Cape Peninsula
University of Technology

DEVELOPMENT OF AN AUTONOMOUS AIRBORNE DOCKING AND UNDOCKING SYSTEM FOR MICRO DRONES TO A MOTHER DRONE

by

INYENI AMAKURO SHOWERS

Thesis submitted in fulfilment of the requirements for the degree

Master of Engineering: Mechanical Engineering

in the Faculty of Engineering and the Built Environment

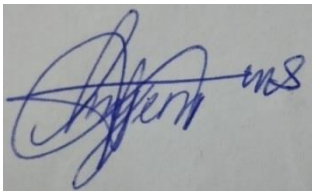
at the Cape Peninsula University of Technology

Supervisor: A/Prof. Oscar Philander
Co-supervisor: Mr. Riddles Morney

Bellville
December 2019

DECLARATION

I, INYENI AMAKURO SHOWERS, declare that the contents of this dissertation/thesis represent my own unaided work, and that the dissertation/thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.



29/07/2020

Signed **Date**

ABSTRACT

This work covers the concept and algorithms for an autonomous quadcopter (micro drone) to deploy from a docked position on a vertical take-off and landing (VTOL) drone, which is the mother drone (AMTL Guardian 4). The deployed quadcopter then returns autonomously to its initial docked position after completing a mission. This airborne docking system allows for refuelling or assigning another mission to the micro drone after successful docking. The mother drone is modelled as an autonomous ground station using the Pitsco education robotics kit. A docking arm mechanism mounted on the mother drone provides a platform for docking the micro drone and a grabbing mechanism for holding the docked drone in position. The micro drone is a DJI tello drone. It is programmed using the DJI tello python SDK. OpenCV, an open source python based computer vision library is used to implement object detection and tracking on the micro drone. For the ground station, LabVIEW Vision Acquisition and Vision Assistant are used to develop object detection and tracking algorithms for the docking system. An algorithm tagged as the hand-shake docking algorithm is used to coordinate autonomous movements in the mother drone and micro drone to achieve docking and undocking of the micro drone from the mother drone. It is projected that this research will advance current drone technology and create increasing interest on research towards autonomous airborne docking systems.

ACKNOWLEDGEMENTS

I wish to thank:

- My parents and family for their relentless support towards the success of my educational pursuits and career.
- My supervisor A/Prof Philander Oscar for his patience, absolute support and mentorship during this research
  .

The financial assistance of the National Research Foundation towards this research is acknowledged. Opinions expressed in this thesis and the conclusions arrived at, are those of the author, and are not necessarily to be attributed to the National Research Foundation.

DEDICATION

This work is dedicated to the Almighty God who is the custodian of all wisdom, understanding and inspiration.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

GLOSSARY

Abbreviations	Definition
VTOL	Vertical Take-off and Landing
AMTL	Advanced Manufacturing and Technology Laboratory
UGV	Unmanned Ground Vehicle
UAV	Unmanned Aerial Vehicle
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle
LiPO	Lithium-Potassium
PID	Proportional-Integral-Derivative
LQR	Linear Quadratic Regulation
SAR	Search and Rescue
GCS	Ground Control System
IR	Infrared
GPS	Global Positioning System
SVM	Support Vector Machine
QoS	Quality of Service
PA	Precision Agriculture
VI	Virtual Instrument (LabVIEW)
VI	Vegetation Indices (Precision Agriculture)
GVI	Green Vegetation Index
NDVI	Normalized Difference Vegetation Index
GNDVI	Green Normalized Difference Vegetation index
SAVI	Soil Adjusted Vegetation Index
PVI	Perpendicular Vegetation Index
EVI	Enhanced Vegetation Index
LiDAR	Light Detection and Ranging
NASA	National Aeronautics and Space Administration
DH	Denavit – Harterberg
SDK	Software Development Kit
CV	Computer Vision

CHAPTER ONE: INTRODUCTION

This chapter starts by introducing the problem statement and objectives of the research. Furthermore, the background of the research problem is briefly discussed. The concept of airborne docking and undocking is explained and some current challenges in drone design are also discussed. Anticipated application of the autonomous docking and undocking systems are discussed. The delineation, significance and contributions of the research are stated. Finally the full thesis outline is described.

1.1 Statement of Research Problem

Research and technology development of Remotely Piloted Aerial Systems (RPAS's) / Drones / Unmanned Aerial Vehicles (UAV's) has seen enormous growth over the past years. Currently, these vehicles are capable of a wide range of flight envelopes and specific mission applications. A new trend in technology development looks at the advent of systems that will provide multi-mission objectives. This can take the form of an Intelligence, Surveillance and Reconnaissance (ISR) drone that can morph into target acquisition and strike capabilities. The work presented in this thesis looks at the development of a drone system with such a multi-mission objective.

The Adaptronics Advanced Manufacturing Technology Laboratory (AMTL), a Technology Innovation Agency Technology Station, located at the Cape Peninsula University of Technology has been involved in drone technology development since 2008. In this time, the Adaptronics AMTL has developed 3 variants of a mini fixed wing UAV that fits into the typical flight envelope of aerial vehicles, i.e. ground roll, take-off, climb, cruise, loiter, cruise, descend, and landing (see figure 1.1).

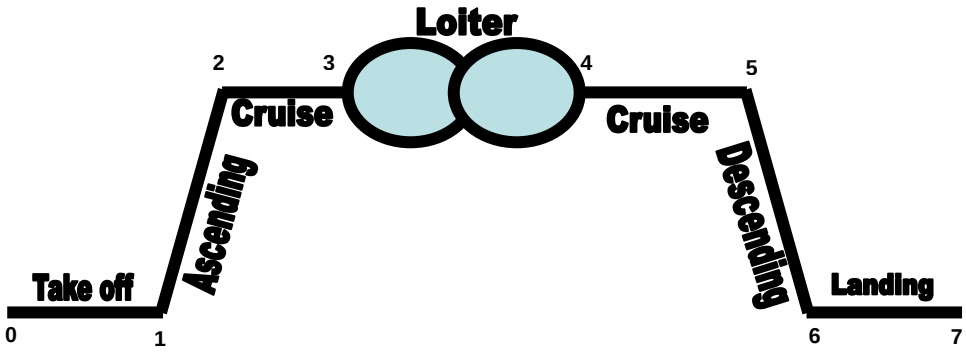


Figure 1.1: Typical Flight Envelope of an Aerial Vehicle

Over the past 2 years, the Adaptronics AMTL has developed systems for Vertical Take-off and Landing capabilities to be integrated into their fixed wing UAV (see Figure 1.2) thus creating a 4th variant. This capability will allow the fixed wing aerial system to take-off/ascend from a stationary position, descend vertically to landing, and perform hovering manoeuvres during loiter phase of flight (or any phase while the vehicle is airborne). This extended the flight envelope of the vehicle and introduced multi-mission capabilities into the conventional fixed wing aerial vehicle.

