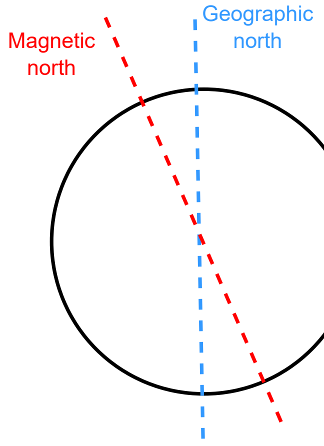


Figure 3.8. Angle difference between the magnetic north and the geographic north.

The representation shown in Figure 3.8, is a simple diagram and the difference between the magnetic north and the geographic north is different across the world. Fortunately, a world magnetic model exists [65] that provides the difference between the magnetic north and the geographic north. In South Africa the difference is approximately -20° [65]. Figure 3.9 shows the way in which to calculate the geographic north, where τ is the value found from the world magnetic map.

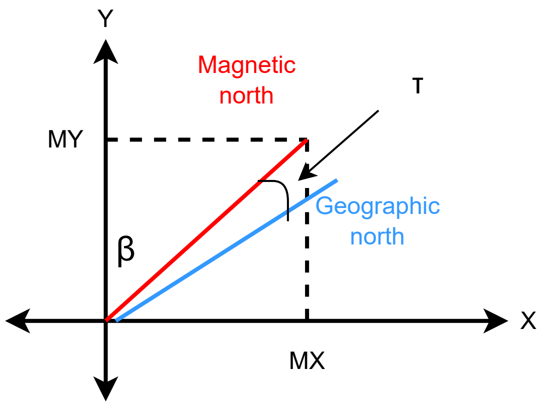


Figure 3.9. Relationship between the magnetic north and the geographic north.

The geographic (g) north is mathematically calculated by

$$g = \beta - \tau. \tag{3.53}$$

A rotational matrix is applied to the magnetometer outputs to compensate if the compass is tilted during flight. The adjusted equations are represented by

$$X_H = X\cos\theta + Y\sin\theta\sin\phi - Z\cos\theta\sin\phi \tag{3.54}$$

and

$$Y_H = Y\cos\theta + Z\sin\theta, \tag{3.55}$$

where the magnetometer outputs are represented by X, Y and Z. The pitch of the quadcopter is represented by θ and the roll is represented by ϕ. The compensated azimuth can be calculated by

$$Azimuth = \arctan\left(\frac{Y_H}{X_H}\right), \tag{3.56}$$

where, X$_H$ represents the earth's magnetic field in the x-axis and Y$_H$ represents the earth's magnetic field in the y-axis. However, by using only the earth's horizontal magnetic field to calculate the heading, an error in the heading will occur if the quadcopter is tilted (pitch and roll). To minimise this issue, the roll(θ) and pitch(ϕ) from the gyroscope can be used [66].

The magnetometer sensor experiences some noise and therefore, the magnetometer sensor has to be calibrated to improve the accuracy of the sensor. The magnetometer sensor noise consists of hard iron noise and soft iron noise. Hard iron noise introduced by metals are difficult to magnetise but stay magnetised for a long period of time [67, 64]. Soft iron noise introduced by metals are easy to magnetise and do not stay magnetised for a long period of time [67, 64]. To reduce the noise, the magnetometer sensor is rotated around the pitch and roll axes. The highest value and lowest value for both the pitch (X_{min}, X_{max}) and roll (Y_{min}, Y_{max}) axes are recorded [68, 64]. The scaling factor is then calculated by [64]

$$X_{sf} = \frac{Y_{max} - Y_{min}}{X_{max} - X_{min}}, \tag{3.57}$$

and

$$Y_{sf} = \frac{X_{max} - X_{min}}{Y_{max} - Y_{min}}. \tag{3.58}$$

The offset values are then calculated by [64]

$$X_{off} = X_{sf} \times \frac{X_{max} - X_{min}}{2 - X_{max}}, \tag{3.59}$$

and

$$Y_{off} = Y_{sf} \times \frac{Y_{max} - Y_{min}}{2 - Y_{max}}. \tag{3.60}$$

Finally, the calibrated values are calculated by [64]

$$X_H = X_H - X_{off}, \tag{3.61}$$

and

$$Y_H = Y_H - Y_{off}. \tag{3.62}$$

The final heading is calculated as [66]

$$Azimuth = \left\{ \begin{array}{ll} 180^\circ - arctan(\frac{Y_h}{X_h}) & X_h < 0 \\ -arctan(\frac{Y_h}{X_h}) & X_h > 0, Y_h < 0 \\ 360^\circ - arctan(\frac{Y_h}{X_h}) & X_h > 0, Y_h > 0 \\ 90^\circ & X_h = 0, Y_h < 0 \\ 270^\circ & X_h = 0, Y_h > 0 \end{array} \right\}. \tag{3.63}$$

3.4.4 Propellers and battery compensation

The propeller selection is a crucial selection for any quadcopter. The most common type of propellers being used, is a two-blade propeller. The advantages of a two-blade propeller are that it requires less power to make a revolution and allows the quadcopter to maneuver faster in the air. The three-blade propellers provide better stability for the quadcopter but sacrifices maneuverability. The three-blade propeller improves stability, as it has one extra point of contact on the air when compared to the two-blade propeller.

Another challenge to resolve is that the quadcopter usually reduces in performance as the LiPo battery voltage reduces, therefore, a small compensation was necessary to allow the quadcopter to perform the same, regardless of the battery voltage. The battery compensation was calculated as

$$throttle+ = BCG \times (V_{max} - V_{current}), \tag{3.64}$$

where V_{max} is the maximum possible battery voltage, $V_{current}$ is the current voltage of the battery and BCG is a predefined battery compensation gain.

3.5 VISION SENSOR

The vision sensor chosen for the real-time image processing applications was a Caddx Ratel First-Person View (FPV) camera. The Caddx Ratel camera was chosen for being lightweight and small in size. The Caddx Ratel camera can operate in low light as well and has a FOV of 160°. The Caddx Ratel camera was also favourable, as it can be powered using, 5 V and has a low latency of 8 ms. A summary of the specifications of the Caddx Ratel FPV is listed in Table 3.5.

Table 3.5. Caddx Ratel FPV camera specifications.

Caddx Ratel FPV camera specifications	
Sensor type	CMOS
Horizontal resolution	1200 TVL
Lens	2.1 mm
Dimensions	19x19x19 mm
Weight	8 g
Voltage supply	DC 5 - 40 V
Image	16:9 or 4:3
FOV	160 °
Latency	8 ms
Illumination	0.0001 Lux

The Caddx Ratel camera also provides a wide range of customisation that can be used for image enhancing such as contrast, sharpness, color gain, day/night mode, brightness and exposure.

To measure the image distortion, the MSE, PSNR and SSIM measurements [29, 30] are used. The MSE and PSNR are defined, respectively, as

$$MSE = 10\log 10\left[\frac{1}{N\times N}\sum_{i=0}^{N\times N}(X_i - Y_i)^2\right], \tag{3.65}$$

and

$$PSNR = 10\log 10\left(\frac{255^2}{MSE}\right), \tag{3.66}$$

where *NxN* represents the rectangle region and X and Y are the images. A lower MSE indicates that the image has a small amount of distortion. Since the MSE is in the denominator of the PSNR equation, a higher PSNR indicates a small amount of distortion in an image.

The SSIM measurement consists of three components, namely: luminance, contrast and similarity [29, 30]. The luminance, contrast and similarity component are respectively calculated by,

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \tag{3.67}$$

$$C(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \mu_y^2 + C_2}, \tag{3.68}$$

$$S(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}. \tag{3.69}$$

where x and y are the images, σ is the variance and μ is the mean. The SSIM equation is then calculated by [29, 30]

$$SSIM = l(x,y)^{\alpha} \times C(x.y)^{\beta} \times S(x.y)^{\gamma}. \tag{3.70}$$

where α, β and γ are used to adjust the importance of the each component. A SSIM value closer to 1 indicates that the image experiences a smaller amount of distortion.

3.6 TARGET DETECTION

The YOLO method is the state of the art, real-time method that is commonly used for target detection [39, 40, 49, 50]. Figure 3.10 shows the flowchart of the YOLO method.

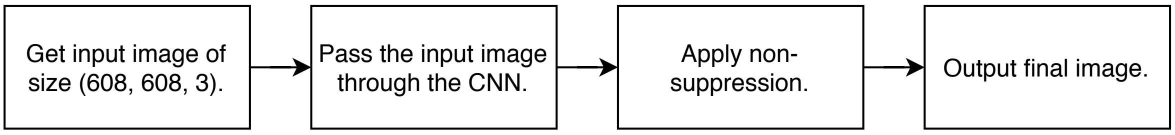

Get input image of size (608, 608, 3).
Pass the input image through the CNN.
Apply non-suppression.
Output final image.

Figure 3.10. Flowchart for the YOLO target detection.

3.6.1 YOLO network

The YOLO used an input image of (608,608,3). The input image was divided in 19x19 grid cells, equating to 361 individual cells. The YOLO network consisted of 23 convolutional layers and 5 max pooling layers. The YOLO network also used batch training; therefore, it was important to use batch normalisation to maintain the distribution of the training data [49, 50]. Table 3.6 shows a summary of the YOLO network.

Table 3.6. Summary of YOLO network

Input image size	(608,608,3)
Number of convolutional layers	23
Number of pooling layers	5

As mentioned previously, the convolutional layers used an activation function [39, 40, 49, 50]. The activation function being used in the YOLO network, was the leaky ReLU function. The leaky ReLU is a non-linear activation function and prevents the activation function from resulting to zero if the input is less than zero [69, 70]. The leaky ReLU function is mathematically described by

$$f(y) = \left\{ \begin{array}{ll} x, & \text{if } x > 0. \\ ax, & \text{otherwise.} \end{array} \right\}, \tag{3.71}$$

where a represents a scaling value. The leaky ReLU function is graphically depicted in Figure 3.11, based on (3.71).

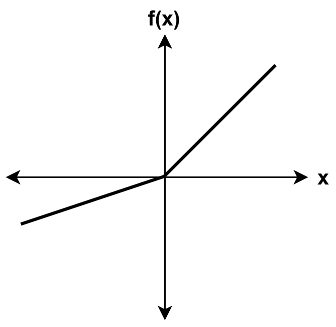


Figure 3.11. Leaky ReLU function.

As mentioned previously, the YOLO network used max pooling layers [39, 40, 49, 50]. Figure 3.12 shows an example of max pooling with a stride of two and a window size of two. The different coloured blocks represent the windows. In each window the max value was found and was represented in the 2x2 block.

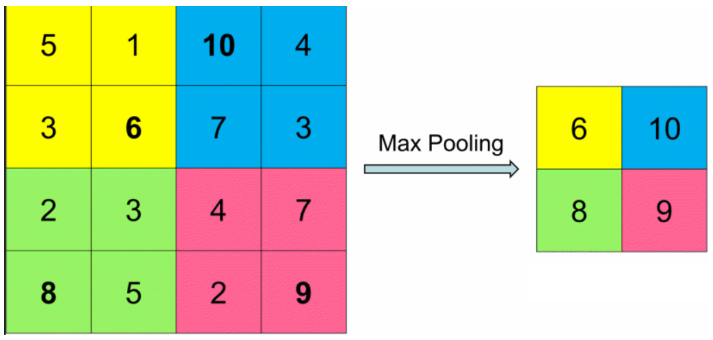


Figure 3.12. CNN architecture (Taken from [71], © 2018 IEEE.).

The YOLO algorithm uses max pooling with a stride size of 2 and a window size of 2.

3.6.2 Non-suppression method

Once the input image had passed through the YOLO network, the output from the YOLO network was passed through the non-suppression method to reduce the amount of bounding boxes. The first step was to select a confidence level threshold (p_c). Typically, p_c is 0.5 and if the p_c value was increased, then some relevant bounding boxes would have been discarded and a too low p_c value would retain unnecessary boxes. Therefore, a p_c value of 0.5 was used in the YOLO network. The bounding boxes with a high confidence level were then kept and the bounding boxes with a small IoU were discarded [39, 40, 49, 50]. Figure 3.13 shows an example of the IoU, where the grey section represents the intersection between the green and yellow bounding box. The IoU in Figure 3.13 can be calculated as

$$IoU = \frac{x_3 \times y_3}{x_1 \times y_1 + x_2 \times y_2}. \tag{3.72}$$

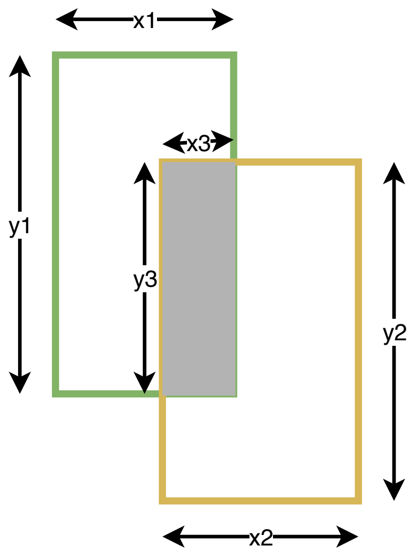


Figure 3.13. Intersection of union.

3.6.3 Colour classification

The first step for the colour classification algorithm was to segment the shirt and the pants of the person detected. Once the target tracking algorithm had identified the person and placed a bounding box around the person of interest, the upper body was assumed to be the top half of the bounding box and the bottom half of the bounding box was assumed to be the lower body of the person of interest.

Once the upper and lower body had been segmented, the K-Means algorithm was used to classify the colours. The K-Means algorithm was set to have a maximum of three clusters. The biggest cluster would determine the dominant colour in the top half of the image and the bottom half of the image. The K-Means clusters contained a hex value of the colour. A set of predefined colours were set; however, in many cases the predefined colours and the dominant colour did not match. To solve the problem, a k Nearest Neighbour (k-NN) algorithm was used to find the closest predefined colour that matches the dominant colour. Figure 3.14 is a flowchart for the colour detection algorithm.

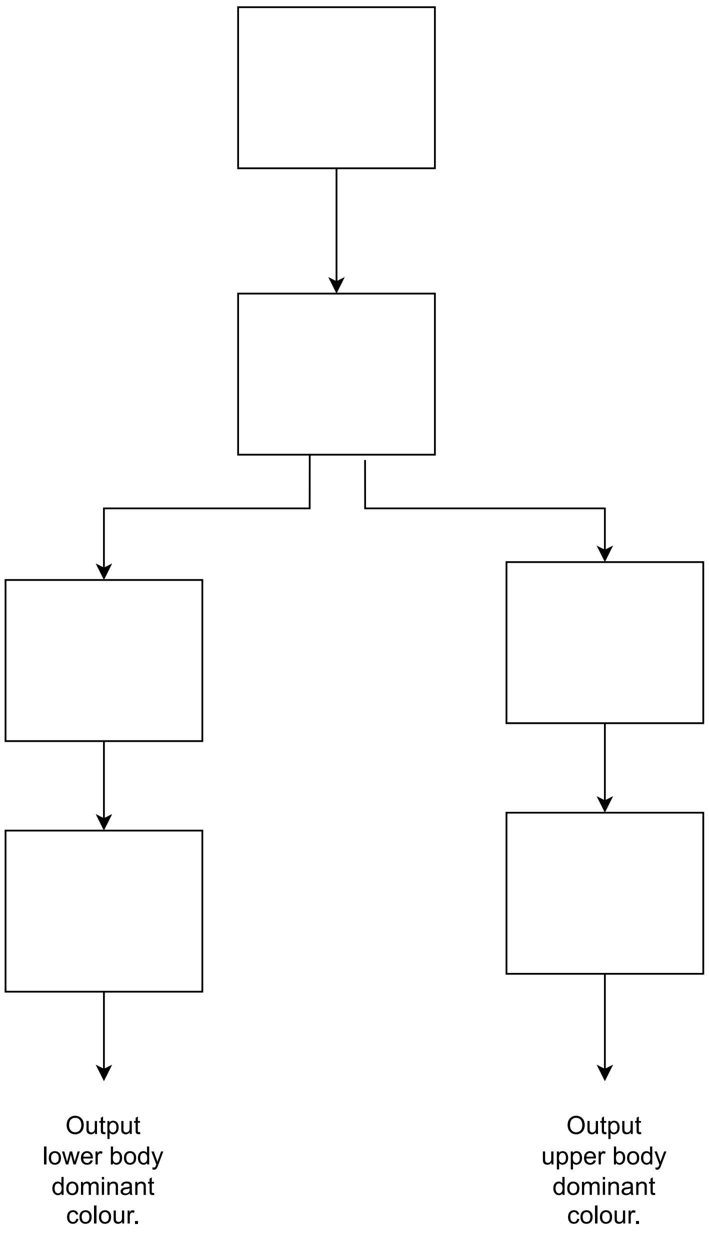


Figure 3.14. Colour detection algorithm.

3.6.4 Distance estimation

To measure the distance between a person or object, the bounding box area from the YOLO algorithm was used. The area of the bounding box was calculated by

$$Area = (y_2 - y_1) \times (x_2 - x_1) \tag{3.73}$$

where (x_1, y_1) shows one corner of the bounding box and (x_2, y_2) shows the opposite corner. It can be seen that a person with stretched out arms produced a massive change in the area of the bounding box. Therefore it was easy to discard areas that had a huge increase in the area of the bounding box, compared to the previous bounding box area.

3.7 COMMUNICATION

There are three main types of communications that are used on the quadcopter:

- Communication from the RC transmitter and the RC receiver,
- Communication from the quadcopter to the base stations,
- Communication from the base station to the quadcopter,

The quadcopter used the FlySky FS-T4B transmitter and receiver to allow the user to gain control of the quadcopter as a fail-safe mechanism [11]. The RC transmitter was capable of performing yaw, pitch, roll and throttle commands [11, 57]. The remaining types of communication were responsible for the communication between the quadcopter and the base station. This was possible by using the EACHINE TS5828L transmitter and the EACHINE ROTG01 receiver to send images from the camera to the base station for further processing. To send flight commands from the base station to the quadcopter, the HC-12 module was used [57]. Those communication devices were carefully chosen to ensure that there were no overlapping frequency bands, as overlapping frequencies can distort the signal and cause the quadcopter to experience unpredictable behaviour.

Figure 3.15 shows the low-level representation of the proposed system.

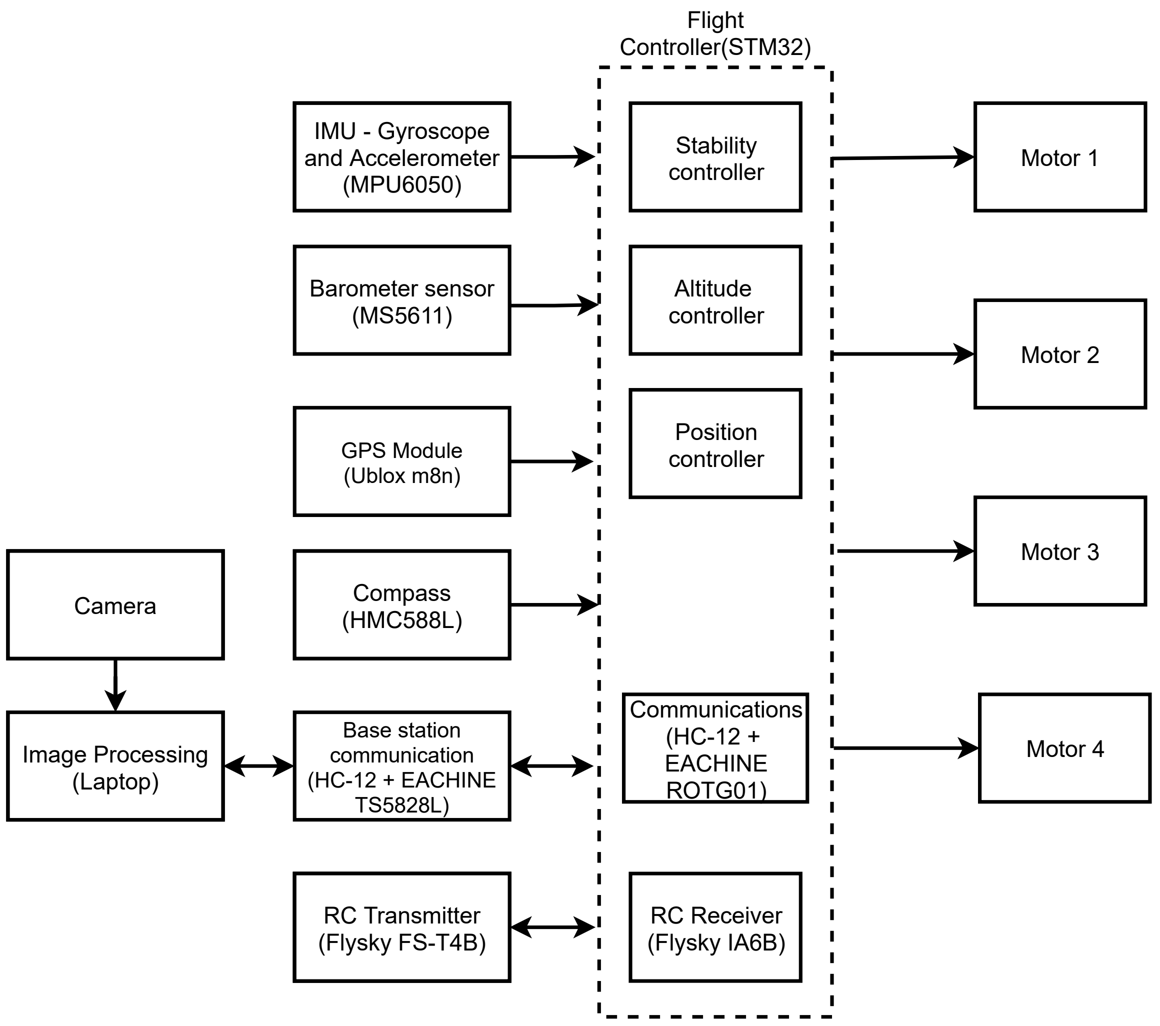

Flight Controller(STM32)
IMU - Gyroscope and Accelerometer (MPU6050)
Stability controller
Motor 1
Barometer sensor (MS5611)
Altitude controller
Motor 2
GPS Module (Ublox m8n)
Position controller
Motor 3
Camera
Compass (HMC588L)
Image Processing (Laptop)
Base station communication (HC-12 + EACHINE TS5828L)
Communications (HC-12 + EACHINE ROTG01)
Motor 4
RC Transmitter (Flysky FS-T4B)
RC Receiver (Flysky IA6B)

Figure 3.15. Low-level representation of the proposed system.

3.8 CHAPTER SUMMARY

This chapter dealt with the design and implementation of the quadcopter and the real-time image processing system. A high-level representation of the system was provided and each component function was described. The stability, position and altitude controllers were designed, implemented and described in detail with references to the appropriate sensor data. This chapter also discussed the advantages and disadvantages of the multi-blade propellers and the effects of the available battery power on the quadcopter performance. The real-time image processing algorithm being used, was not implemented from first principles; however, the working principles were discussed extensively. Finally, the communication system was discussed and the low-level representation of the system was provided.

CHAPTER 4 RESULTS AND DISCUSSION

4.1 CHAPTER OVERVIEW

This chapter presents the final quadcopter, discusses the test procedure and examine the results based on the outcome. This chapter was divided into four main sections. In Section 4.2, the physical built quadcopter is shown. In Section 4.3 the results from the flight controller is presented. This includes results for the quadcopter using the two-blade propellers, as well as three-blade propellers. Furthermore, the position and rotational estimation results are presented in Section 4.3.

In Section 4.4, the image processing results are presented, this includes the colour classification, distance estimation and image distortion tests. The image quality using the two-blade and three-blade propellers are compared in Section 4.4. Finally, in Section 4.5, the person following system is presented.

4.2 QUADCOPTER PLATFORM

Figure 4.1 shows the Caddx camera that was mounted on the quadcopter to capture images from the quadcopter.

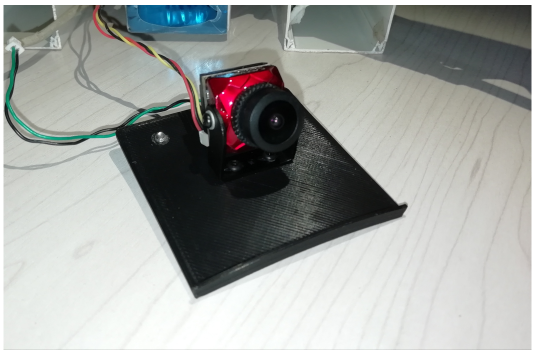


Figure 4.1. Vision sensor- Ratel FPV Caddx camera.

Figure 4.2 shows an image of the quadcopter fully-assembled. The results made use of the two-blade and three-blade propellers.

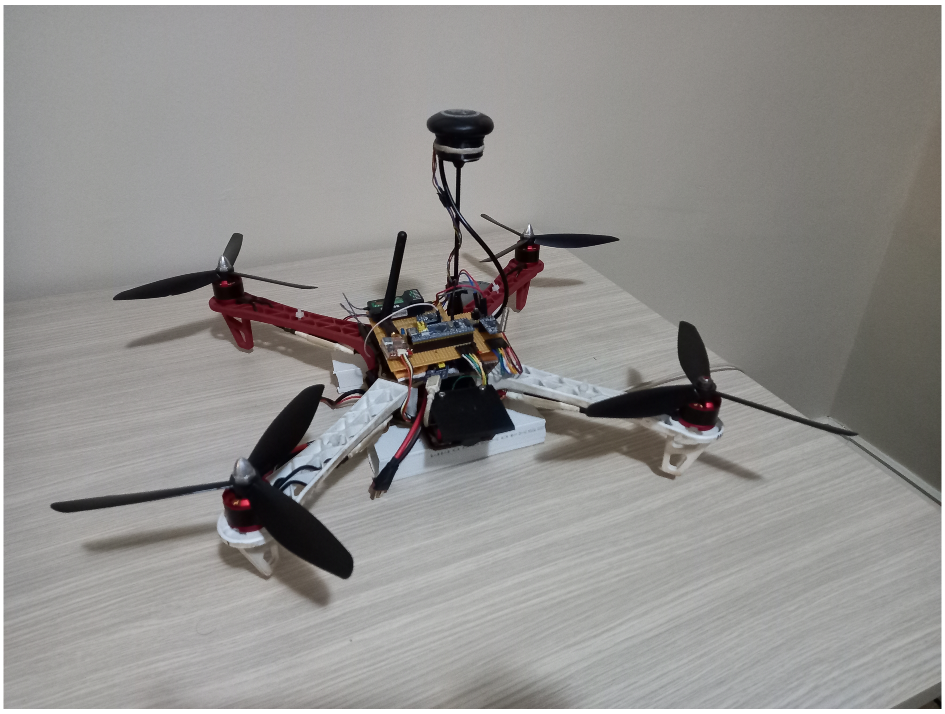


Figure 4.2. The assembly of the quadcopter.

Table 4.1 lists the components used on the quadcopter and Figure 4.3 shows the top view of the quadcopter. The IMU sensor (MPU6050) was chosen to be placed close to the middle of the quadcopter in order to get the most accurate pitch and roll angle of the quadcopter. The barometric sensor (MS5611) was placed higher than the propellers to prevent unnecessary interference from the propellers. The battery was chosen to be placed at the bottom of the quadcopter and closer to the centre of gravity, to prevent unnecessary imbalances. The camera was chosen to be placed in the front quadcopter and underneath a 3D printed plate to prevent the camera images from being saturated from the sun's glare. A step-down regulator was used to reduce the 7.4 V LiPo battery to 5 V to power the electronic components on the board. The rest of the components were placed to keep the design compact. Furthermore, to reduce the vibrations on the quadcopter frame four earplugs were placed between the quadcopter's frame and the flight controller board. Many different vibration damping techniques exist; however, the earplugs were easily accessible and reduced the vibrations.

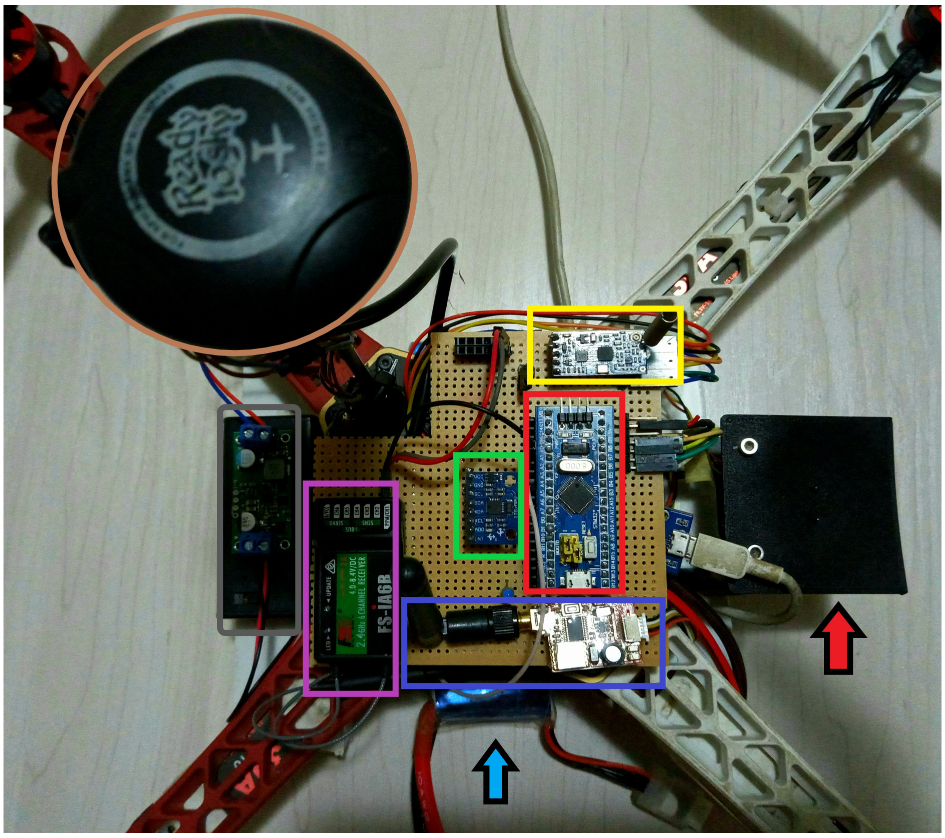


Figure 4.3. Sensors on the quadcopter.

Table 4.1. Component list on the quadcopter.

Colour	**Component**
Brown square	GPS M8N + HMC588L compass + Barometer MS5611
Yellow square	HC-12 wireless communication
Green square	IMU MPU6050
Red square	STM32F103C8T6
Blue square	Transceiver
Purple square	RC receiver
Grey square	Step-down voltage
Blue arrow	3s 3300mAh LiPo battery
Red arrow	Ratel FPV Caddx camera

4.3 FLIGHT CONTROLLER

The stability controller results were recorded by measuring the angle response of the quadcopter, by introducing a pitch or roll angle. The performance of the stability controller was measured by evaluating or calculating the peak angle, peak time, settling time, rise time and setting angle.

The position controller results were recorded by measuring the angle response of the quadcopter by setting a position of the quadcopter, based on latitude and longitude coordinates and by measuring how the quadcopter corrects itself in the air to maintain the desired position. The position estimation was also evaluated by moving the quadcopter to a number of positions and the Euclidean distance error was recorded.

The altitude controller results were recorded in a similar fashion as for the position controller. The height was indirectly set by pressure. The altitude controller's performance was measured by measuring how the quadcopter corrects its height in the air in order to maintain the desired height.

4.3.1 PID basic simulation

Figure 4.4 represents a simulation of a basic PID response in python. The blue graph represents the unit step and the yellow graph represents the PID response.

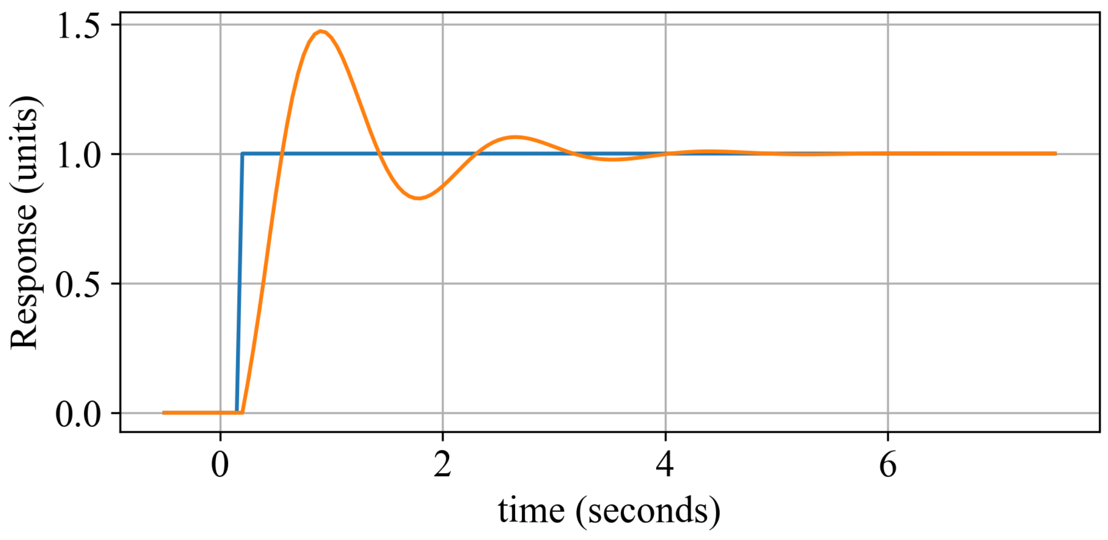


Figure 4.4. PID basic simulation.

4.3.2 PID tuning

The PID controllers initially used the PID gains found in [57], which can be seen in Table 4.2; however, the PID gains found in [57] were not adequate to produce a stable flight controller.

Table 4.2. PID gains from [57].

	K$_p$	**K$_I$**	**K$_D$**
Pitch	8.5	0	4
Roll	8.5	0	4
Yaw	14	0	10
Altitude	40	18	12

The stability controller gains found in [56], were then used, as shown in Table 4.3. The stability controller gains used in Table 4.3 did not work perfectly; however, it had a better response compared to [57]. Therefore, the stability controller gains used in [56], were used as a reference point.

Table 4.3. PID gains from [56].

	K$_p$	**K$_I$**	**K$_D$**
Pitch	0.7	0.15	0.035
Roll	0.75	0.20	0.040
Yaw	3.0	0.00	0.00

Once the reference point for the stability controller had been established, the stability controllers were further improved on a trial and error procedure and based on the stability controller response.

The P gain was first tuned until the quadcopter frame started to overcompensate for the correction angle errors of the quadcopter. Once the overcompensated value had been found, the P gain was reduced until the quadcopter had a decent response. The D gain was then tuned and with an adequate D gain it was possible to safely fly the quadcopter at a higher altitude. The I gain was finally tuned for minor improvements to the stability controllers' performance. The altitude and position controller followed the same procedure. This procedure was then used for both propeller set-up.

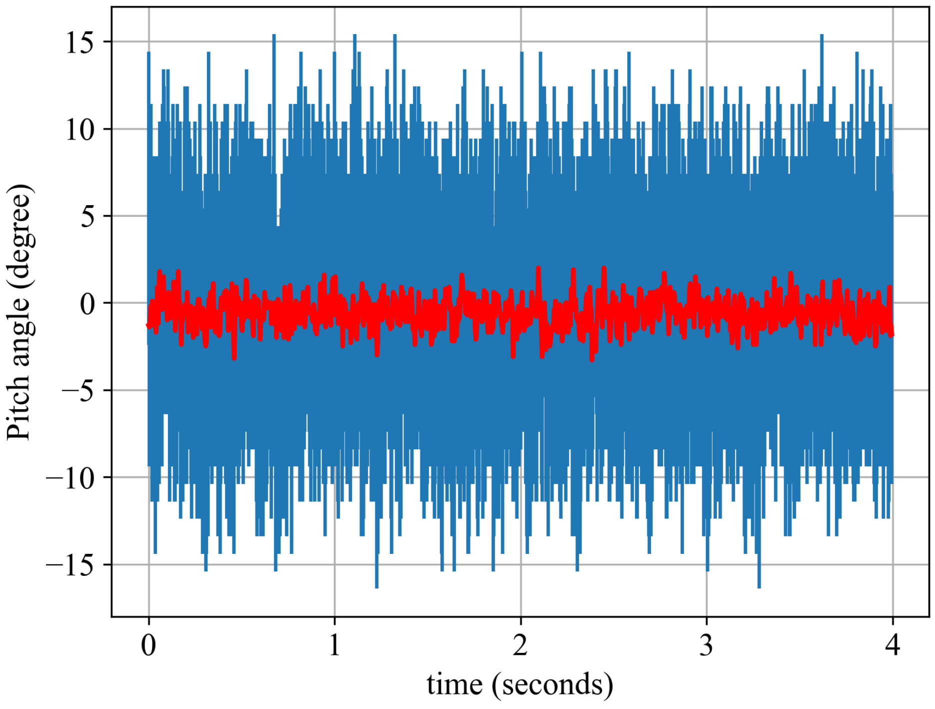
4.3.3 Estimation Kalman filter

This test compared the pitch angle of the quadcopter when the Kalman filter was being used. as well as when the Kalman filter was not being used. In this test the quadcopter was placed on a flat surface and the pitch and roll of the quadcopter was recorded. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests. Figure 4.5(a) and Figure 4.6(a) show the raw response from the quadcopter. Figure 4.5(b) and Figure 4.6(b) reflect the filtered response from the quadcopter.

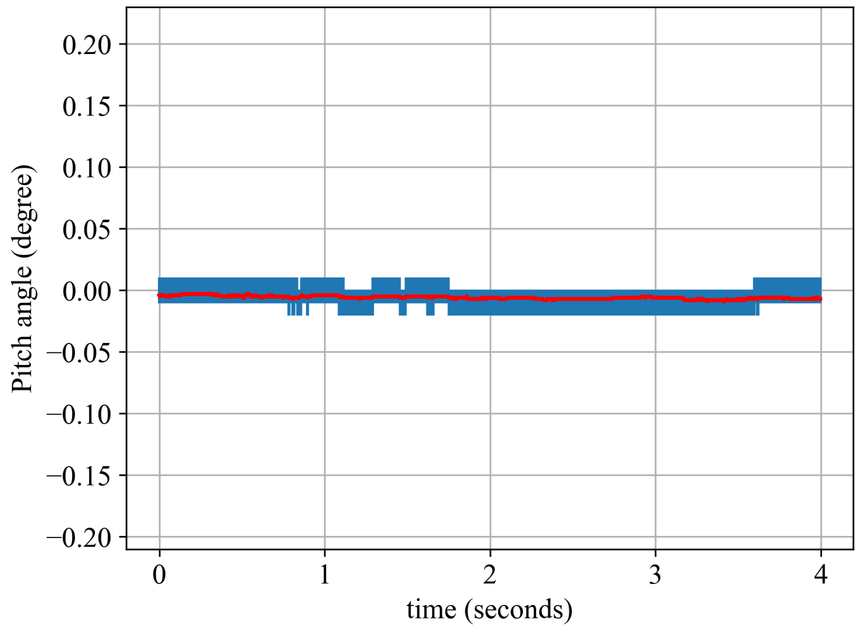
Table 4.4 shows the Mean Absolute Error (MAE) of the pitch and roll angle.

Table 4.4. Pitch and roll mean absolute error.

MAE	**Raw angle** (°)	**Filtered angle** (°)
Pitch	0.821	0.076
Roll	0.653	0.049

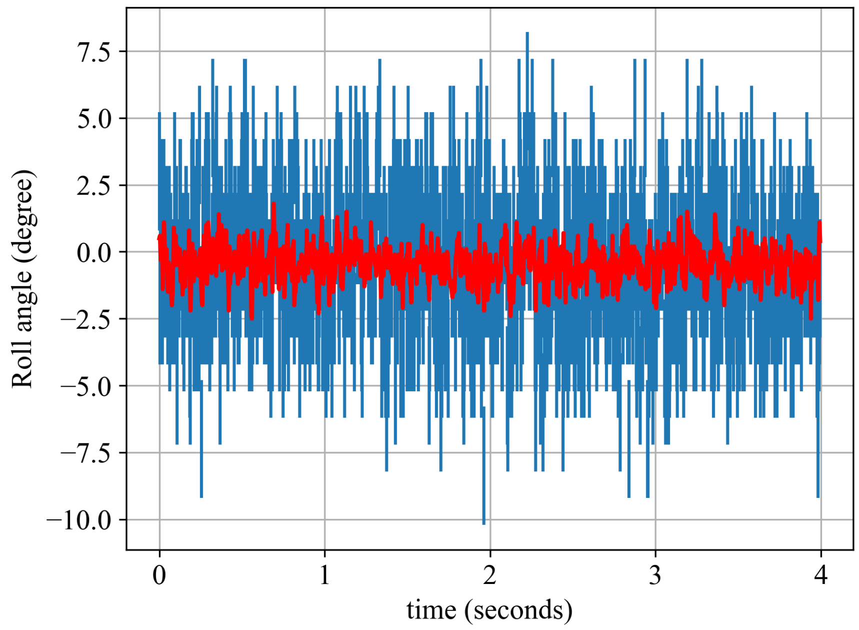


(a) Raw sensor pitch data. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

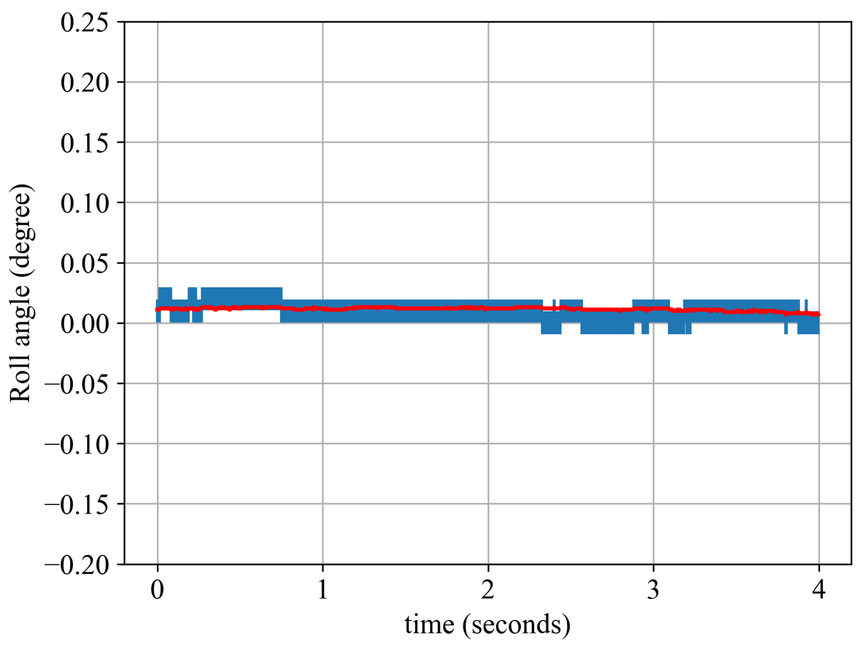


(b) Kalman filtered pitch angle. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Figure 4.5. Raw sensor pitch data vs. Kalman filtered pitch angle estimation.



(a) Raw sensor roll data. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.



(b) Kalman filtered roll angle. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Figure 4.6. Raw sensor roll data vs. Kalman filtered roll angle estimation.

It was concluded that the Kalman filter would suffice for the quadcopter, as it made a huge improvement to the quadcopter's pitch and roll angle and was also less computationally expensive to other estimation filtering methods, such as the particle filter.

4.3.4 Two-blade propellers

The stability controller proportional gain (P) was set to 0.92, the integral gain (I) was set to 0.04 and the derivative gain (D) was set to 16.5. The position controller proportional gain (P) was set to 2.8, the integral gain (I) was set to 0.01 and the derivative gain (D) was set to 4.2. The altitude controller proportional gain (P) was set to 1.1, the integral gain (I) was set to 0.2 and the derivative gain (D) was set to 0.95. Table 4.5 summarises the PID gains being used for the stability, position and altitude controllers.

Table 4.5. PID gains for the two-blade propellers.

Two-Blade propellers			
	Stability controller	**Position controller**	**Altitude controller**
Proportional gain (P)	0.92	2.8	1.1
Integral gain (I)	0.04	0.01	0.2
Derivative gain (D)	16.5	4.2	0.95

4.3.4.1 Stability controller

Figure 4.7 shows the pitch angle correction response of the quadcopter. The test was repeated twenty times and the variance is illustrated by twenty blue graphs. In this tests, the quadcopter was disturbed by a negative pitch angle. The average of the twenty tests is represented by the red graph. The results of the average forward pitch angle correction response can be seen in Table 4.6.

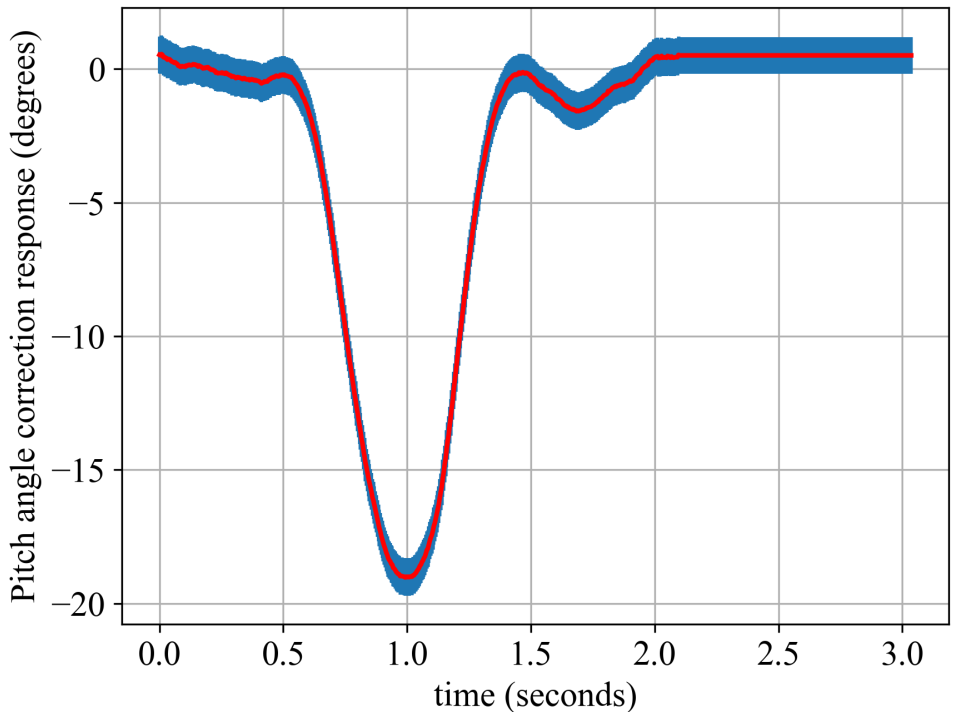


Figure 4.7. Pitch forward correction angle response without position mode activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.6. Characteristics of pitch forward correction angle response without position mode activated graph.

Peak angle (°)	0.3
Peak time (s)	2.1
Settling time (s)	1.8
Rise time (s)	0.7
Settling angle (°)	0.2

Figure 4.8 shows the pitch angle correction response of the quadcopter. In this tests, the quadcopter was disturbed by a positive pitch angle in this test. The test was repeated twenty times and the variance is illustrated by twenty blue graphs. The average of the twenty tests is represented by the red graph. The results of the average backward pitch angle correction response can be seen in Table 4.7.

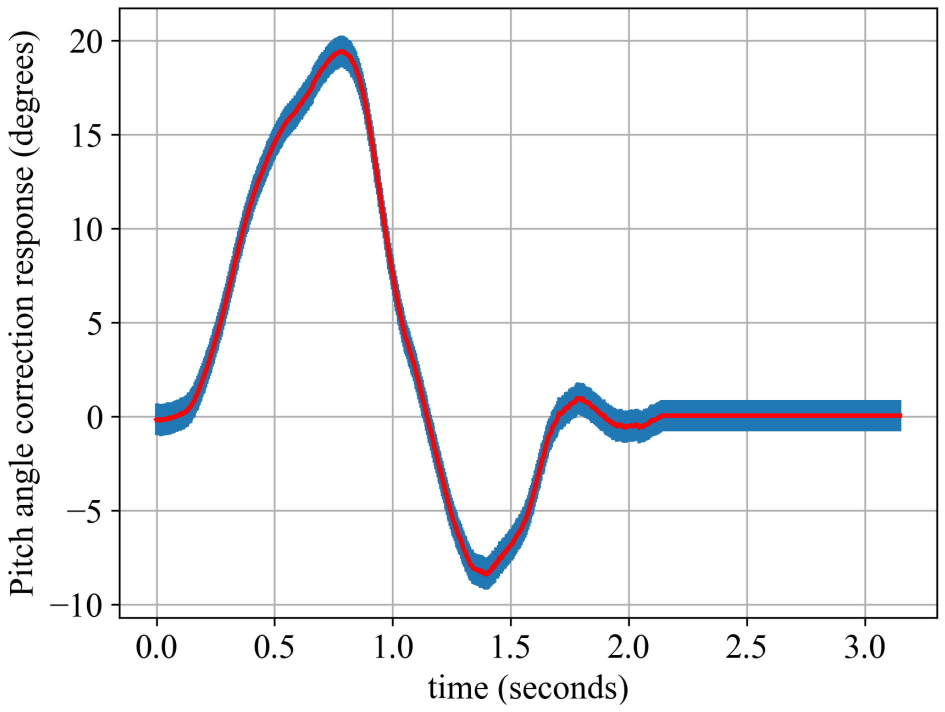


Figure 4.8. Pitch backward correction angle response without position mode activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.7. Characteristics of pitch backward correction angle response without position mode activated graph.