Figure 1.2: The 4th Variation of the Adaptronics AMTL Mini Fixed Wing UAV

This 4th generation UAV is projected to have an impressive airborne endurance of about 6 hours, with additional VTOL capabilities. The work presented in this thesis aims to further

extend the mission objectives of this fixed wing vehicle by deploying smaller and much more manoeuvrable micro drones while in its hovering mode. This will allow micro-drones to have access to confined areas which the larger UAV can never achieve. Micro drones such as small quadcopters can reach confined areas and perform corresponding tasks with much more ease. Quadcopters are stable and have a wide range of applications; however, the best ones currently available can only fly for a little more than 30 minutes (Justin, et al., 2016). This poses a major limitation to current quadcopter applications in terms of range and endurance. In this work, the 4th variant with its long endurance, VTOL, and hovering capabilities thus becomes a “Mother” drone and the quadcopter becomes the Micro Drone. This then gives rise to a 5th variant integrated with the flexibility and sleekness of a quadcopter. All these integrated into one autonomous drone system further extends the flight envelope as described in Figure 1.3 below. In order to realise this concept, a system for airborne docking and undocking of micro drones to the Mother drone is developed in this work.

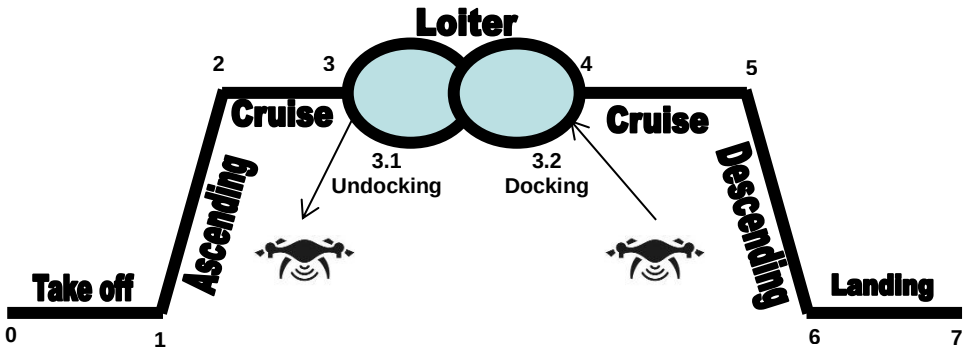


Figure 1.3: Proposed Extended Flight Envelope of the 5th Variation of the Adaptronics AMTL Mini Fixed Wing Unmanned Aerial Vehicle with Undocking and Docking Capabilities of a Micro Drone

1.2 Objectives

The primary objective of the research is to develop an autonomous docking and undocking system for a micro drone to a mother drone. In this case, the DJI Tello was used as the micro drone and a robot ground station with an attached robot arm was used as the mother drone.
In order to achieve the primary objective, sub-objectives were formulated and presented below:

1. To investigate through a literature review current quadcopter drones, application of drones and robot arm kinematics;
2. To Identify programming and hardware platforms that will achieve docking and undocking of the micro drone and mother drone;

3. To develop a Robot arm capable of implementing docking and undocking. This includes
    i. Identifying necessary hardware components for developing the robot arm;
    ii. Developing a 3D CAD concept of the arm;
    iii. Modelling the concept for applicability and integration into the airframe;
    iv. Developing a LabVIEW based computer program for autonomous control of the arm;
    v. Developing the required robot arm mechanism for achieving docking and undocking;
4. To develop the undocking capability which includes:
    i. Developing an undocking algorithm for undocking the micro drone from the mother;
    ii. Developing a LabVIEW based program to autonomously navigate the mother drone and docked micro drone to the undocking area;
    iii. Developing a LabVIEW based program to position the micro drone for release/deploy/take-off at a particular instance in time and location;
    iv. Developing a Python based program to safely initiate release/deploy/take-off of the micro drone at a particular instance of time when the docking arm is prepared for undocking; and
    v. Developing a LabVIEW program to enable the mother drone to recognise when undocking has taken place as well as retract the robot arm into its initial position.
5. To develop the Docking capability which includes:
    i. Developing docking algorithm for docking;
    ii. Developing a python based program to autonomously navigate the micro drone to the docking area at a particular instance in time and location;
    iii. Developing a LabVIEW program to autonomously navigate the mother drone to the docking area at a particular instance of time and location;
    iv. Developing a python based program to enable the micro drone to identify, track and align to the mother drone’s orientation;
    v. Developing a LabVIEW based program to enable the mother drone to identify, track and align to the micro drone’s orientation;
    vi. Developing a python based program to enable the micro drone to provide a docking signal to the mother drone while fully aligned to docking position

vii. Developing a LabVIEW program to recognize the docking signal from the micro drone and extend the docking arm for micro drone to successfully dock;
viii. Developing a python based program for micro drone to dock when the docking arm is prepared for docking
ix. Developing a LabVIEW program to grab micro drone when successfully docked and also retract the robot arm to its original position; and
x. Developing a LabVIEW program to move the mother drone with docked micro drone to a specified location.

1.3 Background of the Research Problem

Drones also referred to as aerial robots or Unmanned Aerial Vehicles (UAVs) have a predominant historical inclination to military use. However, they are now increasingly gaining ground in the global commercial market. Commercial drones, which can be categorised under fixed-wing, multi-rotor, hybrid systems, blimps drones, or ornithopters, are becoming very relevant and useful in our daily activities. These drones are finding effective applications in precision agriculture, reforestation, professional photography and videography, security surveillance, fire fighting, recreation, warfare, weather monitoring, fishing and so much more. Rapid developments in microcontrollers and programming have significantly reduced the complexity involved in controlling drones, especially quadcopters. This has also led to the dominance of quadcopters in the commercial drone industry. Quadcopters are now relatively easier to design and control.

In response to recent customer demands and consumer drone applications, drone manufacturers are consistently intensifying efforts to combine their drone technology with artificial intelligence, robotics, geo-fencing, obstacle evasion, swarm technology and advanced positioning systems to make drones more autonomous, flexible and safe. Aviation authorities and regulatory bodies are also putting appropriate laws in place to control drone usage (Bas, et al., 2016). Considering these trends, it is evident that smart drone platforms will become the norm in the nearest future.