Peak angle (°)	-7.2
Peak time (s)	1.39
Settling time (s)	2.3
Rise time (s)	0.75
Settling angle (°)	0.192

Figure 4.9 shows the roll angle correction response of the quadcopter. In this tests, the quadcopter was disturbed by a negative roll angle. The test was repeated twenty times and the variance is illustrated by twenty blue graphs. The average of the twenty tests is represented by the red graph. The results of the average left roll angle correction response can be seen in Table 4.8.

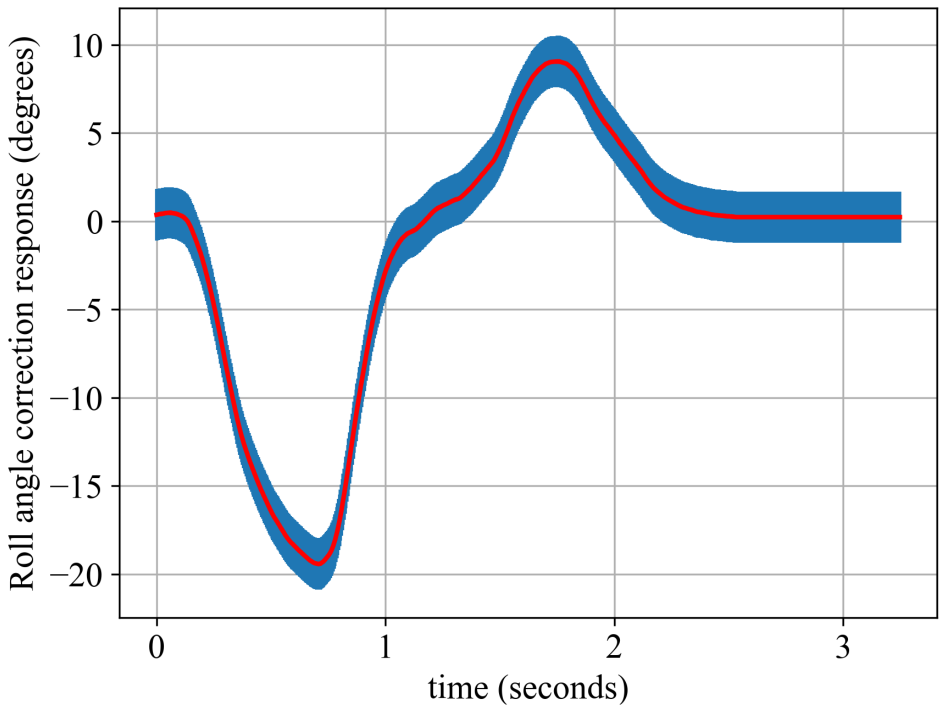


Figure 4.9. Roll left correction angle response without position mode activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.8. Characteristics of roll left correction angle response without position mode activated graph.

Peak angle (°)	8.21
Peak time (s)	1.75
Settling time (s)	2.5
Rise time (s)	0.73
Settling angle (°)	0.3

Figure 4.10 shows the roll angle correction response of the quadcopter. In this tests, the quadcopter was disturbed by a positive roll angle. The test was repeated twenty times and the variance is illustrated by twenty blue graphs. The average of the twenty tests is represented by the red graph. The results of the average right roll angle correction response can be seen in Table 4.9.

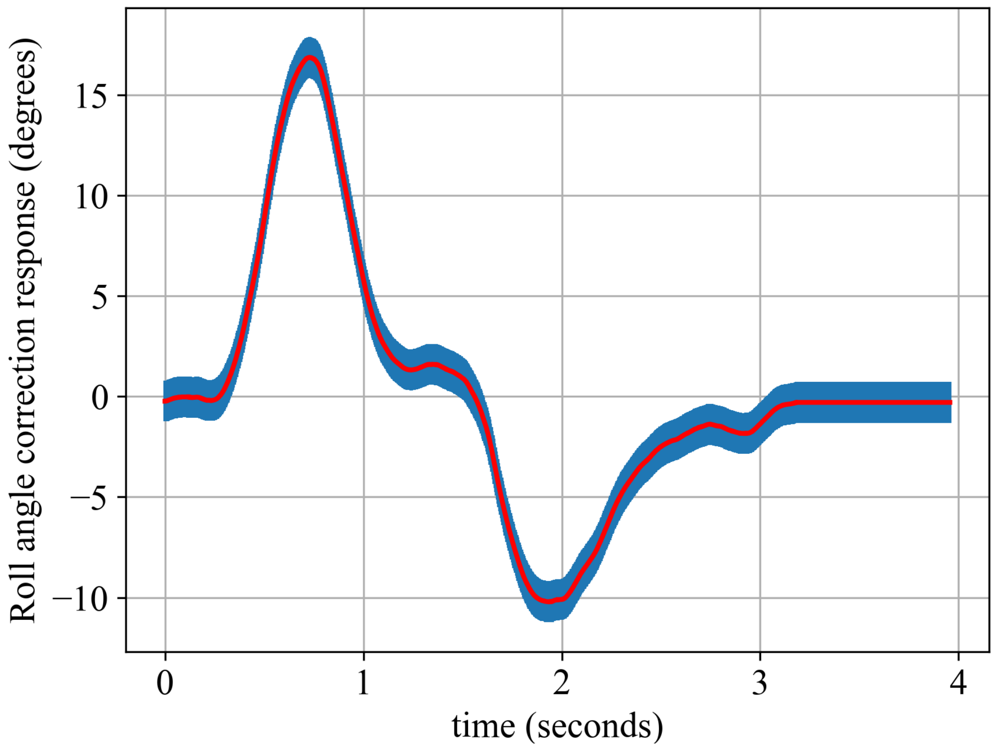


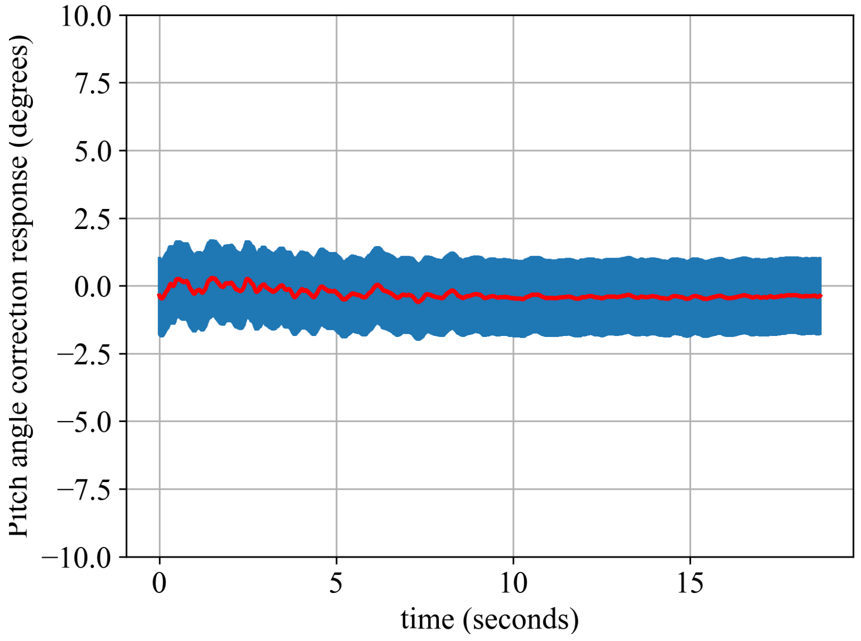
Figure 4.10. Roll right correction angle response without position mode activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.9. Characteristics of roll right correction angle response without position mode activated graph.

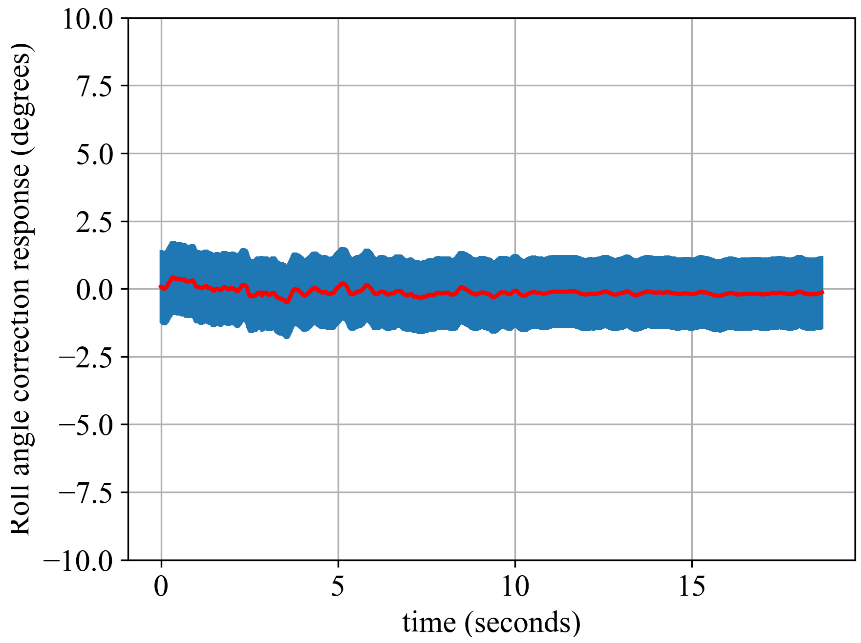
Peak angle (°)	-10.2
Peak time (s)	1.8
Settling time (s)	3.1
Rise time (s)	0.79
Settling angle (°)	-0.2

4.3.4.2 Position controller

In this test, the position hold controller was activated at a set position with no physical disturbances. The roll and pitch angle correction response was then recorded and is represented by Figure 4.11. The test was repeated twenty times and the variance is illustrated by twenty blue graphs. The average of the twenty tests is represented by the red graph. The results of the average pitch and roll angle correction response can be seen in Table 4.10.



(a) Pitch angle correction response with position hold activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.



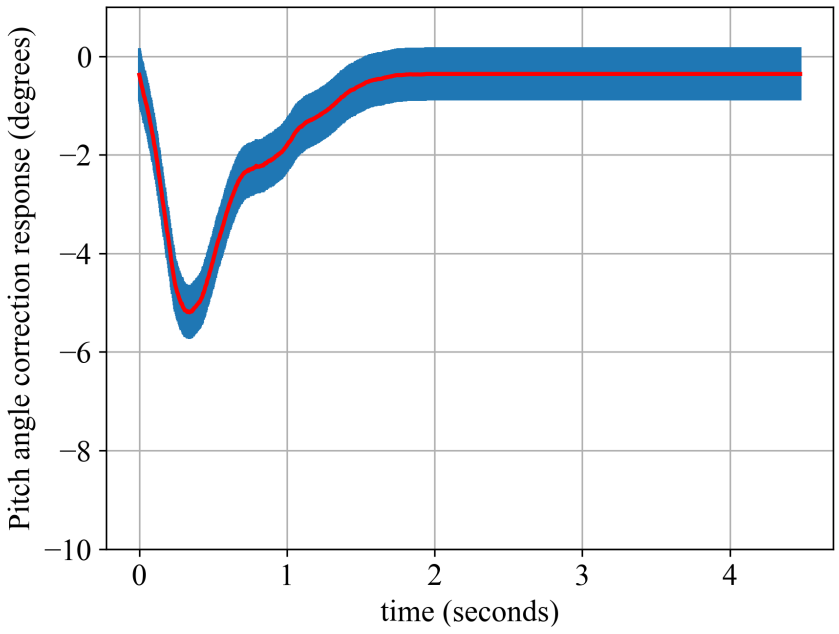
(b) Roll angle correction response with position hold activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Figure 4.11. Pitch and roll angle correction response with position mode activated.

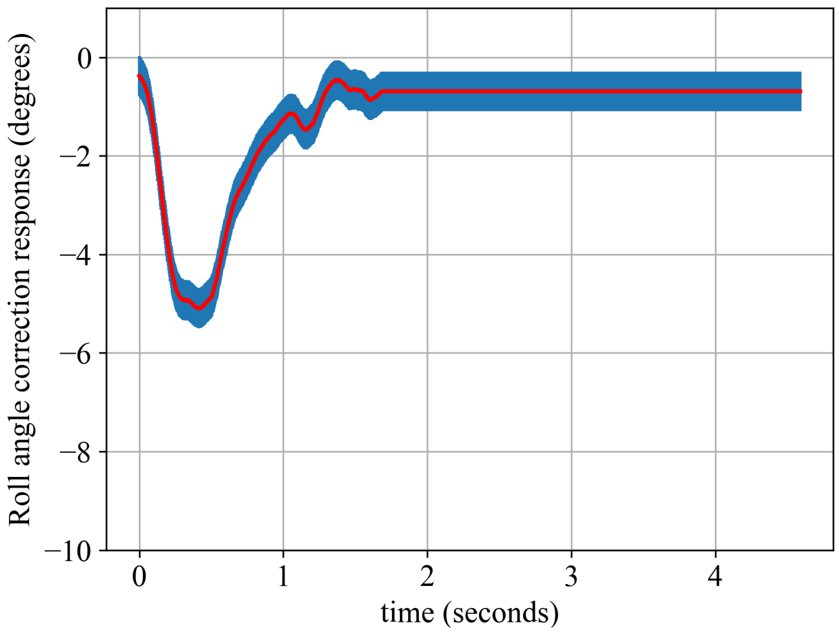
Table 4.10. Characteristics of position hold response.

	Error (°)	
	Min	**Max**
Pitch	-1.11	0.62
Roll	-1.2	0.78

Figure 4.12 represents the roll and pitch angle correction response for the position controller. The position hold controller was activated. The quadcopter was physically pulled away from the set position. The test was repeated twenty times and the variance is illustrated by twenty blue graphs. The average of the twenty tests is represented by the red graph. The results of the average pitch and roll angle correction response can be seen in Table 4.11.



(a) Pitch angle correction response with position hold activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.



(b) Roll angle correction response with position hold activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Figure 4.12. Pitch and roll angle correction response with position mode activated.

Table 4.11. Characteristics of position hold response.

	Degrees (°)
Peak pitch angle	-5.3
Peak roll angle	-5.9
Pitch angle error	(-1.3, 0)
Roll angle error	(-1.58, 0)

4.3.4.3 Altitude controller

The quadcopter was vertically transitioned and was instructed to hold the altitude. Figure 4.13 represents the output pressure of the altitude controller response when the quadcopter was set to hold the current altitude. The altitude hold test was repeated twenty times and the variance is represented by the blue graphs and the average of the twenty repeated tests is represented by the red graph. The results of the average altitude response can be seen in Table 4.12.

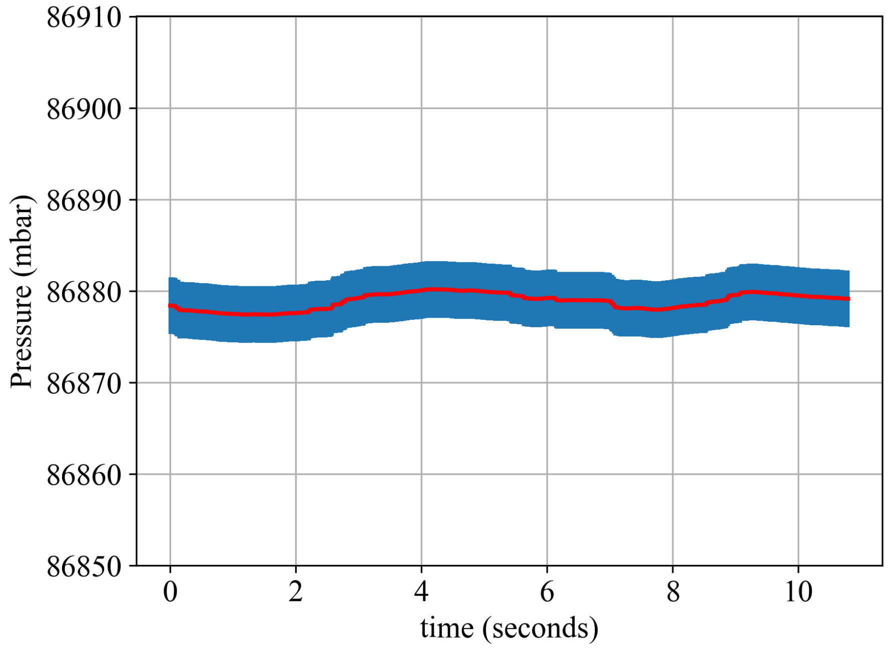


Figure 4.13. Pressure readings from the altitude controller. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.12. Characteristics of the altitude hold response.

Set altitude	86879.2 mbar
Altitude error	(-1.1, 0.5) mbar

In this test the quadcopter was vertically transitioned and was instructed to hold the altitude. Figure 4.14 represents the output pressure of the altitude controller response when the quadcopter was physically pushed away from the set altitude. This test was repeated twenty times and the variance is represented by the blue graphs and the average of the twenty repeated tests is represented by the red graph. The results of the average altitude response can be seen in Table 4.13.

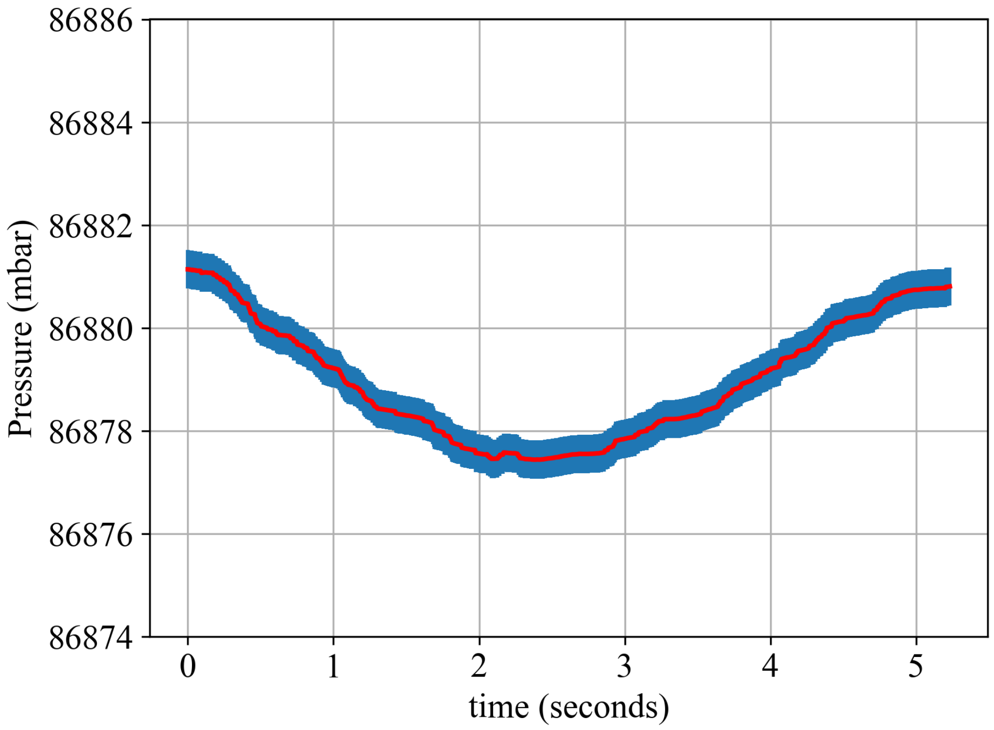


Figure 4.14. Pressure readings from the altitude controller. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.13. Characteristics of the altitude hold response.

Set altitude	86881.52 mbar
Settling time	5s
Altitude error	0.61 mbar

4.3.5 Three-blade propellers

The stability controller proportional gain (P) was set to 1.5, the integral gain (I) was set to 0.01 and the derivative gain (D) was set to 8.0. The position controller proportional gain (P) was set to 2.9, the integral gain (I) was set to 0.00 and the derivative gain (D) was set to 3.0. The altitude controller proportional gain (P) was set to 1.5, the integral gain (I) was set to 0.12 and the derivative gain (D) was set to 0.5. Table 4.14 contains a list of the PID gains being used for the stability, position and altitude controllers.

Table 4.14. PID gains for the three-blade propellers.

Three-blade propellers			
	Stability controller	**Position controller**	**Altitude controller**
Proportional gain (P)	1.5	2.9	1.5
Integral gain (I)	0.01	0.00	0.12
Derivative gain (D)	8.0	3.0	0.5

4.3.5.1 Stability controller

Figure 4.15 shows the pitch angle correction response of the quadcopter. In this tests, the quadcopter was disturbed by a negative pitch angle. The test was repeated twenty times and the variance is illustrated by twenty blue graphs. The average of the twenty tests is represented by the red graph. The results of the average forward pitch angle correction response can be seen in Table 4.15.

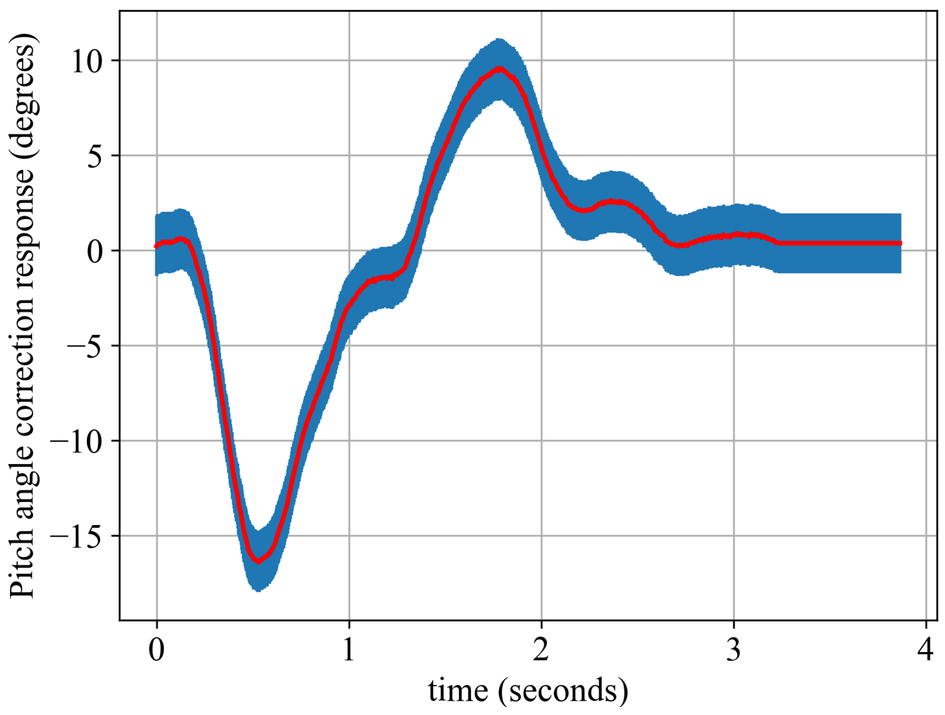


Figure 4.15. Pitch forward correction angle response without position mode activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.15. Characteristics of pitch forward correction angle response without position mode activated graph.

Peak angle (°)	8.2
Peak time (s)	1.81
Settling time (s)	3.1
Rise time (s)	1.1
Settling angle (°)	0.21

Figure 4.16 shows the pitch angle correction response of the quadcopter. In this tests, the quadcopter was disturbed by a positive pitch angle. The test was repeated twenty times and the variance is illustrated by twenty blue graphs. The average of the twenty tests is represented by the red graph. The results of the average backward pitch angle correction response can be seen in Table 4.16.

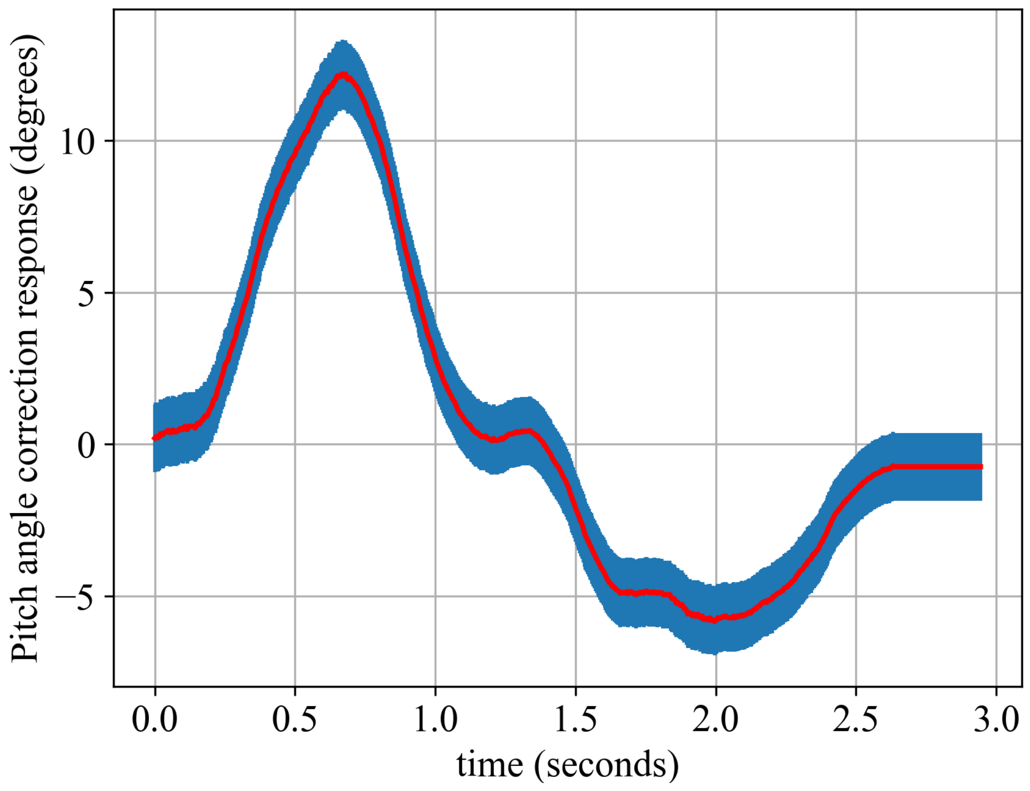


Figure 4.16. Pitch backward correction angle response without position mode activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.16. Characteristics of pitch backward correction angle response without position mode activated graph.

Peak angle (°)	-5.5
Peak time (s)	2.0
Settling time (s)	2.7
Rise time (s)	0.95
Settling angle (°)	-0.59

Figure 4.17 shows the roll angle correction response of the quadcopter. In this tests, the quadcopter was disturbed by a negative roll angle. The test was repeated twenty times and the variance is illustrated

by twenty blue graphs. The average of the twenty tests is represented by the red graph. The results of the average left roll angle correction response can be seen in Table 4.17.

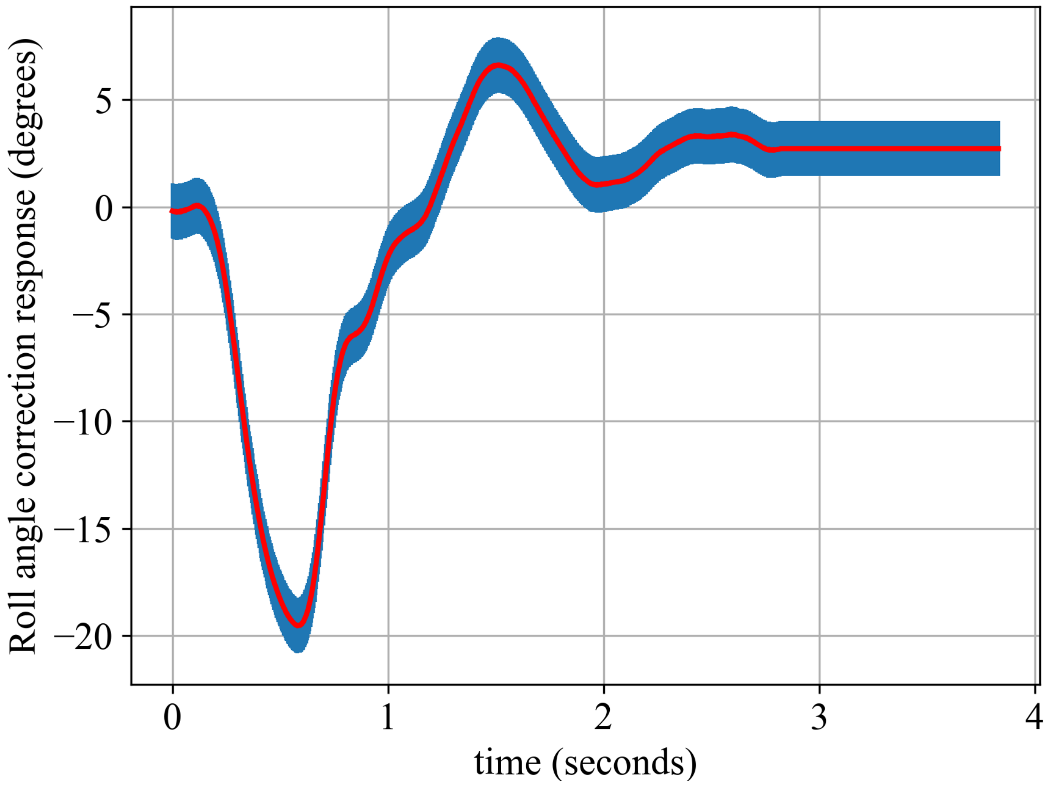


Figure 4.17. Roll left correction angle response without position mode activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.17. Characteristics of roll left correction angle response without position mode activated graph.

Peak angle (°)	6.1
Peak time (s)	1.4
Settling time (s)	2.9
Rise time (s)	1.1
Settling angle (°)	1.3

Figure 4.18 shows the roll angle correction response of the quadcopter. In this tests, the quadcopter was disturbed by a positive roll angle. The test was repeated twenty times and the variance is illustrated by twenty blue graphs. The average of the twenty tests is represented by the red graph. The results of the average right roll angle correction response can be seen in Table 4.18.

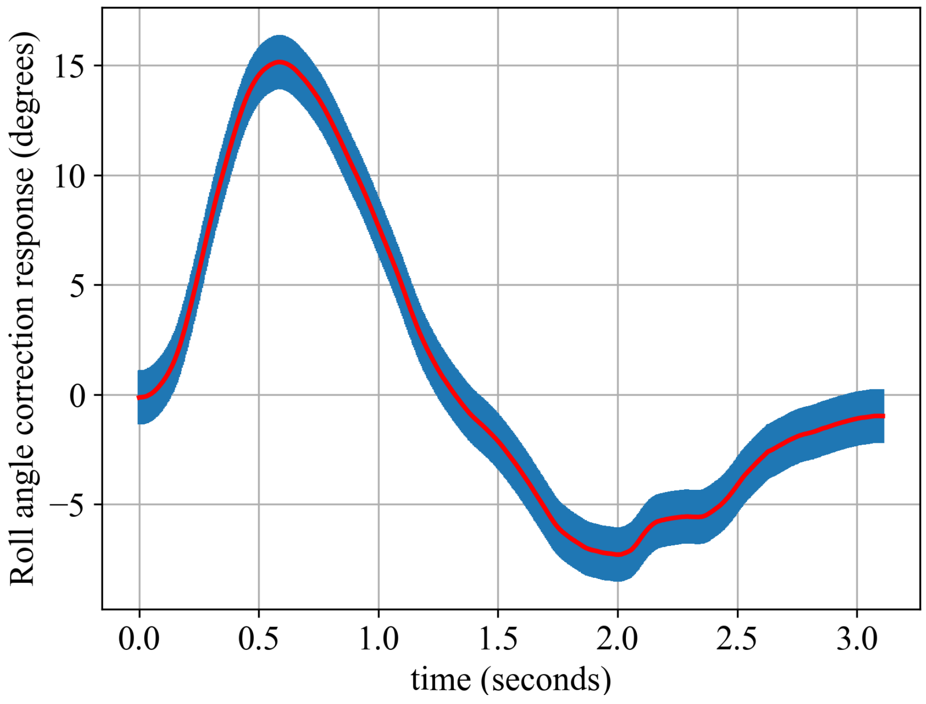


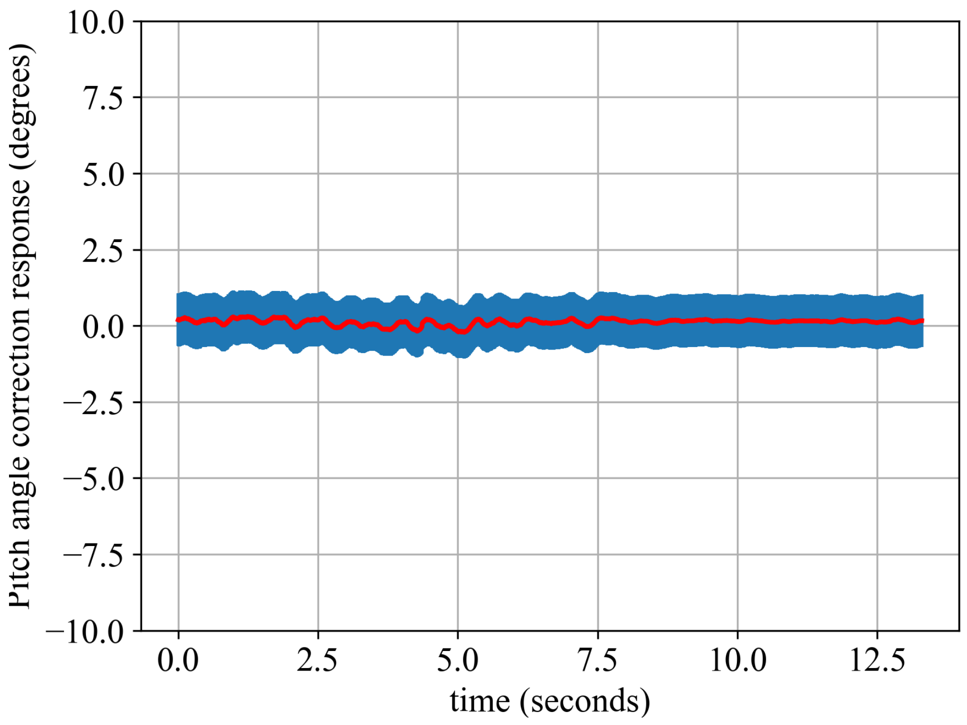
Figure 4.18. Roll right correction angle response without position mode activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.18. Characteristics of roll right correction angle response without position mode activated graph.

Peak angle (°)	-7.5
Peak time (s)	2.0
Settling time (s)	3.05
Rise time (s)	0.95
Settling angle (°)	-1.0

4.3.5.2 Position controller

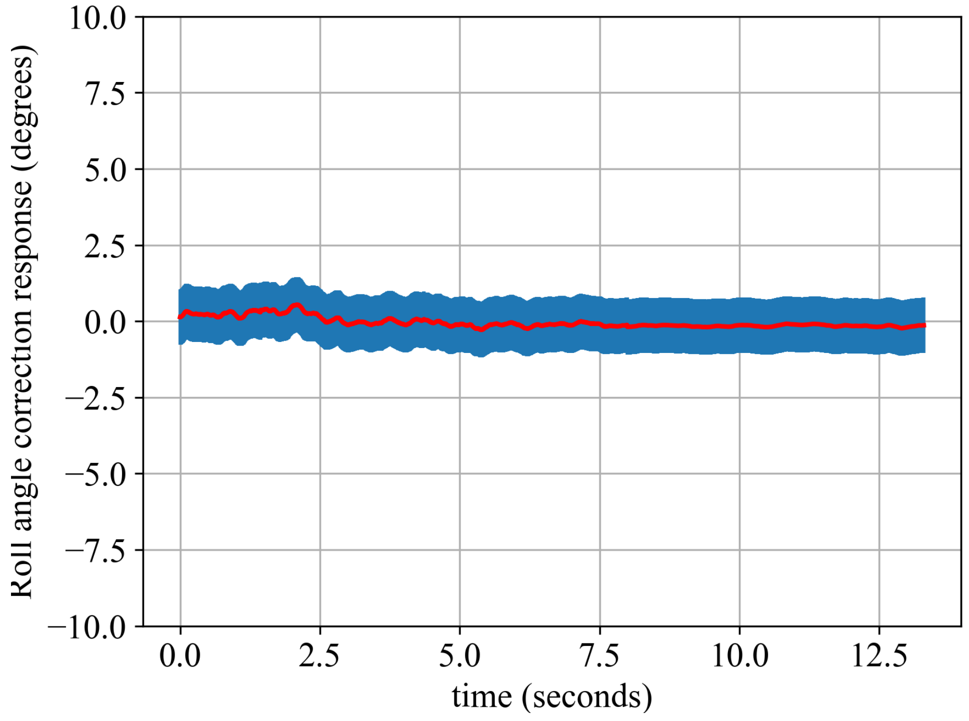
Figure 4.19 represents the roll and pitch angle correction response for the position controller. The test was repeated twenty times and the variance is illustrated by twenty blue graphs. The average of the twenty tests is represented by the red graph. The results of the average pitch and roll angle correction response can be seen in Table 4.19.



(a) Pitch angle correction response with position hold activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.19. Characteristics of position hold response.

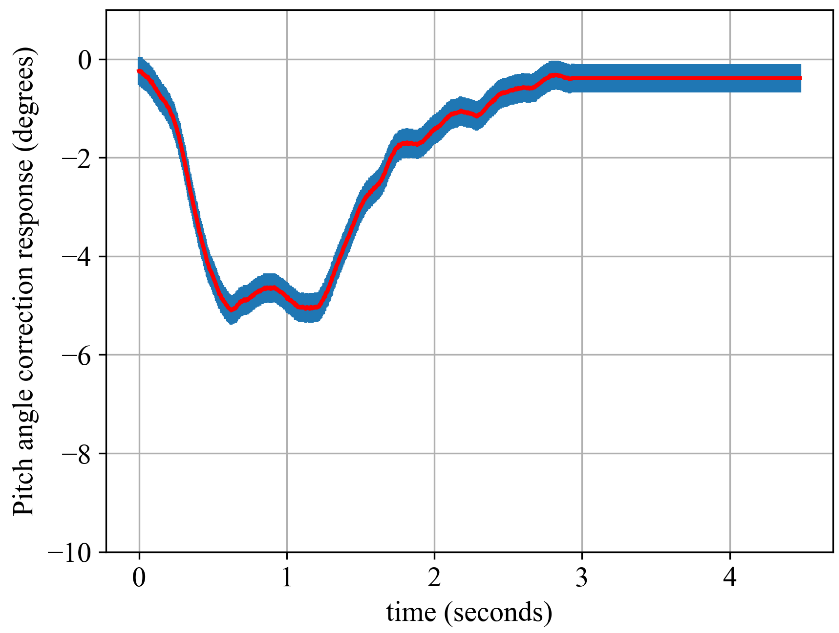
	Error (°)	
	Min	**Max**
Pitch	-0152	0.291
Roll	-0.35	0.44



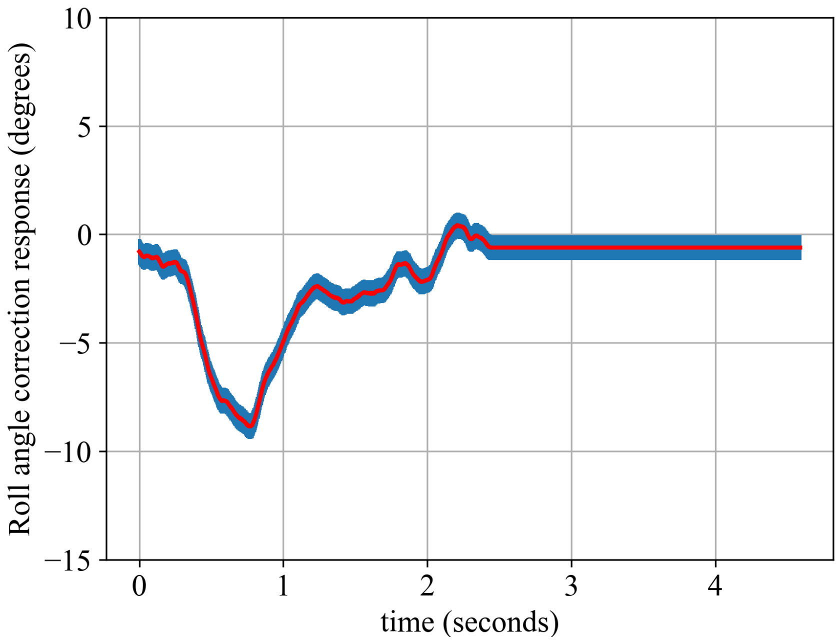
(b) Roll angle correction response with position hold activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Figure 4.19. Pitch and roll angle correction response with position mode activated.

Figure 4.20 represents the roll and pitch angle correction response for the position controller. The position hold controller was activated. The quadcopter was physically pulled away from the set position. The test was repeated twenty times and the variance is illustrated by twenty blue graphs. The average of the twenty tests is represented by the red graph. The results of the average pitch and roll angle correction response can be seen in Table 4.20.



(a) Pitch angle correction response with position hold activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.



(b) Roll angle correction response with position hold activated. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Figure 4.20. Pitch and roll angle correction response with position mode activated.

The important results are tabulated in Table 4.20.

Table 4.20. Characteristics of position hold response.

	Degrees (°)
Peak pitch angle	-5.5
Peak roll angle	-9.1
Pitch angle error	(-0.35, 0.1)
Roll angle error	(-0.2, 0.15)

4.3.5.3 Altitude controller

The quadcopter was vertically transitioned and was instructed to hold the altitude. Figure 4.21 represents the output pressure of the altitude controller response when the quadcopter was set to hold the current altitude. The altitude hold test was repeated twenty times and the variance is represented by the blue graphs and the average of the twenty repeated tests is represented by the red graph. The results of the average altitude response can be seen in Table 4.21.

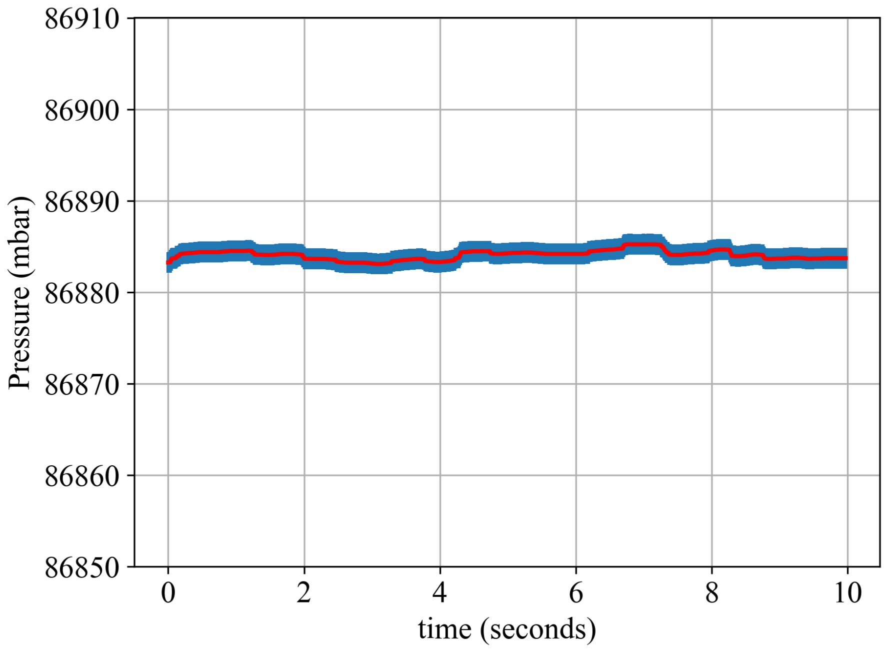


Figure 4.21. Pressure readings from the Altitude controller. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

Table 4.21. Characteristics of the altitude hold response.

Set altitude	86885 mbar
Altitude error	(-0.5, 0.3) mbar

In this test the quadcopter was vertically transitioned and was instructed to hold the altitude. Figure 4.22 represents the output pressure of the altitude controller response when the quadcopter was physically pushed away from the set altitude. This test was repeated twenty times and the variance is represented by the blue graphs and the average of the twenty repeated tests is represented by the red graph. The results of the average altitude response can be seen in Table 4.22.

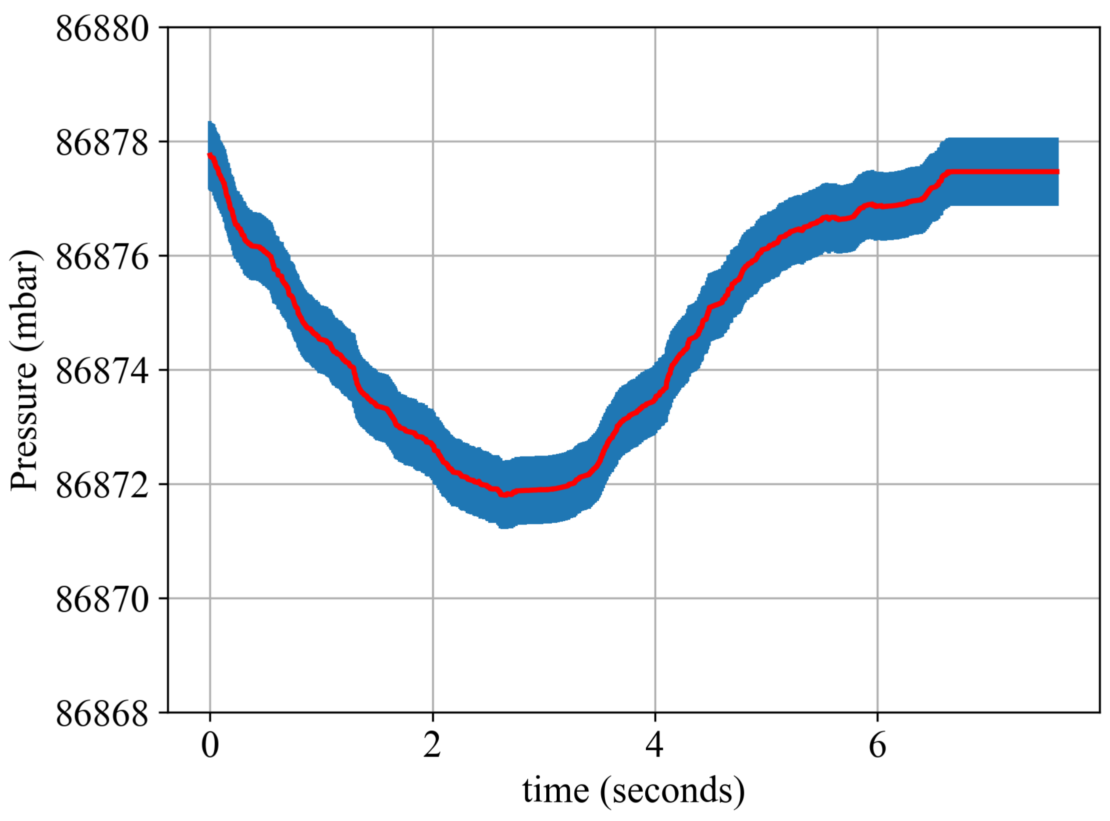


Figure 4.22. Pressure readings from the altitude controller. The blue graphs show the variance of the twenty tests and the red graph shows the average of the twenty tests.

The important results are tabulated in Table 4.22.

Table 4.22. Characteristics of the altitude hold response.

Set altitude	86878.14 mbar
Settling time	6.85
Altitude error	0.40 mbar

4.3.6 Position estimation

Figure 4.23 shows the results of the position estimation experiment. The quadcopter's point of reference was set at position (1, 1, 1). The quadcopter was set to move 3 m forward and increase its altitude by 1 m. That was followed by 3 m to the right and decrease the quadcopter's altitude by 1 m. Lastly, the quadcopter was instructed to move 3 m backwards and decrease the quadcopter's altitude by 0.5 m. The actual position from the reference point is shown by using the green dot and were marked with landmarks before the experiment could commence. The grey dots show the position estimate across multiple tests and the coloured dots shows the average of the multiple tests. The blue dot represents the quadcopter's estimated position at position (1, 1, 1). The red dot represents the quadcopter's estimated position at position (1, 4, 2). The yellow dot represents the quadcopter's estimated position at position (4, 4, 0.5). The purple dot represents the quadcopters estimated position at position (4, 1, 1.5).