Commercial drone builders aim at building drone models to maximise airborne time, cruising range and power economics while the drone performs its specific tasks at relatively affordable costs. This has led to the production of some recent, innovative consumer drone models such as the DJI Mavic Air, DJI Mavic Mini drone, Skydio 2, Parrot Anafi, Yuneec Mantis Q, Halo Drone Pro, GDU O2, and many more available to the public today. The designs of many more advanced drone models are still in progress. According to the commercial drone market projections, the global commercial drone market is to exceed $20 billion by 2021 while the global military market for drones outpaces all other sectors in

spending (Henry, 2018). These collective drone markets will further evolve into a $100bn market by 2020 (Shireen, 2018). Leading countries and global organisations are already investing substantially into autonomous drone technologies because of its potential to improve human living standards.

1.3.1 Docking and Undocking

Docking in this context specifically refers to the joining of two different aircrafts (mother drone and micro drone) into one single flying unit. The reverse process, which is undocking, is the separation of the single unit into two different aircrafts. This concept draws ideas from mother ships and their corresponding tender used for whaling. The tender, which is smaller, faster and more flexible, serves to transport the hunted whale to the mother ship for storage. Modern naval aircraft carriers also borrow from this concept by using a mother ship to deploy relatively smaller aircrafts close to a combat zone. In docking and berthing of spacecrafts, this concept is also very relevant (John, et al., 2018)

1.3.2 Current Challenges in Drone Design

The major challenge for drone manufacturers in recent times is designing and producing eco-friendly and technically flexible drones with long endurance (Alisa, 2018). Limitations in current battery technology has streamlined most battery powered multi-rotor drones to flight durations of a little above 30 minutes while the fixed-wing models can barely endure one or a few hours of continuous flight. The number of different tasks one drone can perform at a time, while airborne is also currently very limited. Relatively larger drones cannot effectively perform the tasks smaller drones perform and vice versa. This limits the extents of possibilities drone technology can achieve. Hence, there is a need for a drone system, platform or design that can give access to this wide range of possibilities.

The solution to these challenges lies within exploring airborne docking and undocking of micro drones to a mother drone. This concept implies that relatively small drones (preferably specialised autonomous quad copters (micro drones) be deployed from a docked position on a larger mother drone. This plays out in such a way that the micro drone, after performing its specific task autonomously, is been recovered back to the mother drone and assumes its initial docked position. Thereafter, the micro drone is ready for another deployment task or flown back to base by the mother drone. This will combine the abilities of the mother drone and micro drone in one flight, thereby maximising flight duration, operation range and flexibility.

1.4 Anticipated applications of a system for autonomous airborne docking and undocking of drones.

This system will find application in many areas. I have only listed two cardinal areas in order to create specific areas of focus.

1.4.1 Agriculture

Generally, the agricultural cycle involves steps such as tillage, planting, spraying (insecticides and pesticides), applying fertilisers, irrigation, harvesting and storage (Chandra, et al., 2016). The enormous levels of labour necessary for large scale or industrial agricultural activities usually require sophisticated farm machinery that can be manually operated or programmed. This has led to various improvements in farming and agricultural technology. Precision farming is also a growing area of interest as it harnesses technology to boost farm produce.

Currently, drones such as quad copters are used for mapping of farms, drone seed planting, spraying, irrigation and much more (Digital Transformation monitor, 2018). Figure 1.4 is an example of an Agricultural drone used for irrigation.

Figure 1.4: Agricultural drone
(Adapted from http://www.dji.com/, 2019)

1.4.1.1 Mapping of Farms

Autonomous docking and undocking of micro drones, when applied in mapping of large farms will be much more effective in covering the farm area in less time. The mother drone

can deploy the micro drones from a central position in the required farmland. Each micro drone can be assigned to a specific area for mapping, and the individual data can be synchronised wirelessly. The mother drone can also act as a recharging or refuelling station for the micro drone. Companies such as Agribotix, Micasons and Sensefly Parrot groups that already offer agricultural mapping services will find this concept very innovative and useful.

1.4.1.2 Drone Seed Planting

Drone seed planting for reforestation can be enhanced through autonomous docking and undocking of micro drones. In this case, the mother drone can act as a seed reservoir for the micro drones that do the actual planting. These micro drones can be deployed by the mother drone from a central position on the particular landscape. This will ensure a large area of land is covered in record time. Companies such as BioCarbon Engineering and DroneSeed will find this concept very relevant to their drone seed planting operations.

1.4.1.3 Spraying and Irrigation

Airborne docking and undocking concept can also be implemented in agricultural spraying of insecticides, pesticides, fertilisers as well as irrigation. In this application, the mother drone doubles as a fluid reservoir for the micro drone. The micro drone can be deployed very close to the required area for spraying of fluid (fertilisers, pesticide or water for irrigation), thereby reducing the transit distance. This concept will save time and hence improve the efficiency of the process. Companies such as DJI, Yamaha and Drone Volt will find this concept relevant to their agricultural drone applications

1.4.2 Fire Fighting

Fire outbreaks can result out of gas leaks, inflammable products, natural disasters, volcanic eruptions, electrical faults and a lot more. This can lead to huge losses in human lives and properties. However, technological advancements have greatly reduced these risks (Larry, et al., 2015) Drones can provide live footage from distress areas that could help save lives. Drones can also be used to supply first aid materials to victims or even actively fight fire outbreaks. Figure 1.5 is an Iranian fire fighting drone

Figure 1.5: Iranian fire fighting drone
(Adapted from https://www.geospatialworld.net, 2019)

Docking and undocking of micro drones to a mother drone can become an invaluable tool in modern fire fighting, providing coverage of a wider area, and giving necessary feedback on situational occurrence in a fire accident. It will provide a better chance of accessing difficult areas during fire outbreaks or even natural disasters. It will also improve the survival chances of victims of such accidents or natural disasters.

1.5 Delineation of Research

- The solution is designed for indoor conditions only
- External disturbances such as wind, rain and pressure will not be considered
- A simulation of an airborne system will be performed on a ground mobile robot
- The study does not include designing of aerofoils or propellers
- Material selection and structural analysis are not considered
- The study does not include designing of flight circuit boards and electronic components
- The focus of the research is on the development of the docking and undocking system concept and the programming algorithms.

1.6 Significance of Research

We live in an information age characterised by spontaneous increase in different kinds of technological advancements. Rapid developments in autonomous drone technology are also at the forefront of this information age. Airborne docking and undocking of drones is a concept that has the potential of positively transforming the applications of drones.

Implementing this technology on the Guardian 4 will definitely transform it into one of the most versatile drones in the world today.

1.7 Contributions of the Research

The major contributions of this research work are

- Development of a docking platform for airborne micro drones
- Programming of an autonomous quadcopter capable of airborne docking and undocking
- Improving the capabilities of the Guardian 4 drone
- A solution to the limitation of current drone technology
- A new approach to the utilisation of drones for different applications

1.8 Thesis Outline

Chapter one starts by introducing the problem statement and objectives of the research. Furthermore, the background of the research problem is briefly discussed. The concept of airborne docking and undocking is explained and some current challenges in drone design are also discussed. Anticipated application of the autonomous docking and undocking systems are discussed. The delineation, significance and contributions of the research are stated. Finally the full thesis outline is described.

Chapter two begins with a brief introduction to drones and classification of unmanned vehicles. The next section focuses on quadcopters, basic components of quadcopters and a brief description of its components. Subsequently, current application of quadcopters are reviewed which include three major areas namely Search and Rescue, Precision Agriculture and Remote Sensing Systems. Each application is first introduced, and then the challenges, research trends and future insights are broadly discussed based on literatures reviewed. The chapter ends with insights on Homogenous transformation modelling convention, forward kinematics and inverse kinematics.

Chapter three begins by laying out the docking and undocking concept as well as explaining it. The next section describes the handshake docking algorithm. The model frame work follows next and the docking system demonstration is explained. Afterwards, the hardware and software used are displayed and mentioned respectively. The software and programming section which follows shows the 7 step object detection and tracking algorithm for the mother drone as well as the LabVIEW subVIs for autonomous movement. Finally, the python based tracking program for the micro drone is presented.

Chapter four presents the narrative of the autonomous docking system demonstration. It shows the mother drone and docked micro drone taking off as a single unit from an initial

position. Afterwards, the micro drone is deployed (undocking). The mother drone moves along its mission path while the micro drone flies along its mission path. Then both units arrive at the docking area and align to each other. The micro drone then gives a signal by flying upwards and the mother drone extends its arm for docking to take place. Docking successfully takes place and the mother drone moves with the docked micro drone back to its initial take-off point.

Chapter five briefly discusses the tracking ball object and why it was chosen. Furthermore the ground station (mother drone) and micro drone tracking algorithms are briefly discussed.

Chapter six briefly presents the research conclusions, possible prospects for the technology and further areas of research.

CHAPTER TWO: LITERATURE REVIEW

This chapter begins with a brief introduction to drones and classification of unmanned vehicles. The next section focuses on quadcopters, basic components of quadcopters and a brief description of its components. Subsequently, current application of quadcopters are reviewed which include three major areas namely Search and Rescue, Precision Agriculture and Remote Sensing Systems. Each application is first introduced, and then the challenges, research trends and future insights are broadly discussed based on literatures reviewed. The chapter ends with insights on Homogenous transformation modelling convention, forward kinematics and inverse kinematics.

Drones also known as unmanned aircrafts or unmanned aerial robots have a predominant historical inclination to military use. The evolution of warfare in humankind has also led to various innovations and improvements on drone concepts. However, the 21st century has witnessed remarkable developments in civilian and commercial drone usage.

2.1 Classification of Unmanned Vehicles

Unmanned vehicles can be classified into five different types with respect to their operation. These five types are;

1. Unmanned Ground Vehicles (UGV)
2. Unmanned Aerial Vehicles (UAV)
3. Unmanned Surface Vehicles ((USV), operating on the surface of water)
4. Unmanned Underwater Vehicles (UUV)
5. Unmanned Spacecrafts

Unmanned vehicles can be either remote guided or autonomous vehicles (Wikipedia, 2018). Figure 2.1 and Figure 2.2 show some statistics on predicted value of UAV solutions in key industries and global UAV payload market predictions 2027.

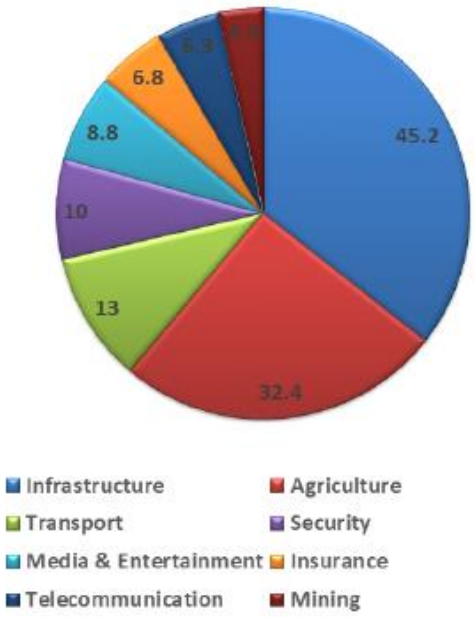


Figure 2.1: Predicted value of UAV solutions in Key Industries (Billion USD) (Adapted from (Hazim, et al., 2019))

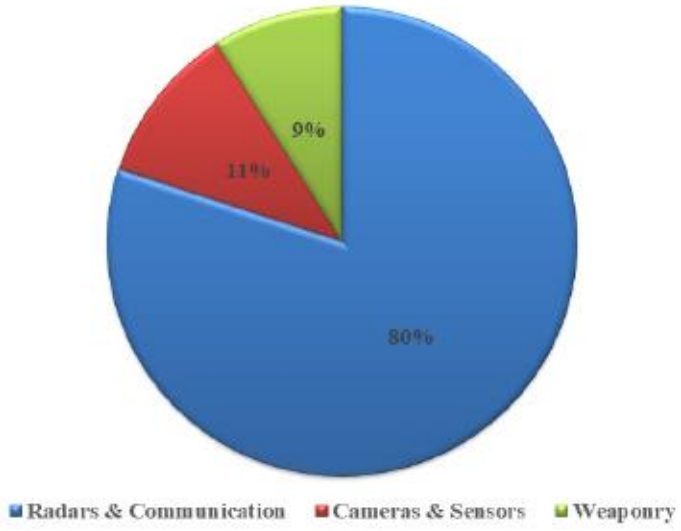


Figure 2.2: Global UAV payload market predictions 2027 (Adapted from (Hazim, et al., 2019))

According to (Yasmina, 2015), several research studies on unmanned vehicles have reported progress towards autonomous systems that do not require human interference. In this conceptual context, autonomous technical systems must be able to make decisions and

react to events without direct interventions by humans. Therefore, some essential elements are common to all autonomous vehicles. These elements include:

1. Ability to sense and perceive the environment
2. Ability to analyse, plan, make decisions and communicate using on-board computers, as well as performing actions which requires vehicle control algorithms.