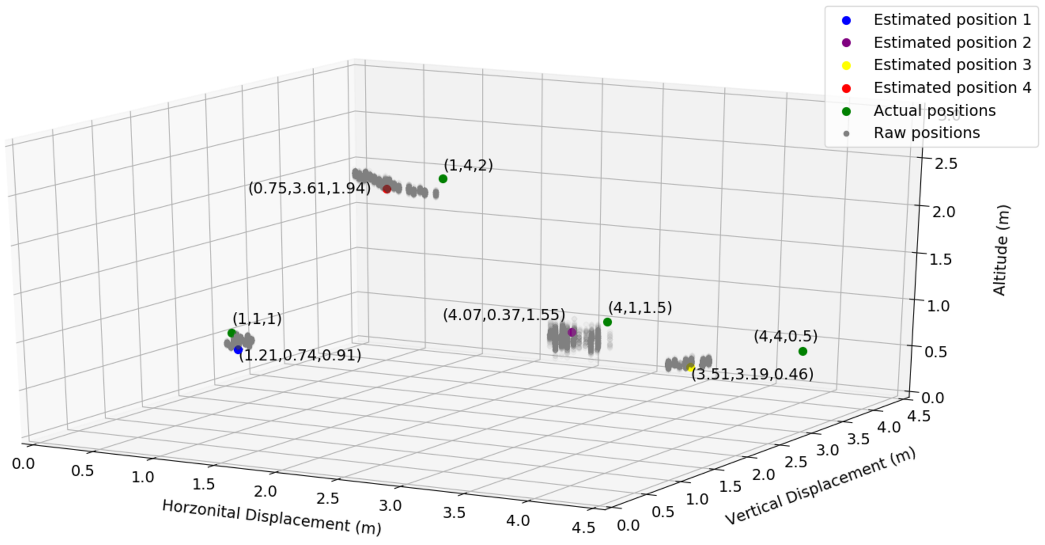


Figure 4.23. Position estimation. Grey dots represent the hundred tests and the coloured dots represent the average of the hundred dots.

The Euclidean error between the actual error and the estimated error was recorded in Table 4.23. The Euclidean error was calculated by

$$e = \sqrt{(x - x_{estimation})^2 + (y - y_{estimation})^2 + (z - z_{estimation})^2}, \tag{4.1}$$

where e represents the Euclidean error. The average error was calculated to be 0.54 m from the actual position.

Table 4.23. Average Euclidean error of the position estimation.

Actual position (x,y)	**Average estimate position (x,y)**	**Average Euclidean error (m)**
(1, 1, 1)	(1.21, 0.74, 0.91)	0.34
(1, 4, 2)	(0.75, 3.61, 1.94)	0.47
(4, 4, 0.5)	(3.51, 3.19, 0.46)	0.94
(4, 1, 1.5)	(4.07, 0.37, 1.55)	0.63

Figure 4.24 shows the position estimation error over twelve minutes as the quadcopter was moving to various positions. During the first 2.5 minutes (purple oval) the quadcopter had an error between 0.1 m and 0.6 m. During the next 5 minutes (red oval) the quadcopter had an error between 0.6 m and 1.1 m. Finally, during the last 3.5 minutes (green oval) the quadcopter had an error between 0.4 m and 0.7 m.

It was noticed that the quadcopter experienced a higher error when it only received six GPS satellites, as shown in the red oval. However, when the quadcopter received more than eight satellites, its position estimation error reduced, as shown in the purple and green ovals.

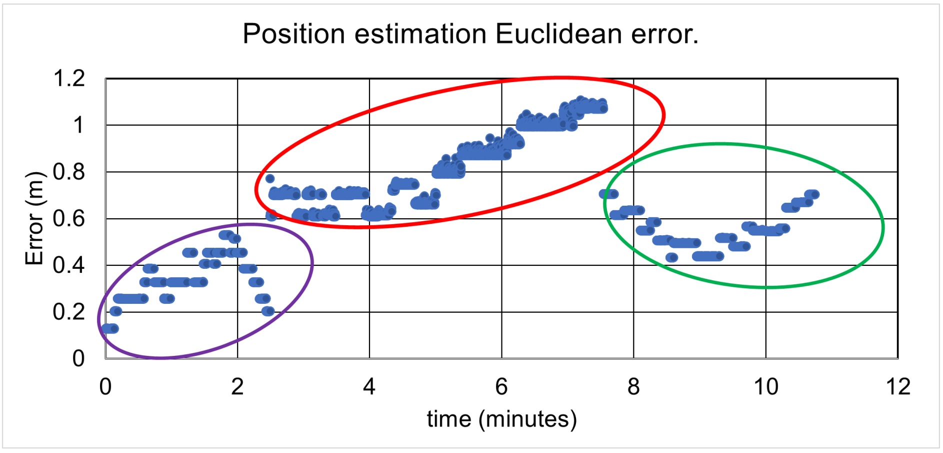


Figure 4.24. Position estimation Euclidean error over time.

Figure 4.25 shows the relationship between the standard deviation and the number of satellites being used for the position estimation algorithm. It can be seen that the standard deviation starts around 2.5 m when only 8 satellites are used and reduces to 0.5 m when 12 satellites are used.

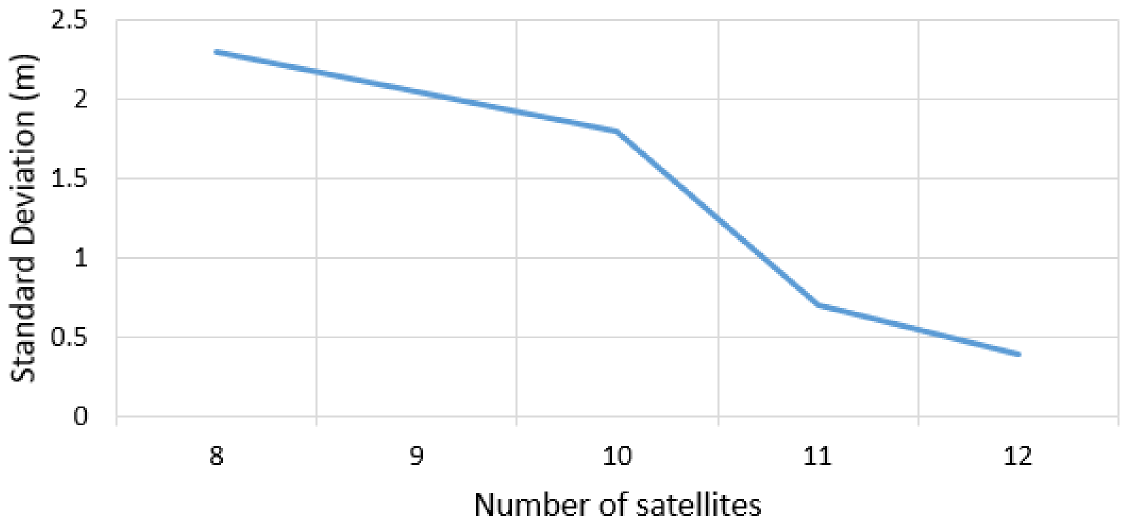


Figure 4.25. Standard deviation vs. number of satellites.

A video of a waypoint test with the quadcopter can be seen at this link.

4.3.7 Rotational estimation

To test the rotation ability of the quadcopter, it was instructed to perform multiple rotational tests when facing 0°. Once the quadcopter had completed the required rotation test, the quadcopter was landed and the change in angle was measured. The rotational estimation test included 30°, 90°, -45°, -90° and 180° tests and measurements. Each test was repeated twenty times and an average estimated rotation angle was calculated. Table 4.24 shows the results found in the rotational estimation test.

Table 4.24. Rotational angle estimation test results.

Instructed rotation (°)	**Average estimated rotation angle** (°)	**Average rotation angle** (°)
30	33	3
90	94	4
-45	-49	4
-90	-88	2
180	177	3
	Total estimated error (°)	3.2

4.3.8 Propeller and controller comparison

4.3.8.1 Stability controller

When comparing the PID gains in Table 4.5 and Table 4.14, it was noticed that the three-blade propellers require lower gain settings compared to the two-blade propellers. The lower gains conform to the literature, as the three-blade propellers have three points of contact with the air, whereas the two-blade propellers only have two points of contact with the air. The extra point of contact helps to improve the balance of the quadcopter and ultimately results in only a smaller correction required to keep the quadcopter stable in the air. It was noticed that the three-blade propellers experienced a lower overall peak angle; however, the rise time was longer for the three-blade propellers. The percentage overshoot of the three-blade propellers was found to be lower, compared to the two-blade propellers. This conforms with the literature, as the three-blade propellers required less correction for the quadcopter to be stable; however, the response time was longer as opposed to the two-blade propellers.

From previous papers, [11] and [56] achieved an equal roll and pitch error of approximately 0° to 3° error and ± 4°, respectively. The stability controller with the two-blade propellers used in this project, achieved a maximum error of between -1.58° and 0.78°. The stability controller with the three-blade propellers used in this project achieved a maximum error of between -0.35° and 0.44°. It can be noted regarding the stability controller with the two-blade propellers, the error was similar to [11] and it provided a better accuracy compared to the stability controller in [56]. It was also seen that the stability controller performed better with the three-blade propellers than with the two-blade propellers. The settling time for the quadcopter built with the two-blade propellers was slightly faster to the settling time found in [57]. However, the settling time using the three-blade propellers was approximately 3 s, which is slightly slower compared to [57]. This was expected, since the three-blade propellers have a slower response time, as opposed to the two-blade propellers.

4.3.8.2 Position controller

The comparison between the position controllers shows that the two-blade position controller performed reasonably well. However, the controller had indeed made various corrections for the quadcopter to maintain the desired position. The three-blade position controller performed better than the two-blade position controller, as the error in the roll and pitch angle was considerably lower than the two-blade roll and pitch angle error. It can be seen that both the two-blade position controller and the three-blade position controller were able to move back to the desired position when the quadcopter was physically moved away from it's desired position. The three-blade propeller performed better than the two-blade propeller, as the error in the roll and pitch angle were considerably lower than the two-blade roll and pitch angle error. The results confirm that the three-blade propellers provide a better position controller.

In comparison to results presented in [11], it can be seen that the quadcopter experienced an overall variation in the horizontal and vertical direction between 0.3048 m and 0.6096 m. This result is an Euclidean error of approximately 0.43 m to 1.54 m. The results from the quadcopter presented herein have a slightly better position estimation in some cases, compared to [11], with an Euclidean error between 0.39 m and 0.83 m. It was found that the yaw average error by the developed system was 3.2°, similar to the average error of 3.4° found in [72].

4.3.8.3 Altitude controller

The altitude controllers for both the two-blade propellers and the three-blade propellers performed similarly. The three-blade propellers had slightly better results than the two-blade propellers. The two-blade propellers did, however, have a faster response time, as opposed to the three-blade propellers.

The findings also proved that by using the combination of the stability, position and altitude controller it was possible to implement a quadcopter that can be used for real-time image processing applications. The two-blade propellers achieved a settling time of 5 s and the three-blade propellers achieved a settling time of 6.85 s. The two-blade propellers had a faster settling time than the results that had been achieved in research paper [57] with a settling time of 6.34 s; however, the three-blade propellers was closer to the results found in research paper [57].

The stability, position and altitude controller performance can be summarised in the two main points: The three-blade propeller has a longer response time and it also has a lower percentage overshoot.

By evaluating the results, it can be noted that the three-blade propeller, in combination with well-tuned controllers, was the better choice for the overall performance, in comparison to a two-blade propeller system. The three-blade propeller system was better suited for a real-time image processing application, using a quadcopter. The findings show that the more the points of contact with the air, the better the overall performance of the quadcopter.

4.4 IMAGE PROCESSING

As previously mentioned, the real-time image processing was not implemented from first principles. The model available from the official YOLO website [73] was used. The YOLOv2 network was processed on a laptop with an Nvidia GTX 1050 4GB GPU. The YOLOv2 algorithm could successfully detect items in the image, such as chairs and people. The YOLOv2 algorithm could process an image within 0.7 s and the Caddx Ratel camera had a latwentycy of 8 ms. Therefore, the overall time to capture an image, send the image to the base station and process the image was 0.7008 s.

4.4.1 Two-blade propellers

Figure 4.26(a) - (f) represent images from the quadcopter after the images had been processed by the YOLOv2 algorithm. The altitude hold and position hold controller were activated in Figure 4.26(a). Figure 4.26(b) represents the image from the quadcopter when the quadcopter was instructed to move to a new position. Figure 4.26(c) - (e) show the quadcopter moving to it's new position and stabilising itself to maintain the new desired position. Figure 4.26(f) represents the quadcopter when it has settled in the new desired position. It was also noticed in Figure 4.26(a) - (f), that the Caddx Ratel camera did experience very little of the jello effect.

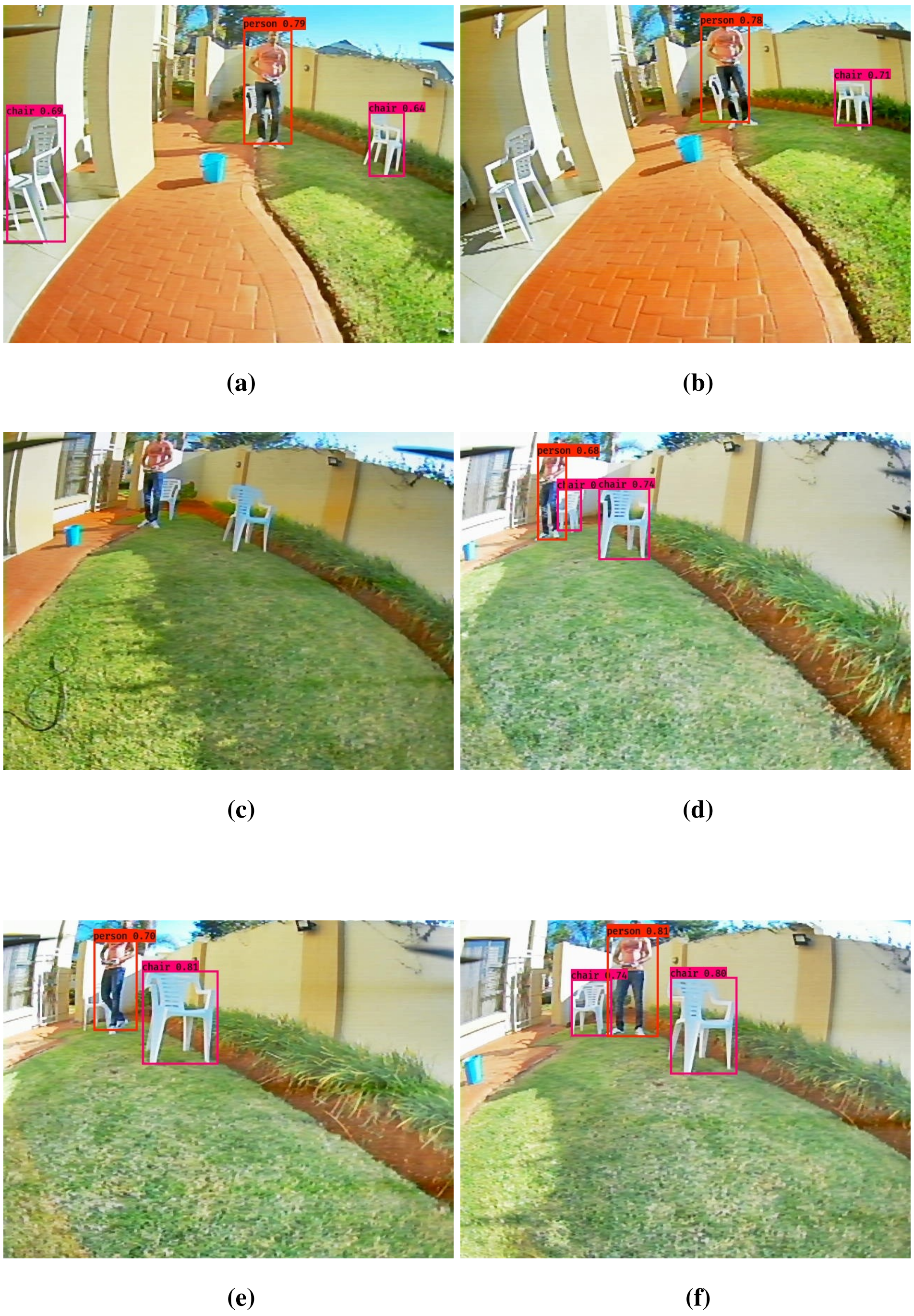
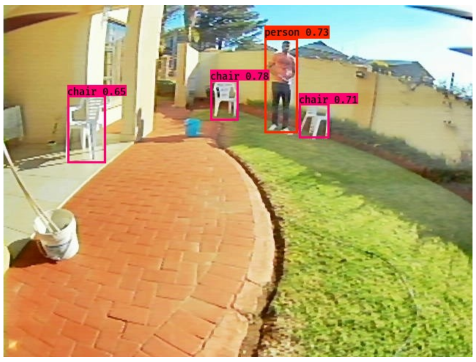


Figure 4.26. Image processing on images from the Caddx Ratel camera, using two-blade propellers.

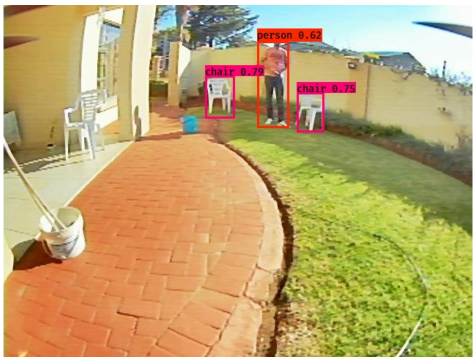
4.4.2 Three-blade propellers

Figure 4.27(a) - (e) represent images from the quadcopter after the images have been processed by the YOLOv2 algorithm. The altitude hold and position hold controller were activated in Figure 4.27(a).

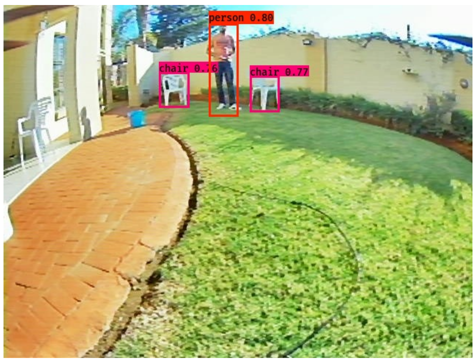
Figure 4.27(b) represents the image from the quadcopter when the quadcopter was instructed to move to a new position. Figure 4.27(c) and Figure 4.27(d) show the quadcopter moving to the new position and stabilising itself to maintain the new desired position. Figure 4.27(e) represents the quadcopter when it has settled in the new desired position. It was also noticed in Figure 4.27(a) - (e), that the Caddx Ratel camera did experienced very little of the jello effect.



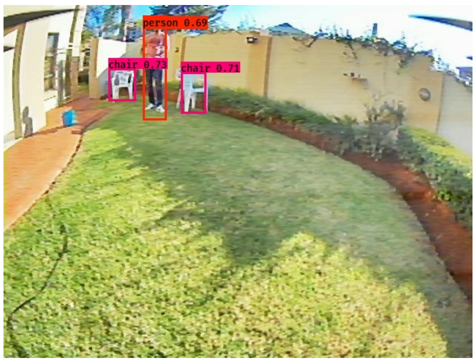
(a)



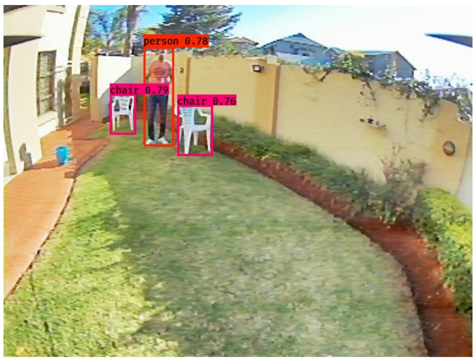
(b)



(c)



(d)



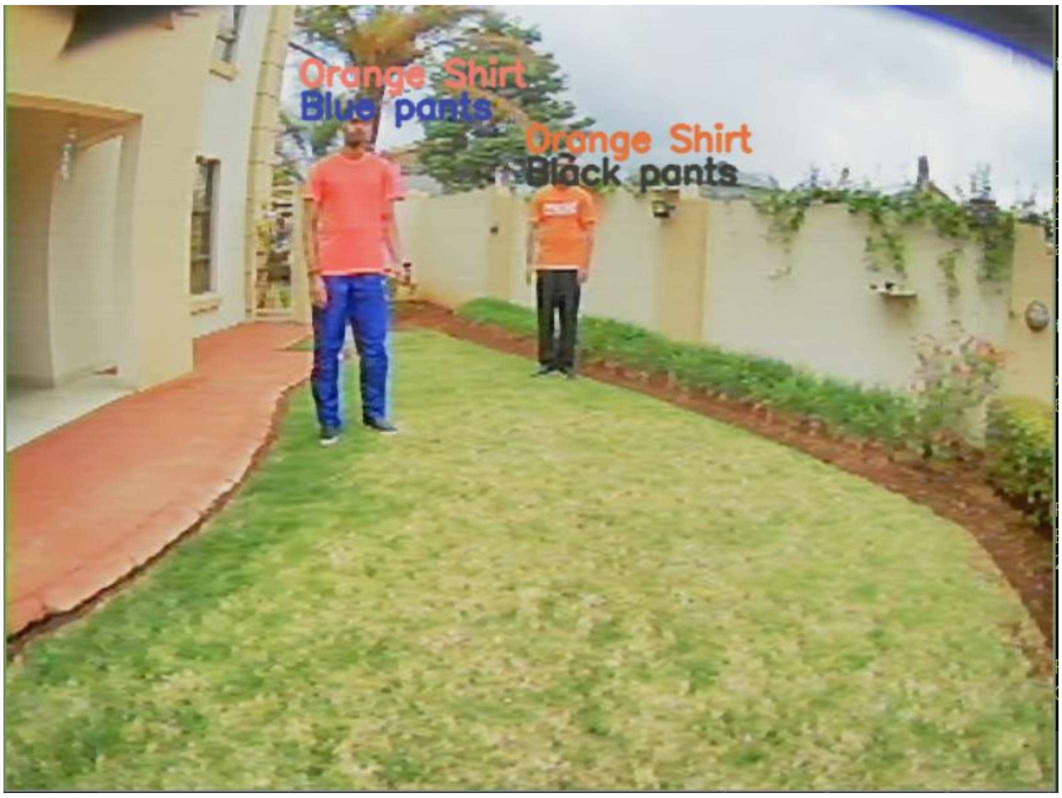
(e)

Figure 4.27. Image processing on images from the Caddx Ratel camera, using three-blade propellers.

It can also be seen that the images being captured while using the three-blade propeller, are more crisp compared to the two-blade propeller images. The three-blade propellers provide a more stable flight, resulting in a better quality of images for the YOLOv2 algorithm.

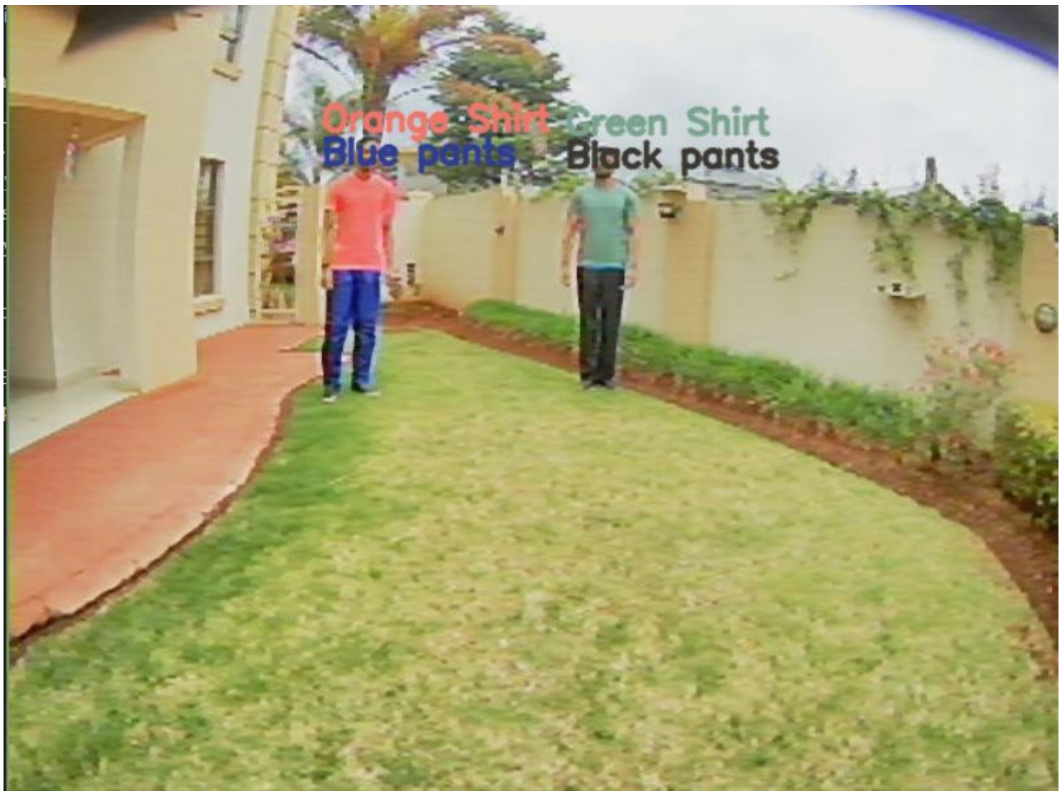
4.4.3 Colour classification test

This test presents the colour classification results. The colours associated with the persons' shirts and pants can be seen above the detected person. Furthermore, the font colour used, was the same colour as the colour classified. Figure 4.28(a) - (e) show the results from the images being captured from the Caddx Ratel camera and processed on the base station.