UAV features may vary depending on the application in order for them to fit their specific task. Therefore, a valid classification of UAVs would require considering their various features as they are widely used for a variety of civilian operations (Korchenko & Illyash, 2013). For example, the utilisation of UAVs as aerial base stations in communications networks can be grouped based on their operating platform, which can be a High Altitude Platform (HAP) (Tozer & Grace, 2001), (Thornton, et al., 2001), and (Karapantazis & Pavlidou, 2005)] or a Low Altitude Platform (LAP) (Al-Hourani, et al., 2014), (Valcarce, et al., 2014), and (Reynaud & Rasheed, 2012) or. LAP refers to a quasi-stationary aerial communications platform that operates at an altitude less than 10 km. On the other hand, HAP operates above 10 km at very high altitudes. Vehicles using this platform are able to remain for a long time in the upper layers of the stratosphere. A comparison between HAP and LAP is presented in Table 2.1. The main UAV types for each platform and its performance parameters are also summarised in this table. Specifically, these parameters are UAV endurance, maximum altitude, weight, payload, range, deployment time, fuel type, operational complexity, coverage range, applications and examples for each UAV type is also shown in this table.

Table 2.1: Platform classification of UAV types and performance parameters (Adapted from (Hazim, et al., 2019))

Issues	**HAP**			**LAP**		
Type	**Airship**	**Aircraft**	**Balloon**	**VTOL**	**Aircraft**	**Balloon**
Endurance	Long endurance	- 15-30 hours JP-fuel - >7 days Solar	Long endurance Up to 100 days	Few hours	Few hours	From 1 day To few days
Max. Altitude	Up to 25 km	15-20 km	17-23 km	Up to 4 km	Up to 5 km	Up to 1.5 km
Payload (kg)	Hundreds of kg's	Up to 1700 kg	Tens of kg's	Few kg's	Few kg's	Tens of kg's
Flight Range	Hundreds of km's	From 1500 to 25000 km	Up to 17 million km	Tens of km's	Less than 200 km	Tethered Balloon
Deployment time	Need Runway	Need Runway	custom-built Auto launchers	Easy to deploy	Easy to launch by catapult	Easy to deploy 10-30 minutes
Fuel type	Helium Balloon	JP-8 jet fuel Solar panels	Helium Balloon Solar panels	Batteries Solar panels	Fuel injection engine	Helium

Operational complexity	Complex	Complex	Complex	Simple	Medium	Simple
Coverage area	Hundreds of km's	Hundreds of km's	Thousands of km's	Tens of km's	Hundreds of km's	Several tens of km's
UAV Weight	Few hundreds of kg's	Few thousands of kg's	Tens of kg's	Few of kg's	Tens of kg's	Tens of kg's
Public safety	Considered safe	Considered safe	Need global regulations	Need safety regulations	Safe	Safe
Applications	Testing environmental effects	GIS Imaging	Internet Delivery	Internet Delivery	Agriculture application	Aerial base station
Examples	HiSentinel80	Global Hawk	Project Loon Balloon (Google)	LIDAR	EMT Luna X-2000	Desert Star 34cm Helikite

The quadcopter, which is a very popular type of drone, has gained a lot of relevance in recent times. According to (Anudeep, et al., 2014), the quadcopter has become relevant to several applications because of the following reasons amongst others;

- No gearing is required between the rotor and motor
- No need to vary the propeller pitch for an alternating quad rotor
- The complexity of the mechanics involved is minimal
- Low maintenance
- Less load on the centre plates
- Augmentation of payload

The rapidly improving field of advanced computing and programming has also made autonomous quadcopters more reliable. Figure 2.3 depicts a quadcopter's block diagram

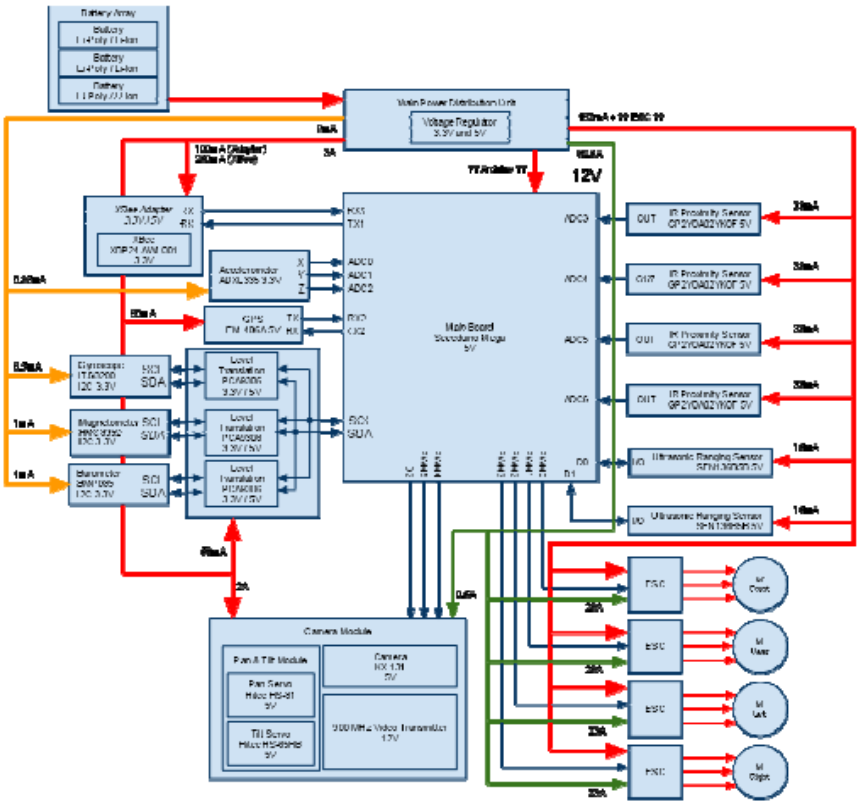


Figure 2.3: Quadcopter block diagram
(Adapted from (Gordana, et al., 2015))

2.2 Basic Components of a Quadcopter

(Andreas, 2015) and (Anudeep, et al., 2014) agree that some basic parts are necessary for an autonomous quadcopter to function properly. Below is a list of these basic parts and their basic functions

i. Body frame

The quadcopter body frame provides a rigid physical structure for the entire aircraft. It also houses the necessary electronic components that connects the motors to the rest of the aircraft. The frame must be light weight yet large enough to allow the propellers rotate without collision. Figure 2.4 shows the body frame of a quadcopter.

Figure 2.4: Carbon Fibre FPV Frame Kit QAV210
(Adapted from http://www.suscevets.co.uk)

ii. Brushless DC motors

Brushless DC motors provide the torque needed to turn the propellers at high rotational speeds. As opposed to brushed motors, brushless motors have a high thrust-to-weight ratio but require complex Electronic Speed Controllers (ESC). Brushless motors are typically given Kv ratings and current ratings, which indicate how fast the motor will spin (RPM/V) and the maximum current the motor can draw respectively. Figure 2.5 shows a brushless DC motor

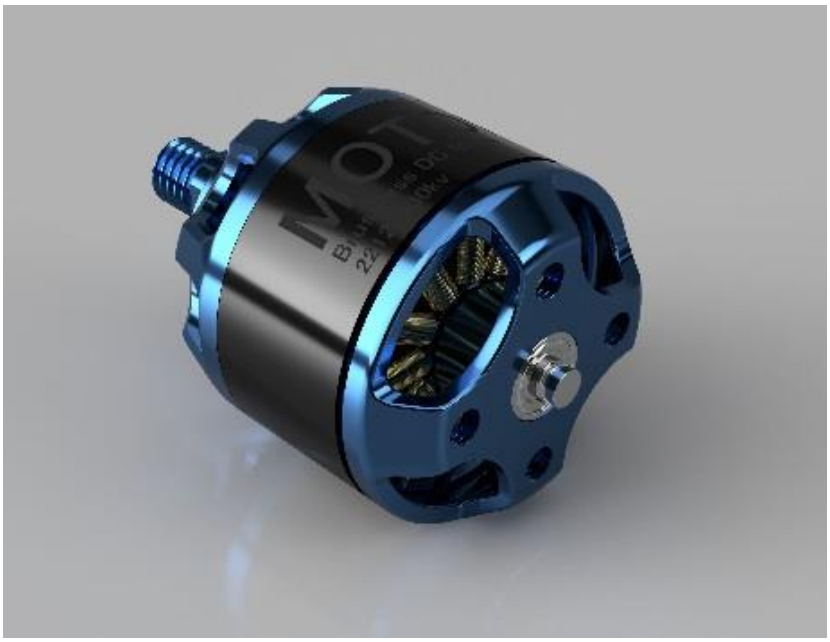


Figure 2.5: Brushless DC motor
(Adapted from https://www.gallery.autodesk.com/projects/88134/brushless-dc-motor)

iii. Propellers

The propellers are aerofoils that produce lift or thrust when they rotate in air. They are available in various sizes and materials. They are also measured according to their diameter and pitch. In order to create the right amount of lift and reduce overheating of motor, propeller selection must be done correctly. Figure 2.6 shows a set of propellers.

Figure 2.6: Propellers
(Adapted from https://www.heliguy.com/tello-propellers-p4871)

iv. Electronic Speed Controllers (ESC)

Each brushless motor requires an Electronic Speed Controller (ESC) to function properly. The ESCs receive command inputs in the form of PWM signals and give out corresponding current for the appropriate motor speed. ESCs usually have a current rating, indicating the maximum current it can provide the brushless motor without overheating. This rating must be considered when choosing an ESC for a particular motor. Figure 2.7 depicts a picture of an Electronic Speed Controller.

Figure 2.7: Electronic speed controllers

(Adapted from https://www.cityelectronics.pk/Shop/30a-esc-brushless-motor-speed-controller/#&gid=1&pid=1)

v. Flight Control board

The flight control board does the rigorous computational operations required to keep the quad copter stable and controllable. It processes input signals from the sensors and gives output signals required by the motors to keep the quadcopter under control. Some boards have integrated sensors and customised software for easy operation. Figure 2.8 is a picture of the ArduPilot APM 2.8 Controller board.

Figure 2.8: ArduPilot APM 2.8 Flight Controller Board

(Adapted from Nerokas Engineering Solutions ltd)

vi. Transmitter (Radio) and Receiver

The transmitter and receiver control the flow of radio signals from the controller to the quadcopter. The signals are transmitted through the transmitter and received by the receiver, after which they are converted to the control signals for each channel. These signals can make the quad copter perform manoeuvres such as roll, pitch, yaw or throttle. Modern receivers work on the 2.4GHz radio frequency while older ones often use 72MHz frequency. Figure 2.9 is a picture of a transmitter and receiver.

Figure 2.9: 2.4 G 4CH Radio Model RC Transmitter and Receiver
(Adapted from https://www.rcmoment.com/p-rm175.html)

vii. Servo leads

Servo leads are used as connecting wires to link the servos to a flight controller board. Figure 2.10 is a picture of servo leads.

Figure 2.10: Servo leads
(Adapted from https://www.getfpv.com/male-to-female-servo-extension-cable-26awg-jr-style-5-pcs.html)

viii. Power Distribution board (PDB)

The Power Distribution Board is a printed circuit board used to distribute power from the quadcopter battery to all the different components of the multirotor. Figure 2.11 is a picture of power distribution board.

Figure 2.11: Power Distribution Board

(Adapted from https://www.dronematters.com/index.php/atas-mini-power-distribution-board-pro.html)

ix. Batteries

Batteries are used to provide the electric power required to run the brushless motors. Lithium-Potassium (LiPo) batteries are preferable in quad copter applications because of their lightweight and high specific energy. These batteries usually have a capacity rating, in milliamp-hours (mAh) and a discharge rating (C). Figure 2.12 shows some Lipo batteries.

Figure 2.12: Lipo Batteries

(https://rogershobbycenter.com/lipoguide)

x. Sensors

Quadcopters need special sensors to aid flight stability and other automated movements that would be nearly impossible to control manually. These sensors include accelerometers, IR sensors, gyroscopes, GPS sensors etc. Sensors measure flight parameters such as altitude and flight orientation of the quadcopter. Sensors such as attitude sensors provide the flight controller with readings of the quadcopter's orientation in space. Most quadcopters incorporate an accelerometer and a gyroscope to accurately read flight orientation parameters. Figure 2.13 and Figure 2.14 show different sensors utilised by quadcopters. A quadcopter application may use a 6-axis inertial measurement unit (IMU) consisting of a gyroscope and accelerometer on the same board such as the Sparkfun 9DOF Sensor Stick, seen in Figure 2.15. This board includes an ADXL345 accelerometer, an ITG-3200 gyroscope, and an HMC3885L magnetometer.