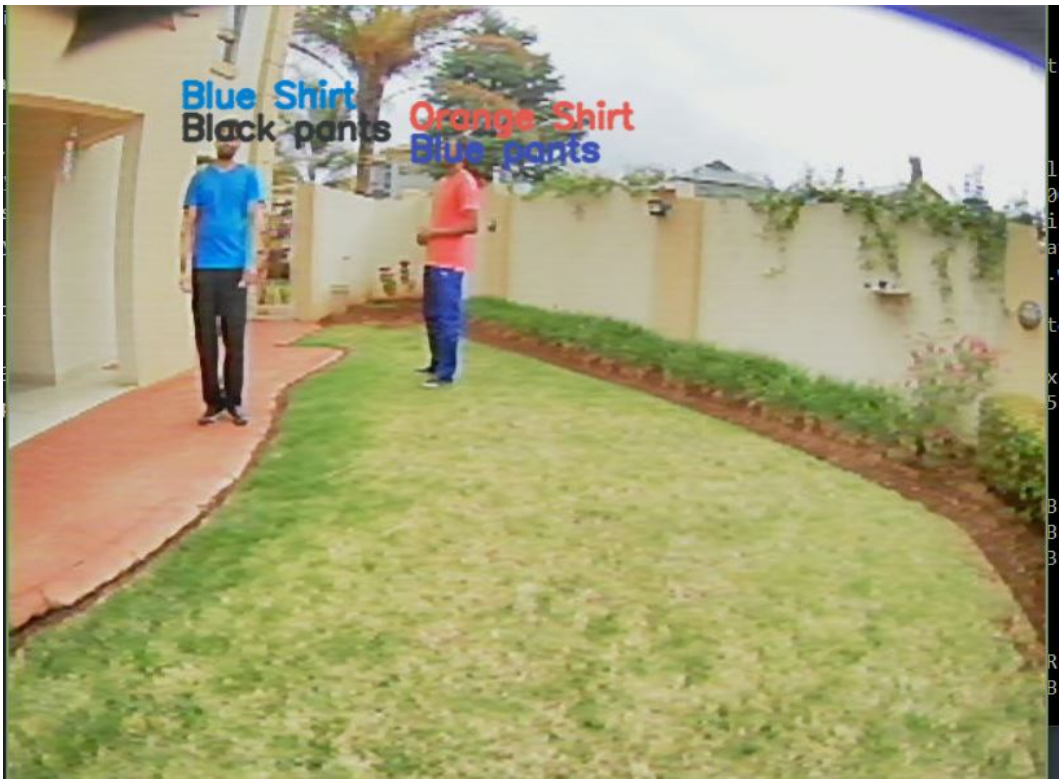


(a)

It can be seen that the colour classification algorithm was able to classify the colour of the shirts and pants correctly.



(b)

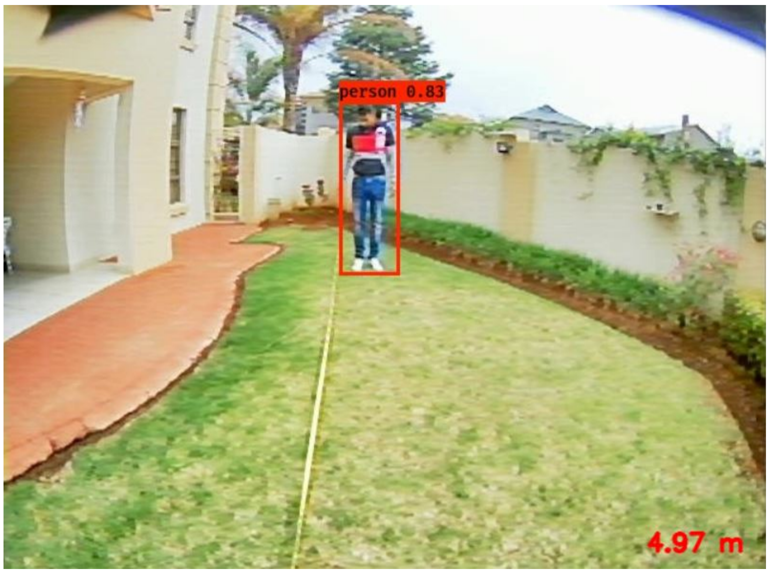


(c)

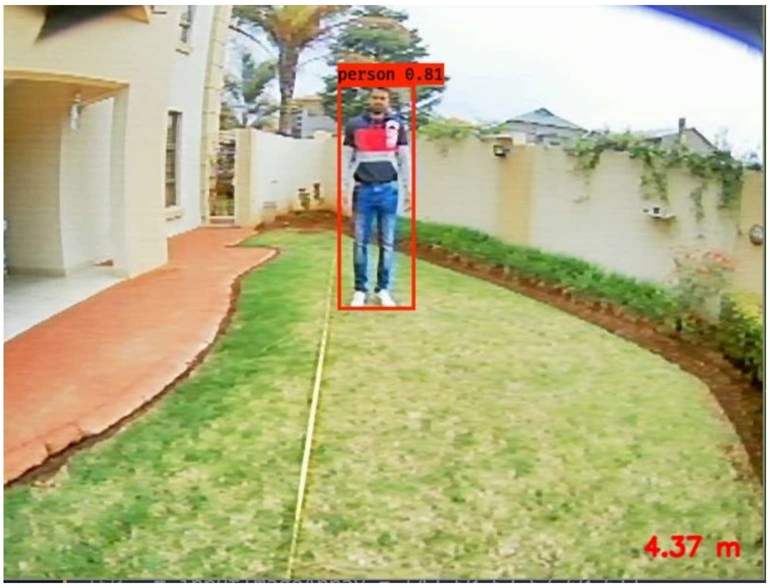
Figure 4.28. Colour detection results.

4.4.4 Distance estimation test

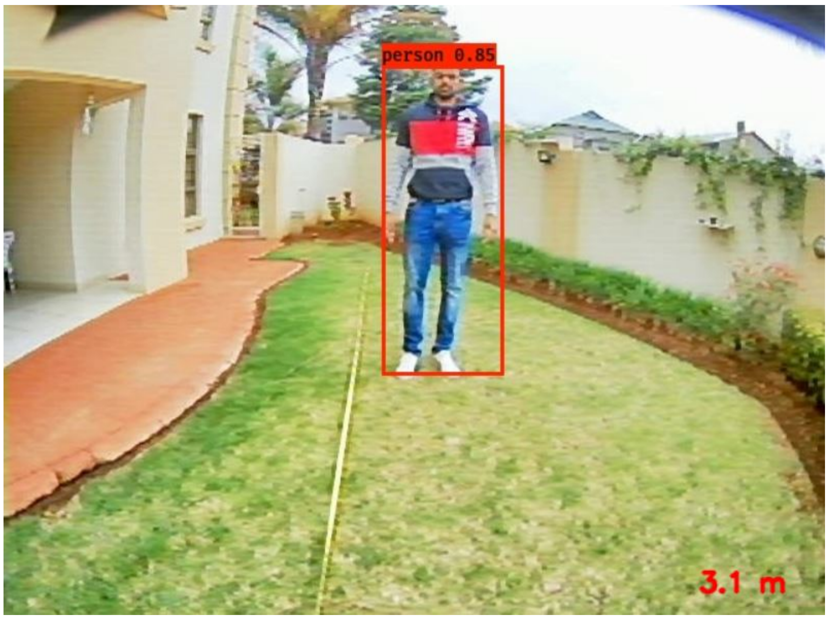
This test presents the distance estimation results. The distance test was conducted at 3 m, 4 m and 5 m from the quadcopter. Figure 4.29(a) - (c) show the results when the person was facing the quadcopter at a distance of 5 m, 4 m and 3 m, respectively. Figure 4.30(a) show the result when the person was not facing the quadcopter and Figure 4.30(b) show the result when a person stretches out his/her hands.



(a) Estimated distance = 4.97 m, Actual distance = 5 m.

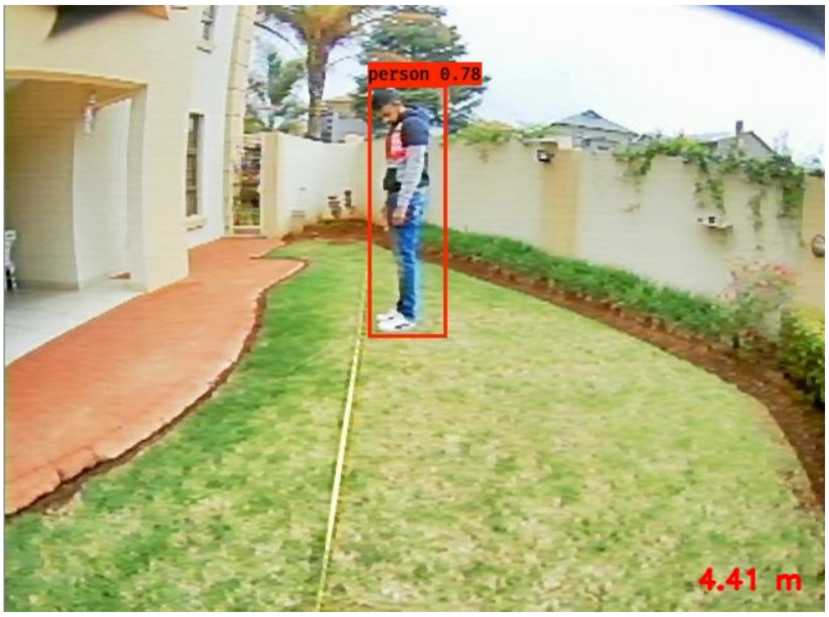


(b) Estimated distance = 4.37 m, Actual distance = 4 m.

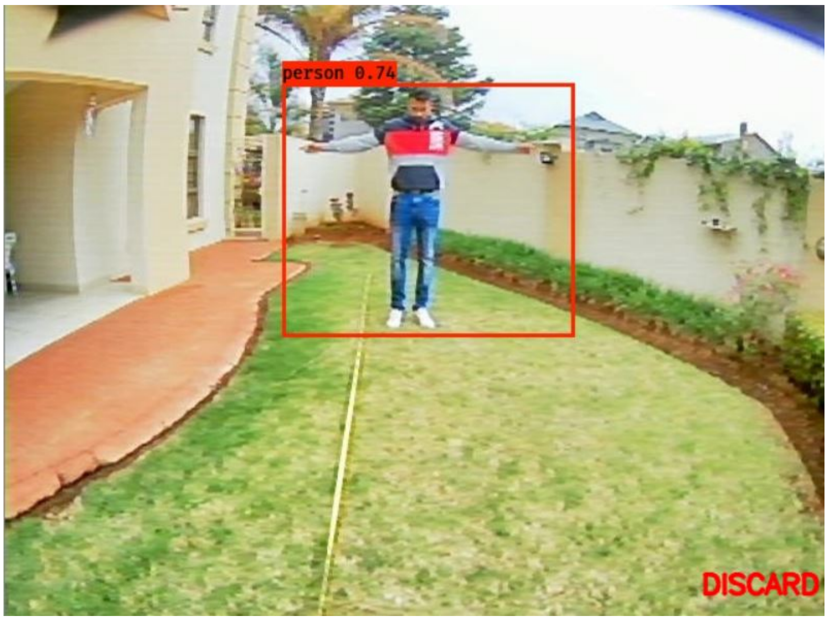


(c) Estimated distance = 3.1 m, Actual distance = 3 m.

Figure 4.29. Distance estimation results (Person's actual height = 1.75 m).



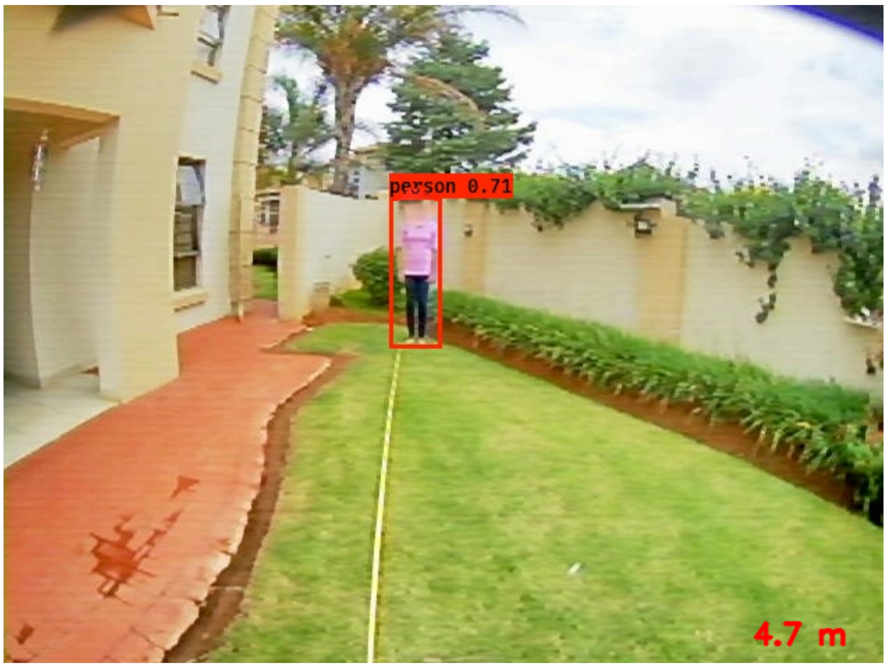
(a) Person not facing the quadcopter directly. Estimated distance = 4.41 m, Actual distance = 4 m.



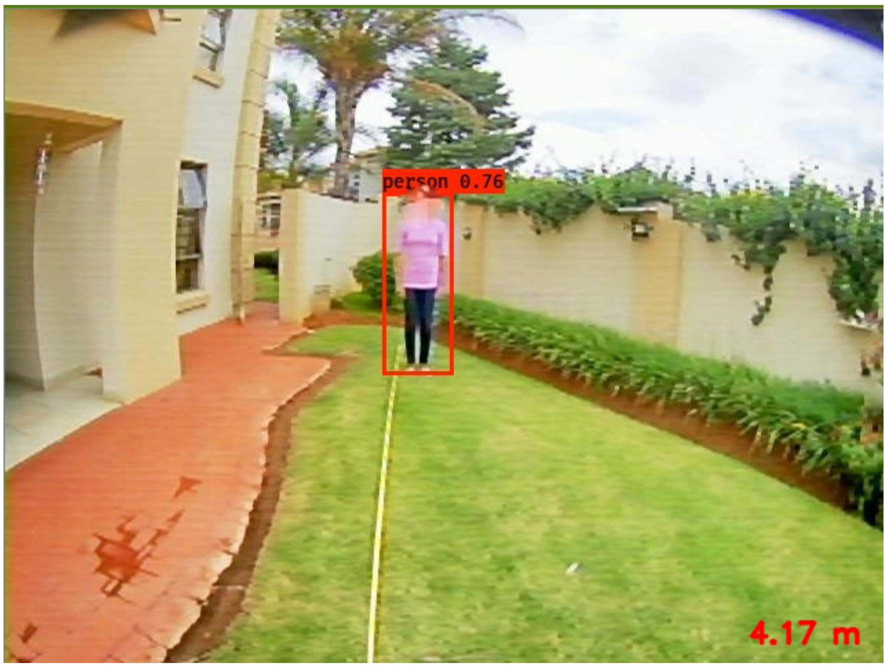
(b) Image discarded.

Figure 4.30. Possible distance estimation abnormalities (Person's actual height = 1.6 m).

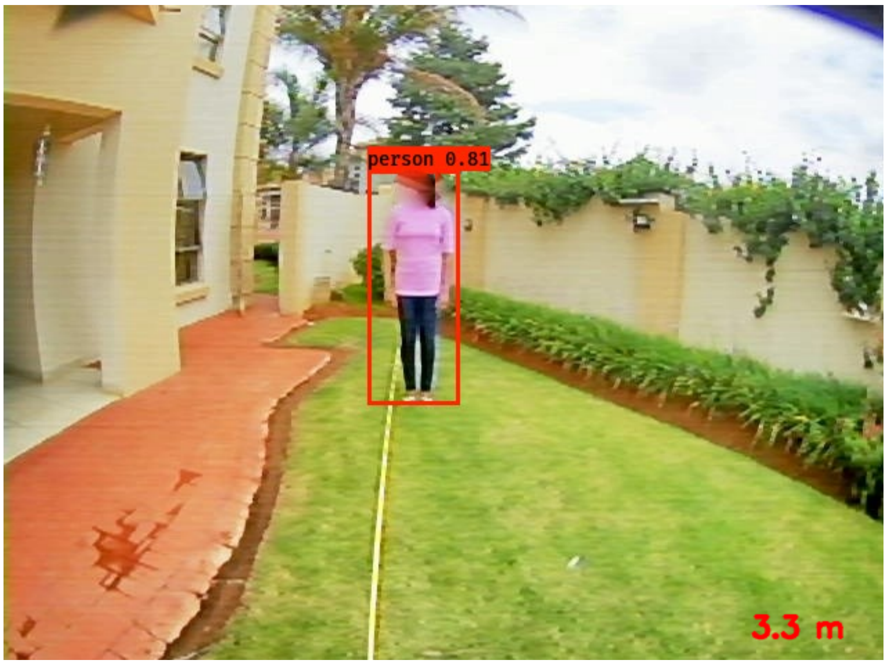
Figure 4.31(a) - (c) show a similar distance estimation test performed with a shorter person. Figure 4.32(a) - (b) show the distance abnormalities for a shorter person.



(a) Estimated distance = 4.7 m, Actual distance = 5 m.

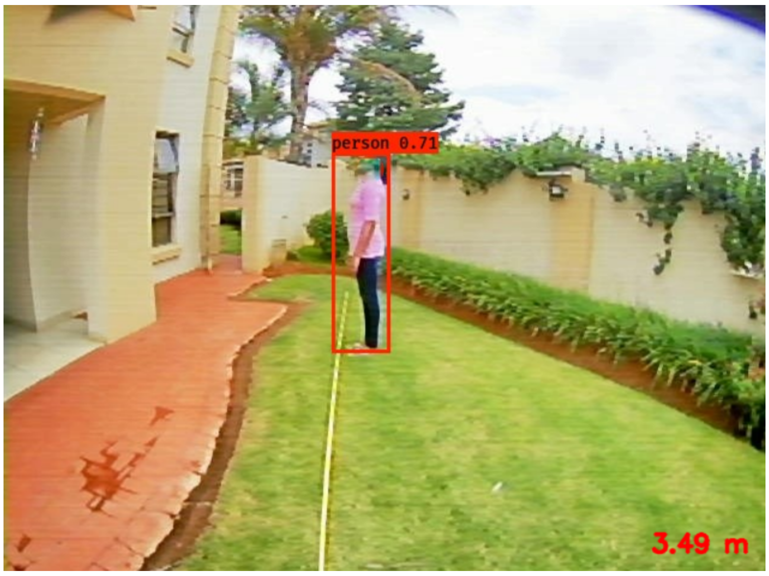


(b) Estimated distance = 4.17 m, Actual distance = 4 m.

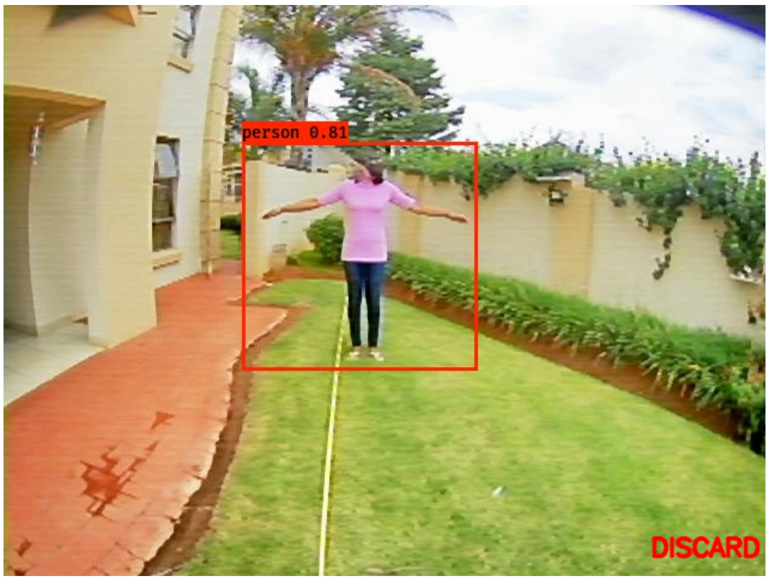


(c) Estimated distance = 3.3 m, Actual distance = 3 m.

Figure 4.31. Distance estimation results (Person's actual height = 1.6 m).



(a) Person not facing the quadcopter directly. Estimated distance = 3.49 m, Actual distance = 3 m.



(b) Image discarded.

Figure 4.32. Possible distance estimation abnormalities (Person's actual height = 1.6 m).

It was found that when the person was not directly facing the quadcopter the distance estimation algorithm introduced error. However, it provided a distance estimation still worth using. It could be seen that when a person would stretch out his/her arms, the algorithm successfully discarded the result. After performing twenty distance estimation tests with a shorter and taller person, the standard deviation was between 0.3 m and 0.33 m. Table 4.25 shows a summary of the twenty tests performed with a shorter and taller person.

Table 4.25. Summary of the twenty distance estimation tests.

Actual distance (m)	**5m**	**4m**	**3m**	**Average estimated error (m)**
Taller person (1.75m)	4.79	4.39	3.4	0.33
Shorter person (1.6m)	4.72	4.27	3.35	0.30

4.4.5 Image distortion test

This test was conducted to determine the quality of the images and to quantify the image distortion, using Python and OpenCV. The reference image was an image taken when the camera was kept still and the distorted image was an image taken during the flight from the quadcopter. These two images were compared and evaluated, using the MSE, PSNR and SSIM measurements. This test was repeated twenty times and was tabulated in Table 4.26.

Table 4.26. Image distortion quality test results.

Image quality assessment			
Test Nr.	**MSE (dB)**	**PSNR (dB)**	**SSIM**
1	874.85	18.71	0.744
2	2549.80	14.07	0.705
3	2004.00	15.11	0.731
4	1952.56	15.22	0.649
5	796.51	19.11	0.591
6	1222.57	17.26	0.802
7	1272.27	17.09	0.799
8	1035.20	17.98	0.807
9	2480.81	14.18	0.735
10	407.17	22.03	0.711
Average	1465.12	17.08	0.727

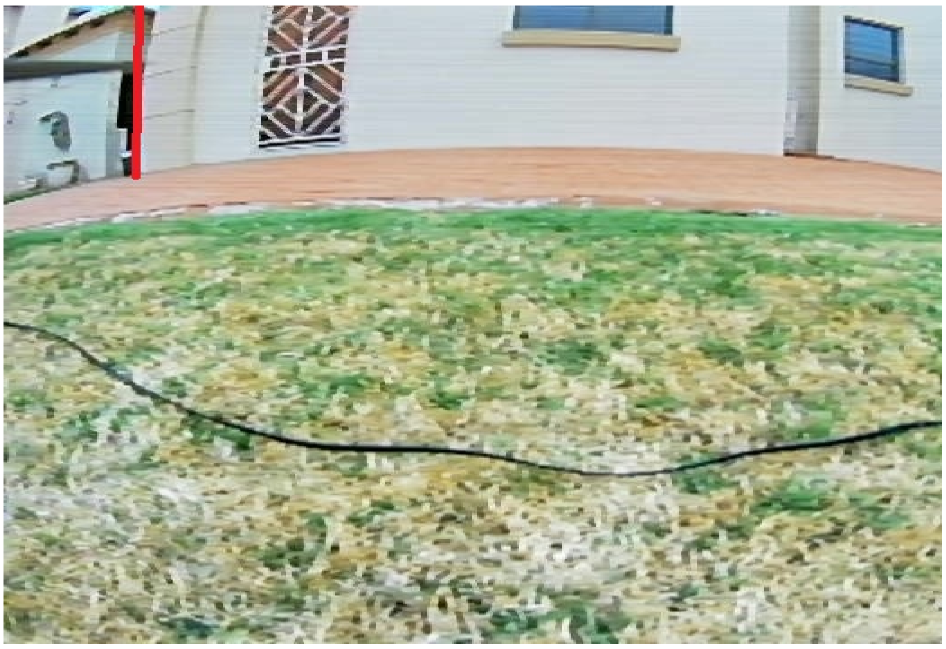
From the results obtained, the MSE is high which indicating that there was a high level of distortion in the image. The high MSE can be seen in the PSNR average as well, since the PSNR algorithm makes use of the MSE average. Despite the high level of distortion indicated by the MSE algorithm, the SSIM average was 0.727.