Figure 2.13: Align dual satellite system
(Adapted from https://www.amainhobbies.com)

Figure 2.14: IR distance sensor
(Adapted from https://www.adafruit.com)

Figure 2.15: Sparkfun 9DOF Sensor Stick
(Adapted from https://www.Sparkfun.com)

2.3. Quadcopter Control Algorithm

(Erginer & Altu, 2007) and (Nonami, et al., 2010) considered and implemented the PID (Proportional-Integral-Derivative) control algorithm in their literature to control the hover altitude of the quadcopter. PID control is a type of linear control widely used in the robotics and automation industry (Nonami, et al., 2010). The PID algorithm is popularly used mainly because (Heong Ang, et al., 2005):

• It has a simple structure

• It provides good performance

• It can be tuned even if the specific model of the controlled system is unavailable.

A PID controller works by calculating the error or difference between a measured output and a desired set-point. Then it adjusts the system control inputs in such a way that the calculated error is minimised. The PID algorithm comprises mainly of three control parameters, P – Proportional, I – Integral and D – Derivative. The mathematical expression of the discrete-time PID algorithm is given in (Equation 2.1). 'P' determines the reaction to the current error; 'I' determines the reaction based on a sum of recent errors while 'D' responds to the rate at which the error has been changing. Calculation of the control input by control algorithms such as PID control may return a control input gain which may be too high for a quadcopter system. This results in a large control input magnitude which may be beyond the recognisable limits of a quadcopter system. The linear quadratic regulation (LQR) method can be employed as a solution to this problem. LQR is a form of linear optimal control regulation which reduces the magnitude of the control input without affecting the performance of the control algorithm (Prasad, et al., 2011). The LQR algorithm is used to obtain the parameter settings that will minimise the undesired deviations (altitude, for quadcopters), while concurrently limiting the energy expended by the control action. This is

done by using a mathematical algorithm that minimises a cost function or performance index with weighting factors. The cost function or performance index refers to the cumulative of deviations of measured values from its desired values (Prasad, et al., 2011). For a discrete-time LQR, the performance index is defined as (Equation 2.1). By adjusting the weight parameters R and Q, the optimal control sequence that minimises the performance index is given by (Equation 2.2) (Meng Leong, et al., 2012). Figure 2.16 shows the steps involved in the empirical methodology to obtain control parameters for a quadcopter.

$$J = \sum_{k=0}^{N} (x_k^T\, Qbx_k + u_k^T\, Ru_k)$$

Equation 2.1

$$u_k = -\mathrm{F}_k x_{k-1}$$

Equation 2.2

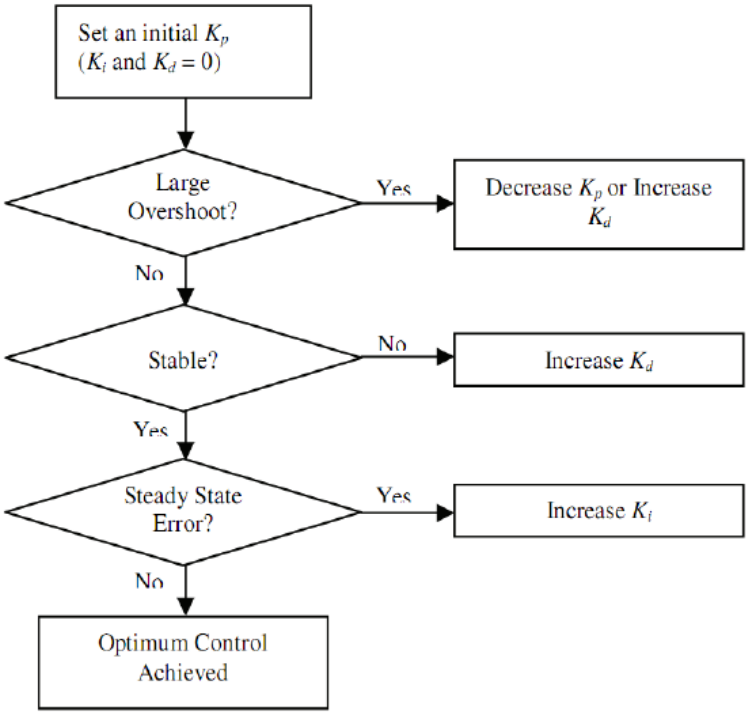


Figure 2.16: The steps involved in the empirical methodology to obtain control parameters

2.4 Applications of Quadcopters

2.4.1 Search and Rescue (SAR)

Remarkably new scientific developments have caused increasing speculations with regards to future prospects of UAVs in the context of public and civil domains. It is widely accepted that UAVs are indispensable in these domains, especially in support of SAR operations, public safety and disaster management. In the occurrence of man-made or natural disasters such as hurricanes, Tsunamis, terrorist attacks or floods, critical infrastructure like telecommunications systems, water utilities, power utilities and transportation can be affected by the disaster. This emphasises the need for rapid solutions to provide communications coverage in support of SAR operations (Hayat, et al., 2016). UAVs can be used to efficiently provide timely disaster alarms and assist in accelerating SAR operations, when the public communications networks are disrupted. UAVs can also be used to transport medical supplies to inaccessible locations. According to (Silvagni, et al., 2017), In certain disastrous situations like avalanches, poisonous gas infiltration, search for missing persons or wildfires, UAVs can play a supportive role in fast-tracking SAR operations. Furthermore, UAVs can quickly provide coverage of a large area without risking the security or safety of the personnel involved.

2.4.1.1 UAV-Based SAR System

SAR operations utilising traditional aerial systems such as aircrafts and helicopters are typically very expensive. Moreover, aircrafts need specialised flight personnel, permits and licenses for taking off and landing areas. However, the use of UAVs in SAR operations minimises costs, resources and human risks. Unfortunately, large amounts of money and time are yearly wasted on SAR operations using traditional aerial systems (Silvagni, et al., 2017).

UAVs can substantially contribute towards reducing the resources needed in SAR operations, thereby increasing efficiency. The two types of SAR systems are; single UAV systems, and Multi-UAV systems. A single UAV system is illustrated in Figure 2.17. A typical single UAV SAR system takes the following steps (Doherty & Rudol, 2007);

1. The rescue team defines the search region
2. The search operation is initiated by scanning the target area using a single UAV which is equipped with thermal or vision cameras.
3. Real-time aerial videos/images from the targeted area are sent to the Ground Control System (GCS). These images and videos are analysed by the rescue team to optimally direct the SAR operations.

In Multi-UAV systems, UAVs with on-board imaging sensors are used to identify and locate the position of missing persons. The following processes summarize the typical SAR operations that use these systems;

1. Path planning is conducted by the rescue team to compute the optimal trajectory of the SAR mission. Then, the GCS assigns paths to each UAV.
2. The search process is started. In the course of this process, all UAVs navigate through their assigned trajectories to scan the targeted region. This process utilizes video/image transmission, object detection, collision avoidance and obstacle evasion algorithms.
3. The detection process commences. During this process, if a UAV detects an object, it hovers over it, while the other UAVs act as relay nodes to facilitate coordination between all the UAVs and communications with the GCS.
4. The UAV swaps to data dissemination mode and setup a multi-hop communications link with the GCS.
5. The location of the targeted object as well as related videos and images are then transmitted to the GCS.