The results found in [30] are shown in Table 4.27. The results obtained in the quadcopter image distortion test show that the MSE, PSNR and SSIM are better in [30]. A possible cause for the image distortion on the quadcopter could be vibration from the motors, that are propagated through the frame.

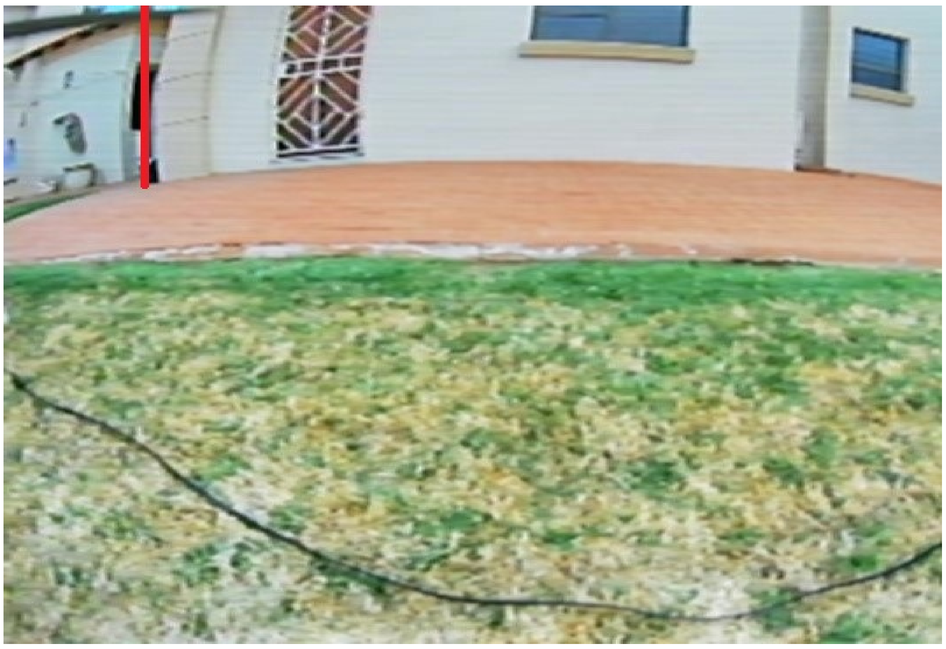
Table 4.27. Average results found in [30].

	SSIM	**MSE**	**PSNR**
Average	0.85	648.96	18.55

Figure 4.33(a) shows an image from the quadcopter camera that has a low distortion and Figure 4.33(a) shows an image from the quadcopter camera that has a high distortion. A vertical red line was drawn on both images to indicate the distortion of the pillar in the image. It can be seen in Figure 4.33(b) that the pillar deviates more from the red line, when compared to Figure 4.33(a).



(a) Low distortion image.



(b) High distortion image.

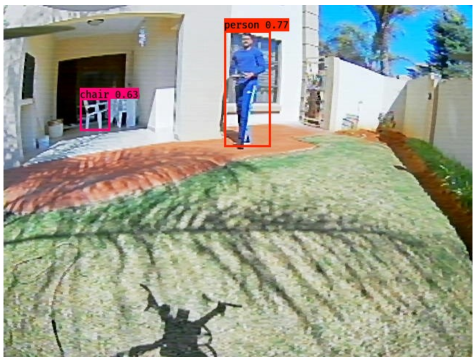
Figure 4.33. Comparison of distortion. (a) Low distortion image. (b) High distortion image.

4.5 INTEGRATED SYSTEM

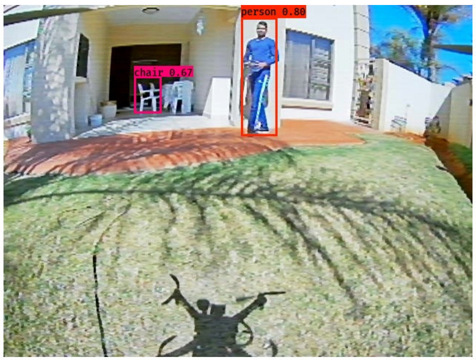
The control, navigation and image processing subsystems were integrated and the results are shown in this section.

4.5.1 Single person

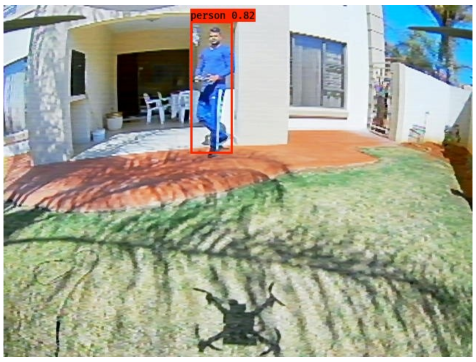
This test shows the person-tracking ability of the system. The image from the quadcopter's camera was sent to the base station for image processing. After the image had been processed an instruction was sent from the base station to the quadcopter to keep the person in the centre of the image's frame. Figure 4.34(a) - (h) show the images from the quadcopter. It is clear that, as the person was moving, the platform rotated to keep the person in the centre of the frame.


person 0.77
chair 0.63

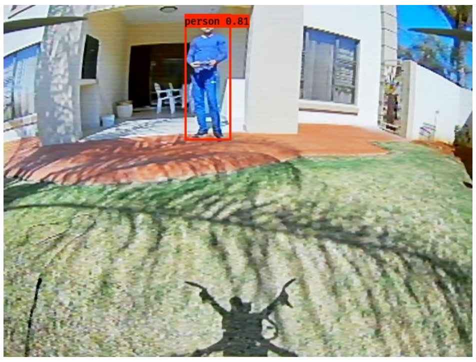
(a)


person 0.80
chair 0.67

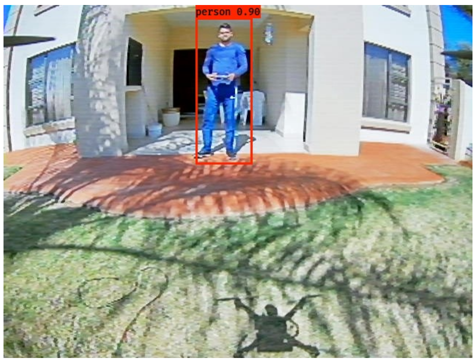
(b)


person 0.82

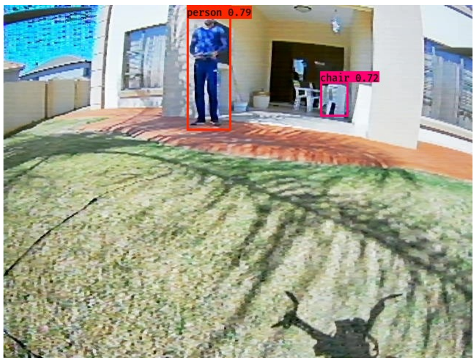
(c)


person 0.81

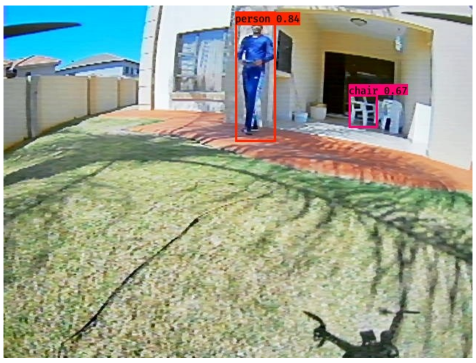
(d)


person 0.90

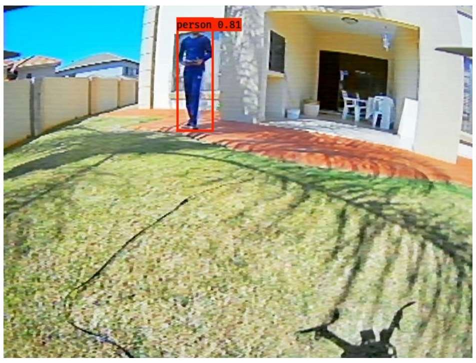
(e)


person 0.79
chair 0.72

(f)



(g)



(h)

Figure 4.34. Tracking test - Quadcopter rotates to centre the person in the frame.

The video of person-tracking with a single person test can be seen at this link.

4.5.2 Multiple people

This test shows the person-tracking ability of the system. The image from the quadcopter's camera was sent to the base station for image processing. After the image had been processed an instruction was sent from the base station to the quadcopter to keep the person of interest in the centre of the frame, as well as to keep a 5 ± 0.5 m distance between the person of interest and the quadcopter. This test was carried out with four people:

- Person 1 : Blue shirt and blue pants
- Person 2 : Grey shirt and black pants
- Person 3 : Orange shirt and black pants
- Person 4 : Grey shirt and blue pants

The person of interest was set to Person 3 (orange shirt and black pants). It can be seen that Person 2 and Person 3 had the same colour pants but had distinctive colour shirts. The quadcopter was set to fly slightly higher than head height and the camera was angled slightly downwards to make sure that the subject's entire body would be detected in each frame. The quadcopter also detected and measured the distance to the surrounding people to ensure their safety. Figure 4.35 to Figure 4.38 form part of a single complete test; however, the figures were split up for effortless readability.

From Figure 4.35(a) - (c) it can be seen that the quadcopter detected the four people in the images. It can be seen that the person of interest proceeded to the left of the quadcopter and the quadcopter rotated on its axis to keep the person of interest in the middle of the frame.

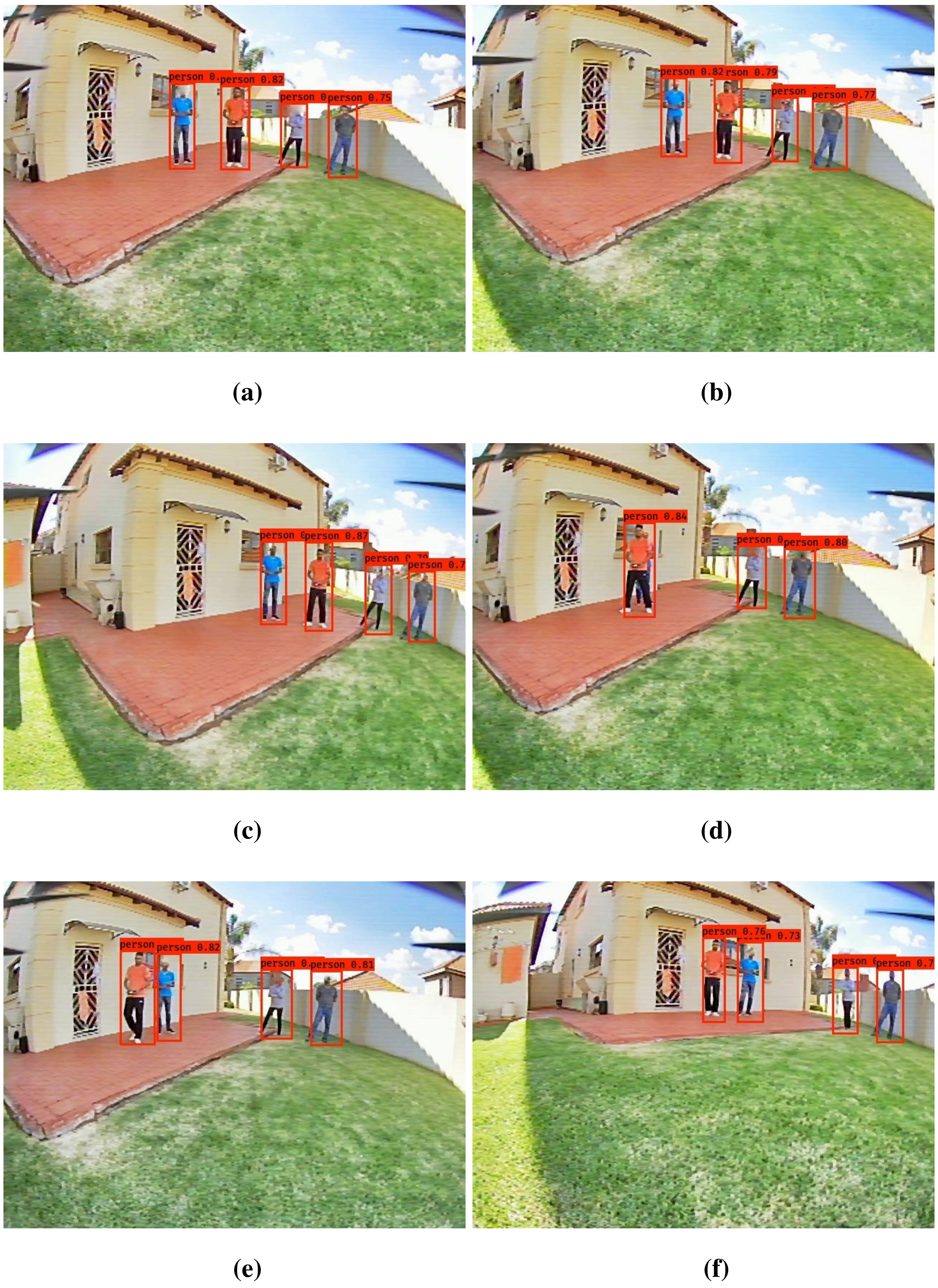
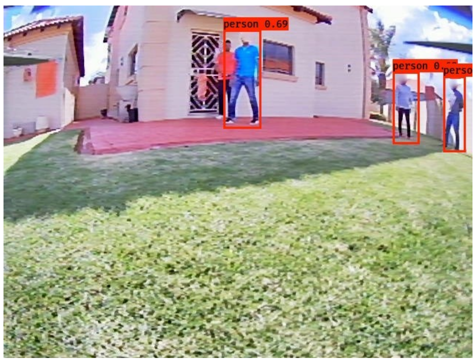


Figure 4.35. Person-tracking (multiple people): Quadcopter rotation test.

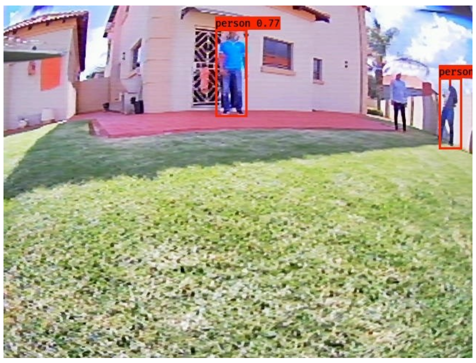
In Figure 4.36(a) - (e) a person of non-interest blocked the person of interest. It can be seen that the quadcopter did not perform any movements when it could not locate the person of interest in the frame and did not follow a person of non-interest.

(a)

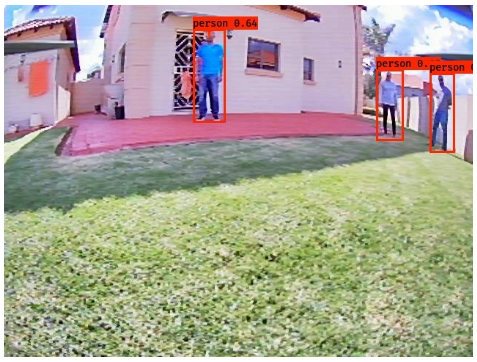
(b)



(c)



(d)



(e)

Figure 4.36. Person-tracking (multiple people): Quadcopter could not locate the person of interest.

In Figure 4.37(a) - (d) the person of non-interest walked past the person of interest; therefore the quadcopter could continue tracking the person of interest. In Figure 4.37(e) - (f) the person of interest proceeded to the left of the quadcopter and the quadcopter successfully followed the person of interest.

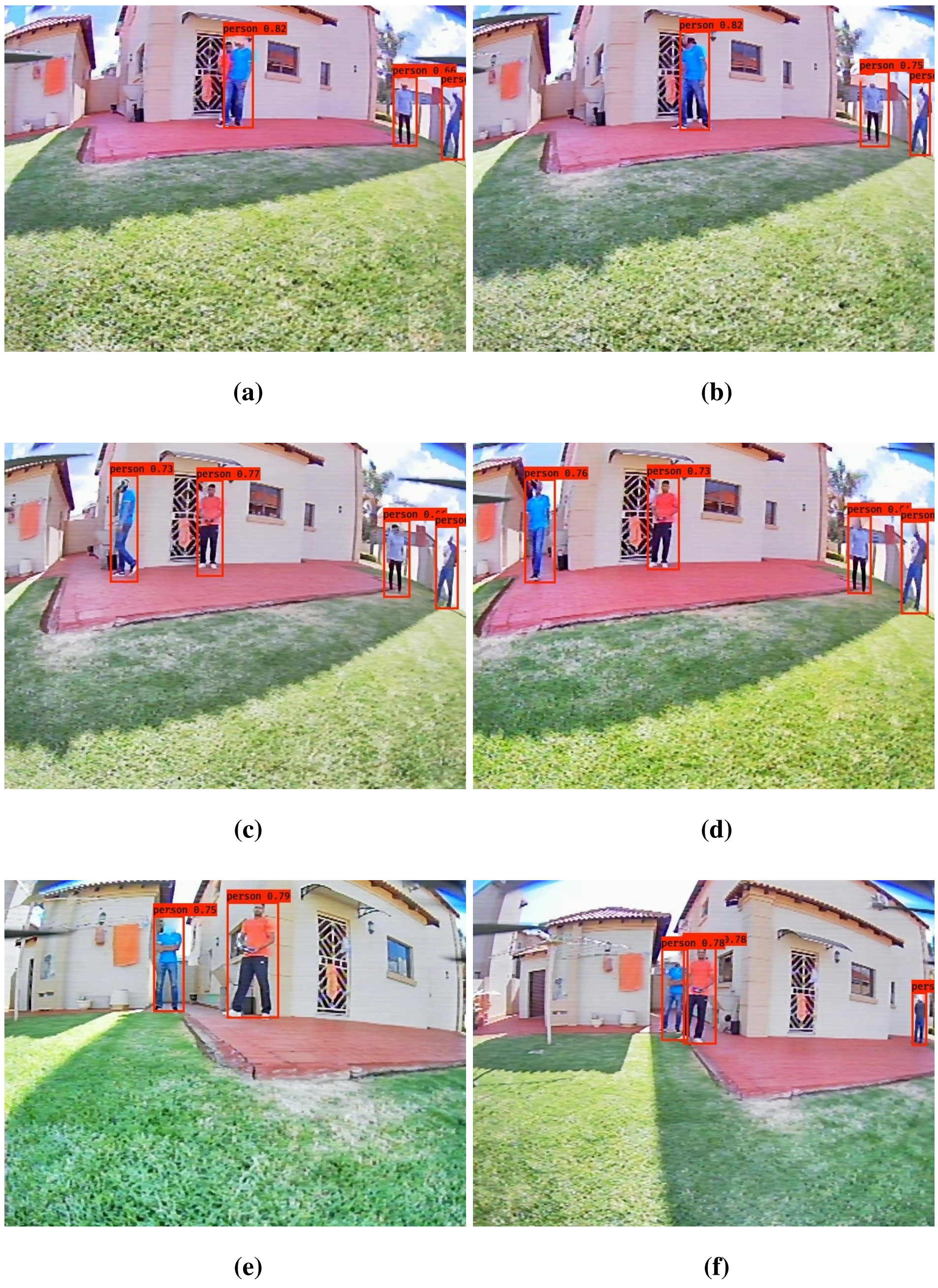
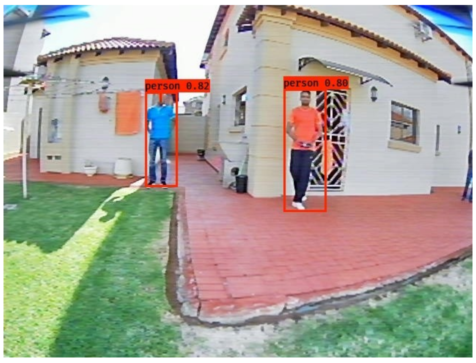
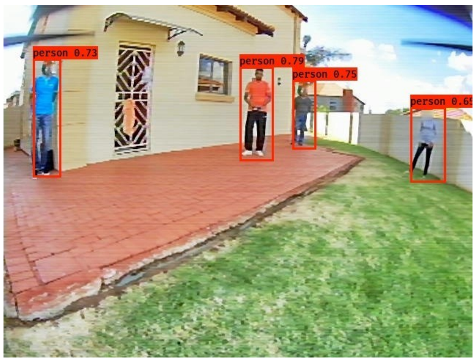


Figure 4.37. Person-tracking (multiple people): Quadcopter located the person of interest after being blocked.

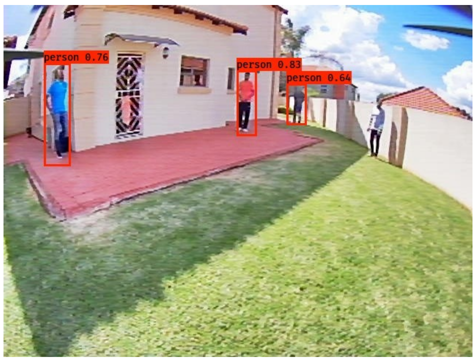
In Figure 4.38(a) - (j) the person of interest proceeded to the right of the quadcopter and the quadcopter successfully rotated on its axis to follow the person of interest. In Figure 4.38(c) - (g) it can be seen that the person of interest had moved away from the quadcopter; therefore the quadcopter moved forward to keep the person of interest at the predefined distance of 5 ± 0.5 m. Finally, in Figure 4.38(h) - (j) the person of interest moved forward; therefore the quadcopter moved backwards to keep the person of interest at the predefined distance of 5 ± 0.5 m.



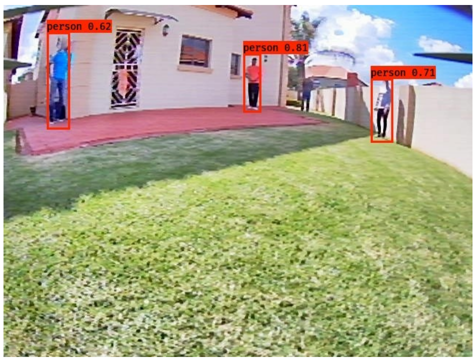
(a)



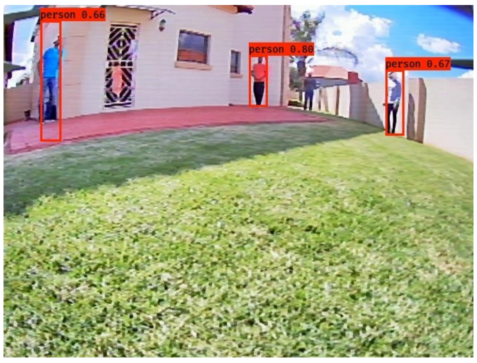
(b)



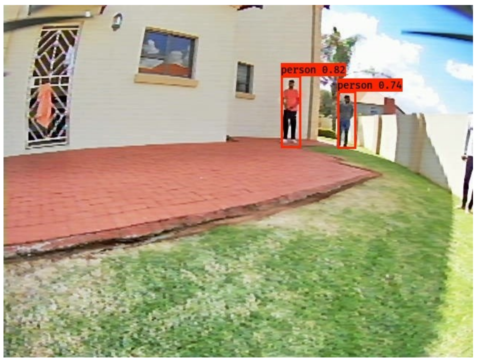
(c)



(d)

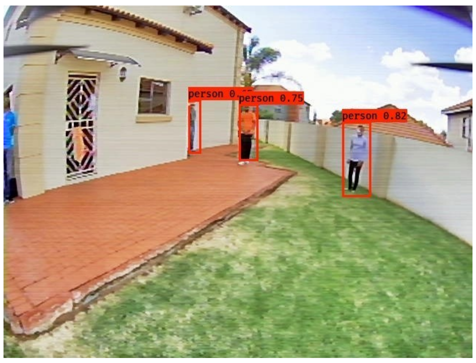


(e)

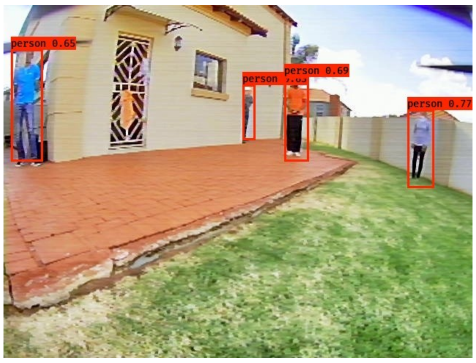


(f)

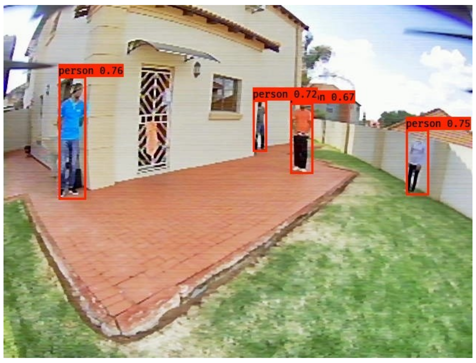
(g)



(h)



(i)



(j)

Figure 4.38. Person-tracking - multiple people: Quadcopter translation and rotation to keep the person of interest at the predefined distance and to keep the person of interest in the centre of the frame.

The time of each frame was 0.71 s, since that is the time it takes to capture an image, send it to the base station, undergo image processing and to send it back to the quadcopter. The distance that the person had moved, was recorded and the number of frames it took the quadcopter to make the necessary movements, was recorded. The speed of the person was calculated by :

$$speed = \frac{distance}{time}. \tag{4.2}$$

After twenty tests had been conducted, it was found that the quadcopter could follow a person at an average speed of 1.15 m/s. The video of person-tracking with multiple person test can be seen at this link.

4.6 CHAPTER SUMMARY

This chapter presented the results found in the experiments that had been performed to demonstrate the capabilities of the quadcopter. The improvements of the Kalman filter were presented. The stability controller in the pitch an roll direction was presented and various important aspects were noted, such as settling time, rise time and overall error was discussed. The altitude controller and the position controller results were presented and discussed. For each controller the effects of two-blade and three-blade propellers were discussed. It was found that the pitch and roll angle of the three-blade propellers had a lower overshoot; however, it had a longer response time compared to the two-blade propellers. The position estimation algorithm was evaluated by using the Euclidean distance error. Each controller was compared to previous research papers to evaluate the performance.

The person detection, colour classification and distance estimation algorithms' results were presented. An image distortion test was also conducted and measured using various metrics. Finally, the integrated system was presented. That included the integration of the navigation and control system, person detection, colour classification and distance estimation algorithms. The application chosen for the integrated system was a person-tracking system that demonstrated the navigation and control of the quadcopter to follow a person of interest, based on the colour of his/her shirt and pants. It was found that real-time image processing algorithms could successfully detect and track a person of interest without the need for image distortion corrections.

Links to a few videos were added to demonstrate the real-time image processing from the quadcopter point-of-view.

CHAPTER 5 CONCLUSION

5.1 CONCLUSION

The objective of this research has been to develop a stability and navigation controller that can be used for real-time image processing applications. Most existing quadcopter studies are focused on the image processing application and use an off-the-shelve stability and navigation controller. While the real-time image processing application is an intricate system, it will prove to be useless without a quality stability and navigation controller.

In this dissertation a complete stability and navigation controller for a quadcopter was developed, with the intent to be used for real-time image processing applications. The stability controller was responsible for keeping the quadcopter stable and predictable. The navigation controller was responsible for estimating the quadcopter's current and next position, as well as manipulating the motors to carry out the instruction(s) required. Person-tracking was used as an image processing application to demonstrate the stability and the navigation capabilities.

The stability controller used data from a gyroscope and accelerometer that were combined to measure pitch, roll and yaw of the quadcopter. The pitch, roll and yaw were used to compute a PID value to keep the quadcopter stable during flight. The stability controller was intensely tested and was accompanied by multiple pitch and roll angle correction response graphs, along with multiple tables to summarise the important information that had been deduced from the graphs. The stability controller used a PID controller and performed well, in order to provide a stable flight.

The comparison between the two-blade propellers and the three-blade propellers proved to be an interesting addition to the overall stability of the quadcopter. Both the propellers provided a stable flight for the quadcopter. The three-blade propellers were advantageous for stability; however, they had a slightly longer response time. The choice of propellers is primarily dependent on if a fast response time is required. The stability controller with the two-blade propellers used in this project, achieved a maximum error of between -1.58° and 0.78°. The stability controller with the three-blade propellers used in this project, achieved a maximum error of between -0.35° and 0.44°. The settling time for the two-blade propellers was slightly faster than [57]. It can be noted that the stability controller with the two-blade propellers had a similar error to [11] and provided better accuracy compared to the stability controller in [56]. It was found that the yaw average error found by the developed system was 3.2°, which was similar to the average error of 3.4° found in [72].

The navigation controller made use of the GPS and accelerometer data that were combined by using a Kalman filter to provide horizontal position (x-axis, y-axis) estimations, as well as pressure data to estimate altitude (z-axis). The navigation controllers were accompanied by multiple pitch, roll and yaw angle response graphs with tables to evaluate the response of the controllers when the quadcopter was transitioned to a new location, as well as when the quadcopter was moved from its set position. The position controller successfully managed to maintain the position of the quadcopter with a reasonable amount of error and allowed the quadcopter to transition between a set of positions.

The altitude controller allowed the quadcopter to hold the desired altitude with a reasonable amount of error. For the altitude controller, the two-blade propellers achieved a settling time of 5 s and the three-blade propellers achieved a settling time of 6.85 s. The two-blade propellers had a faster settling time than the results achieved in research paper [57], with a settling time of 6.34 s; however, the three-blade propellers was closer to the results that had been found in research paper [57]. By using the Kalman filter it was possible to combine the GPS data and the accelerometer data to improve the position estimations. The results from the quadcopter presented herein have a slightly better position estimation in some cases, as compared to [11], with a Euclidean error between 0.35 m and 0.81 m. The navigation algorithm was able to perform at a similar standard that had been achieved in related research and in some case, even better.

The person-tracking application dealt with following a person based on the colour of the shirt and pants. The person-tracking was made possible by developing three subsystems: person detection, colour classification and distance estimation. The person-tracking application was evaluated by including images and by evaluating the accuracy of each subsystem. The time to capture an image, perform image processing and to send an instruction to the quadcopter was calculated to be 0.7 s. The distortion of the image was also evaluated by making use of metrics, such as MSE, PSNR and SSIM. The results obtained from the quadcopter image distortion test showed that the MSE, PSNR and SSIM in [30] were better. Despite the small amount of distortion present, the platform was able to acquire reliable and distortless images. The person-tracking application proved that the quadcopter developed, can be used for image acquisition and can be used for real-time applications.

The Caddx Ratel camera was also extremely lightweight, which made it favourable to mount on a quadcopter. Vibrations and unstable behaviour of the quadcopter would have affected the camera from capturing reasonable quality images, and it should be noted that the stability controller of the quadcopter was especially important to make the image processing possible. The camera was also covered with a 3D printed cover, as direct light from the sun would have had a negative effect on the camera. It was found that the images had some distortion; however, the distortion did not inhibit the image processing performance.

In conclusion, the research produced a controller that can be used for real-time image processing applications. The position controller and the altitude controller provided the quadcopter with the ability to smoothly transverse through a 3D space. The images from the quadcopter were stable and experienced some distortion; however, the results show that it can function reliably for real-time image processing applications.

5.2 FUTURE IMPROVEMENTS

For future improvements, it can be recommended to investigate the use of a smaller-size quadcopter that would slightly reduce the dependence on the controllers, as small errors in the calculations are more easily noticeable on a larger frame. A smaller frame could also improve maneuverability and make it safer to navigate in smaller areas, such as indoors. If funds would be more accessible, an investigation into the use of LiDAR sensors could prove to be beneficial for altitude measurements and distance estimation, etcetera. Even though the main focus of the quadcopter has not been image processing, the image processing algorithms could be further improved by using image processing systems, such as SLAM, for navigation in areas where GPS is unavailable.

REFERENCES

[1] A. Yousaf, K. Khurshid, M. J. Khan, and M. S. Hanif, "Real time video stabilization methods in ir domain for uavs — a review," in *2017 Fifth International Conference on Aerospace Science Engineering (ICASE)*, Islamabad, Pakistan, Nov. 2017, pp. 1–9.

[2] V. Kangunde, R. S. Jamisola, and E. K. Theophilus, "A review on drones controlled in real-time," *International Journal of Dynamics and Control*, vol. 9, no. 4, pp. 1832–1846, Dec 2021. [Online]. Available: https://doi.org/10.1007/s40435-020-00737-5

[3] Wahyudi, M. S. Listiyana, Sudjadi, and Ngatelan, "Tracking object based on gps and imu sensor," in *2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, Semarang, Indonesia, Sep. 2018, pp. 214–218.

[4] K. M. Thu and G. A. Igorevich, "Modeling and design optimization for quadcopter control system using L1 adaptive control," in *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, Canada, Oct. 2016, pp. 1–5.

[5] R. Barták, A. Hraško, and D. Obdržálek, "A controller for autonomous landing of AR.Drone," in *The 26th Chinese Control and Decision Conference (2014 CCDC)*, Changsha, China, June 2014, pp. 329–334.

[6] N. O-larnnithipong and A. Barreto, "Gyroscope drift correction algorithm for inertial measurement unit used in hand motion tracking," in *2016 IEEE SENSORS*, Orlando, FL, USA, Nov. 2016, pp. 1–3.

[7] B. Suwandi, T. Kitasuka, and M. Aritsugi, “Vehicle vibration error compensation on imu-accelerometer sensor using adaptive filter and low-pass filter approaches,” *Journal of Information Processing*, vol. 27, no. 1, pp. 33–40, Jan. 2019.

[8] A. Katiar, R. Rashdi, Z. Ali, and U. Baig, “Control and stability analysis of quadcopter,” in *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, Sukkur, Pakistan, Mar. 2018, pp. 1–6.

[9] V. Orosco, D. Chavez, O. Camacho, and J. Aguilar, “A parameter tuning approach of the Sliding Mode Control for a Quadcopter based on Genetic Algorithms,” in *2018 IEEE Third Ecuador Technical Chapters Meeting (ETCM)*, Cuenca, Ecuador, Oct. 2018, pp. 1–6.

[10] N. Bao, X. Ran, Z. Wu, Y. Xue, and K. Wang, “Research on attitude controller of quadcopter based on cascade PID control algorithm,” in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chengdu, China, Dec. 2017, pp. 1493–1497.

[11] V. A. Sharma and M. Rajesh, “Building a quadcopter: An approach for an Autonomous Quadcopter,” in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Bangalore, India, Sept. 2018, pp. 1252–1258.

[12] P. Norvig and S. J. Russell, *Artificial Intelligence A Modern Approach*, 3rd ed. Upper Saddle River, N.J.: Prentice Hall, Dec. 2009.

[13] C. C. Veedhi and V. S. D. Yeedi, “Estimation of altitude: using ultrasoinc and pressure sensors.” Master’s thesis, Blekinge Institute of Technology, 2020.

[14] *NMEA Reference Manual*, SiRF, 2005, accessed: June 14, 2020. [Online]. Available: https://www.sparkfun.com/datasheets/GPS/NMEA%20Reference%20Manual1.pdf

[15] A. Mishra. (2021, Dec.) *2 Blade vs 3 Blade Propeller: Which is Better for your Drone?* [Online]. Available:

https://circuitdigest.com/article/2-blade-vs-3-blade-propeller-which-one-is-better-for-your-drone: :text=Two%20blades%20props%20make%20more,better%20for%20stability%20and%20thrust.

[16] S. Thrun, W. Burgard, D. Fox, and R. Arkin, *Probabilistic Robotics*, ser. Intelligent Robotics and Autonomous Agents series. London, England: MIT Press, 2005.

[17] M. Doi, A. Tsujimoto, A. Kimachi, S. Nishi, and N. Ikoma, "Color Based Person Tracking with Illumination Compensation by using Multiple Statistics of Scene," in *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*, Toyama, Japan, Dec. 2018, pp. 427–430.

[18] K. Du, Y. Ju, Y. Jin, G. Li, S. Qian, and Y. Li, "MeanShift tracking algorithm with adaptive block color histogram," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, Yichang, China, Apr. 2012, pp. 2692–2695.

[19] A. Y. Ivanov, G. I. Borzunov, and K. Kogos, "Recognition and identification of the clothes in the photo or video using neural networks," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Moscow and St. Petersburg, Russia, Feb. 2018, pp. 1513–1516.

[20] B. Chen, N. Zhou, and G. Huang, "Study on target tracking algorithm integrating color and contour features," in *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, Nanjing, China, Nov. 2017, pp. 1–5.

[21] N. N. A. Aziz, Y. M. Mustafah, A. A. Shafie, M. A. Rashidan, and N. A. Zainuddin, "Real-Time Tracking Using Edge and Color Feature," in *2014 International Conference on Computer and Communication Engineering*, Kuala Lumpur, Malaysia, Sept. 2014, pp. 247–250.

[22] G. Yuqing, G. Yanning, X. Hang, and F. Zhen, "Vision-Based Detection and Tracking of a Moving Target for a Smart Car," in *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, Chongqing, China, June 2018, pp. 586–591.

[23] M. J. Abbaspour, M. Yazdi, and M. M. Shirazi, “Robust approach for people detection and tracking by stereo vision,” in *7th International Symposium on Telecommunications (IST’2014)*, Tehran, Iran, Sept. 2014, pp. 326–331.

[24] F. da Silva Guizi and C. S. Kurashima, “Real-time people detection and tracking using 3D depth estimation,” in *2016 IEEE International Symposium on Consumer Electronics (ISCE)*, Sao Paulo, Brazil, Sept. 2016, pp. 39–40.

[25] M. Z. Brown, D. Burschka, and G. D. Hager, “Advances in computational stereo,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 993–1008, Aug. 2003.

[26] Hamamatsu. (2020, May) *A visual guide to CCD vs. EM-CCD vs. CMOS*. Accessed: Aug. 2, 2020. [Online]. Available: https://camera.hamamatsu.com/jp/en/technical_guides/visual_guide/index.html

[27] Y. Cai, E. Lam, T. Howlett, and A. Cai, “Spatiotemporal analysis of ’jello effect’ in drone videos,” in *Advances in Human Factors in Robots and Unmanned Systems*, J. Chen, Ed. Washington D.C., USA: Springer International Publishing, June 2019, pp. 197–207.

[28] W. J. Jeong and Y. S. Moon, “Removing rolling shutter distortion using optical flow and RANSAC,” in *2015 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, Jan. 2015, pp. 337–338.

[29] A. D’Angelo, L. Zhaoping, and M. Barni, “A Full-Reference Quality Metric for Geometrically Distorted Images,” *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 867–881, Nov. 2009.

[30] D. Preethi and D. Loganathan, “Image Quality Assessment Metrics based on Distortion Measures,” in *2018 International Journal of Engineering Science Invention (IJESI)*, 2018, pp. 52–58.

[31] Y. Alginahi, “Preprocessing techniques in character recognition,” in *Character Recognition*, M. Mori, Ed. London, United Kingdom: IntechOpen, 2010, ch. 1.

[32] X. Sun, K. Zhang, Y. Ji, S. Wang, S. Yan, and S. Wu, “Correlation filter tracking algorithm based on multiple features and average peak correlation energy,” *Multimedia Tools and Applications*, vol. 79, no. 21, pp. 14 671–14 688, June 2020. [Online]. Available: https://doi.org/10.1007/s11042-019-7216-1

[33] Y. Ma, S. Feng, and Y. Wang, “Fully-Convolutional Siamese Networks for Football Player Tracking,” in *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, Singapore, June 2018, pp. 330–334.

[34] P. Turaga, R. Chellappa, and A. Veeraraghavan, “Advances in video-based human activity analysis: Challenges and approaches,” in *Advances in Computers*, ser. Advances in Computers, M. V. Zelkowitz, Ed. Elsevier, June 2010, vol. 80, pp. 237 – 290.

[35] F. Fang, K. Qian, B. Zhou, and X. Ma, “Real-time RGB-D based people detection and tracking system for mobile robots,” in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, Takamatsu, Japan, Aug. 2017, pp. 1937–1941.

[36] X. Zheng and N. Liu, “Color recognition of clothes based on k-means and mean shift,” in *2012 IEEE International Conference on Intelligent Control, Automatic Detection and High-End Equipment*, Beijing, China, July 2012, pp. 49–53.

[37] Y. Liu, “An Improved Faster R-CNN for Object Detection,” in *11th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 02, Hangzhou, China, Dec 2018, pp. 119–123.

[38] S. M. Abbas and D. S. N. Singh, “Region-based Object Detection and Classification using Faster R-CNN,” in *4th International Conference on Computational Intelligence Communication Technology (CICT)*, Ghaziabad, India, Feb. 2018, pp. 1–6.

[39] P. Ren, W. Fang, and S. Djahel, “A novel YOLO-Based real-time people counting approach,” in *International Smart Cities Conference (ISC2)*, Wuxi, China, Sept. 2017, pp. 1–2.

[40] W. Lan, J. Dang, Y. Wang, and S. Wang, “Pedestrian Detection Based on YOLO Network Model,” in *International Conference on Mechatronics and Automation (ICMA)*, Changchun, China, Aug. 2018, pp. 1547–1551.

[41] E. Karami, S. Prasad, and M. S. Shehata, “Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images,” *ArXiv*, vol. abs/1710.02726, Oct. 2017.

[42] E. Rosten, R. Porter, and T. Drummond, “Faster and Better: A Machine Learning Approach to Corner Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 105–119, Nov. 2008.

[43] M. Andersen, R. Andersen, and C. Larsen, “Interactive assembly guide using augmented reality.” Master’s thesis, Aalborg University, June 2009.

[44] J. D., J. C. Mello-Román, S. Gómez-Guerrero, and M. García-Torres, “Predictive Models for the Medical Diagnosis of Dengue: A Case Study in Paraguay,” *Computational and Mathematical Methods in Medicine*, vol. 2019, pp. 1 – 7, July 2019, Accessed: Sept. 7, 2020. [Online]. Available: https://doi.org/10.1155/2019/7307803

[45] M. Mazumdar, V. Sarasvathi, and A. Kumar, “Object recognition in videos by sequential frame extraction using convolutional neural networks and fully connected neural networks,” in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, Chennai, India, Aug. 2017, pp. 1485–1488.

[46] J. W. Oh, D. Park, and J. Jeong, “Fault Detection for Lubricant Bearing with CNN,” in *2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS)*, Singapore, Mar. 2019, pp. 142–145.

[47] B. Liu, W. Zhao, and Q. Sun, “Study of object detection based on Faster R-CNN,” in *2017 Chinese Automation Congress (CAC)*, Oct. 2017, pp. 6233–6236.

[48] H. Yanagisawa, T. Yamashita, and H. Watanabe, “A study on object detection method from manga images using CNN,” in *2018 International Workshop on Advanced Image Technology (IWAIT)*, Chiang Mai, Thailand, Jan. 2018, pp. 1–4.

[49] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” *CoRR*, vol. abs/1612.08242, 2016, Accessed: Sept. 17, 2020. [Online]. Available: http://arxiv.org/abs/1612.08242

[50] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *CoRR*, vol. abs/1804.02767, 2018, Accessed: Sept. 19, 2020. [Online]. Available: http://arxiv.org/abs/1804.02767

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 779–788.

[52] B. Banerjee, S. V. G, and K. M. Buddhiraju, “Satellite image segmentation: A novel adaptive mean-shift clustering based approach,” in *2012 IEEE International Geoscience and Remote Sensing Symposium*, Munich, Germany, July 2012, pp. 4319–4322.

[53] T. Luukkonen, “Modelling and control of quadcopter,” *Independent research project in applied mathematics, Espoo*, vol. 22, p. 22, Aug. 2011.

[54] T. Jirinec, “Stabilization and control of unmanned quadcopter,” Master’s thesis, Czech Technical University in Prague, May 2011.

[55] R. S. M. Sadigh, “Optimizing PID Controller Coefficients Using Fractional Order Based on Intelligent Optimization Algorithms for Quadcopter,” in *2018 6th RSI International Conference on Robotics and Mechatronics (IcRoM)*, Tehran, Iran, Oct. 2018, pp. 146–151.

[56] M. Waliszkiewicz, K. Wojtowicz, and Z. Rochala, “Experimental method of controller tuning for quadcopters,” in *2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, Turin, Italy, June 2019, pp. 165–170.

[57] M. A. Lukmana and H. Nurhadi, "Preliminary study on Unmanned Aerial Vehicle (UAV) Quadcopter using PID controller," in *2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)*, Surabaya, Indonesia, Oct. 2015, pp. 34–37.

[58] *MS5611-01BA03 Barometric Pressure Sensor, with stainless steel cap*, TE connectivity, 2012, Accessed: Nov. 21, 2020. [Online]. Available: https://www.rgbautomatyka.pl/de/p/file/73f41de91e83143aeb124aaf925df30d/MS561101BA03-50__TE_CONNECTIVITY_1_2.pdf

[59] Z. Mustapa, S. Saat, S. H. Husin, and N. Abas, "Altitude controller design for multi-copter UAV," in *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, Langkawi, Malaysia, Sept. 2014, pp. 382–387.

[60] A. Deep, M. Mittal, and V. Mittal, "Application of Kalman Filter in GPS Position Estimation," in *2018 IEEE 8th Power India International Conference (PIICON)*, Kurukshetra, India, Dec. 2018, pp. 1–5.

[61] Y. Yang, J. Zhou, and O. Loffeld, "GPS receiver tracking loop design based on a Kalman filtering Approach," in *Proceedings ELMAR-2012*, 2012, pp. 121–124.

[62] A. P. Dhongade and M. Khandekar, "GPS and IMU Integration on an autonomous vehicle using Kalman filter (LabView Tool)," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, Madurai, India, May 2019, pp. 1122–1125.

[63] *NEO-M8 u-blox M8 concurrent GNSS modules Datasheet*, u-blox, 2015, accessed: June 14, 2020. [Online]. Available: https://www.u-blox.com/sites/default/files/NEO-M8_DataSheet_%28UBX-13003366%29.pdf

[64] M. J. Caruso, "Applications of magnetoresistive sensors in navigation systems," in *SAE International Congress and Exposition*. United States: SAE International, Feb. 1997. [Online]. Available: https://doi.org/10.4271/970602

[65] NOAA/NCEI and CIRES. (2019) *Maps of Magnetic Elements from the WMM2020*. Accessed: Dec. 3, 2020. [Online]. Available: https://ngdc.noaa.gov/geomag/WM

[66] M. J. Caruso, "Applications of magnetic sensors for low cost compass systems," in *IEEE 2000. Position Location and Navigation Symposium (Cat. No.00CH37062)*, San Diego, CA, USA, Mar. 2000, pp. 177–184.

[67] F. Merrifield, "Chapter iv - ship magnetism (1)," in *Ship Magnetism and the Magnetic Compass*, ser. The Commonwealth and International Library of Science, Technology, Engineering and Liberal Studies: Navigation and Nautical Courses, F. Merrifield, Ed. Pergamon, 1963, pp. 34 – 37, Accessed: Dec. 3, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780080097695500077

[68] A. Kuncar, M. Sysel, and T. Urbanek, "Calibration of low-cost accelerometer and magnetometer with differential evolution," in *2017 International Conference on Military Technologies (ICMT)*, Brno, Czech Republic, June 2017, pp. 414–418.

[69] S. Zhang, L. Chai, and L. Jin, "Vehicle Detection in UAV Aerial Images Based on Improved YOLOv3," in *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, Nanjing, China, Nov. 2020, pp. 1–6.

[70] A. Wang, "Vehicle recognition algorithm based on improved YOLOV3," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 9, Chongqing, China, Dec. 2020, pp. 2301–2305.

[71] Y. Zheng, B. K. Iwana, and S. Uchida, "Discovering Class-Wise Trends of Max-Pooling in Subspace," in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Niagara Falls, NY, USA, Aug. 2018, pp. 98–103.

[72] Sumardi, M. S. Sulila, and M. A. Riyadi, "Particle swarm optimization (PSO)-based self tuning proportional, integral, derivative (PID) for bearing navigation control system on quadcopter," in *2017 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, Semarang, Indonesia, Oct. 2017, pp. 181–186.

[73] J. Redmon and A. Farhadi. (2016) *YOLO: Real-Time Object Detection*. Accessed: Sept. 17, 2020. [Online]. Available: https://pjreddie.com/darknet/yolov2/