Figure 2.18 illustrates the use of multi-UAV systems in support of SAR operations (Scherer, et al., 2015). The Quad-rotor drones are most commonly used in SAR missions, e.g. the FALCON (AIRBORNEDRONES, 2018).

Figure 2.17: Use of Single UAV Systems in SAR Operations
(Adapted from (Hazim, et al., 2019))

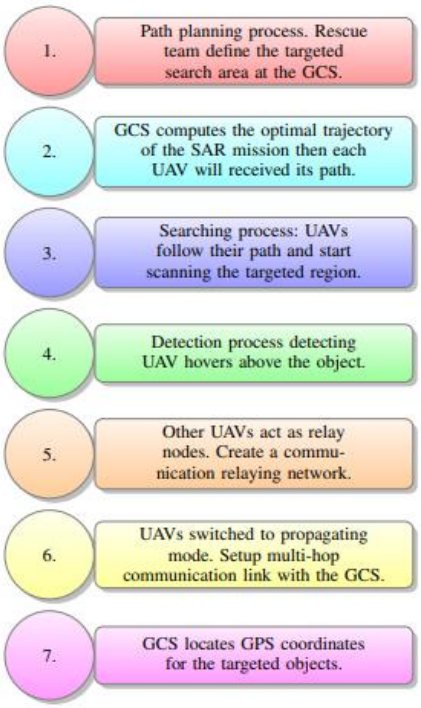


Figure 2.18: Use of Multi UAV systems in SAR operations, locate GPS coordinates for missing persons
(Adapted from (Hazim, et al., 2019))

2.4.1.2 How SAR Operations Utilize UAVs

One of the most common use-cases of UAVs is in SAR operations. Recently, their use in SAR operations have attracted considerable attention and become a topic of interest. UAVs can be utilised in SAR missions as follows:

i. Acquiring high resolution images and videos

This entails taking high resolution images and videos using on-board cameras to survey a given target area (stricken region). In this case, UAVs are used for post disaster aerial assessment or damage evaluation. This evaluation helps in the assessment of the magnitude of damage done to the infrastructure by the disaster. After this assessment, rescue teams can commence with identifying the targeted search area and proceed with SAR operations accordingly.

ii. Reduction of risks

SAR operations using UAVs can be performed autonomously and accurately without introducing unnecessary risks (Silvagni, et al., 2017). According to (Alcedo, 2018), In the Alcedo project, a prototype was developed using a lightweight quadrotor UAV equipped with GPS to help in finding lost persons. In a Capstone project (Joern, 2015), using UAVs in support of SAR operations in snow avalanche scenarios is explored. The used UAV make

use of thermal infrared imaging and Geographic Information System (GIS) data. In such scenarios, UAVs can be utilized to find avalanche victims or lost persons.

iii. Package delivery

UAVs can also be used to deliver food, water and medicines to the injured. UAVs in SAR operations are safer compared to manned aerial vehicles which poses potential dangers to crews of the flight. However, UAVs still suffer from capacity scale problems and limitations to their payloads. In (Jo & Kwon, 2017), a UAV with vertical take-off and landing capabilities was designed. A high power propellant system was also added to allow the UAV to lift heavy cargo between 10-15 Kg, which could include medicine, food, and water.

iv. Aerial Base Stations

UAVs can be used as aerial base stations for quick service recovery after complete communications infrastructure breakdown in disaster stricken areas. This is illustrated in Figures 2.19 and 2.20 for outdoor and indoor environments, respectively.

Figure 2.19: Use of UAV to provide Wireless Coverage for Outdoor users (Adapted from (Hazim, et al., 2019))

Figure 2.20: Use of UAV to provide Coverage for Indoor users (Adapted from (Hazim, et al., 2019))

2.4.1.3 Some Limitations to Utilisation of UAVs for SAR

i. Legislation

Currently in the United States, the FAA does not permit commercial use of swarms of autonomous UAVs. But it is possible to adjust the regulations to allow this type of use. Swarms of UAVs can be used to coordinate the operations of SAR teams (Smith, 2015).

ii. Weather conditions

Weather conditions pose a major challenge to UAVs as they might result in deviations in their predetermined paths. In cases of natural or man-made disasters, such as Tsunamis, Hurricanes, or terrorist attacks, weather becomes a tough and critical challenge. In such scenarios, UAVs may fail in their missions as a result of the detrimental weather conditions (Jordan, 2015).

iii. Energy Limitations

Energy consumption is one of the most important challenges facing UAVs. Usually, UAVs are battery powered. Most UAVs use batteries for UAV hovering, wireless communications, data processing and image analysis. Some SAR operations require UAVs to be operated for extended time durations over disaster stricken regions. The power limitations of UAVs demand that a decision be taken on whether UAVs should perform data and image analysis on-board in real-time, or data should be stored for later analysis to minimise power consumption (Gupta, et al., 2016), (Vergouw, et al., 2016).

2.4.1.4 Research Trends in SAR

i. Image Processing

SAR operations using UAVs can utilise image processing techniques to quickly and accurately find targeted objects. In autonomous single and multi-UAV systems image processing methods can be used to locate potential targets in support of SAR operations. Moreover, location information can be augmented to aerial images of target objects (Macke Jr, 2013). Target detection technologies such as thermal and vision cameras can be integrated with UAVs. Thermal cameras (e.g., IR cameras) can detect the heat profile to locate missing persons. Two stage template based methods can be used with such cameras (Rudol & Doherty, 2008). Vision cameras can also help in the detection process of objects and persons (Hernandez-Lopez, et al., 2012), (Mikolajczyk, et al., 2004). Methods that utilize a combination of thermal and vision cameras have been reported in the literature as in (Rudol & Doherty, 2008). During SAR operations, image processing is either done at the GCS, post target identification, or at the UAV itself, utilising on-board processors with real-time image processing capabilities. In (Sun, et al., 2016), the authors implemented a target identification method using on-board processors, and the GCS. This method utilizes image processing techniques to identify the targeted objects and their coarse location coordinates.

The UAV makes use of terrestrial networks to send the images and their GPS locations to the GCS. An alternative approach requires the UAV to capture and save high resolution videos which is analysed later at the GCS.

ii. Machine Learning

Machine learning techniques can be applied on images captured by UAVs to help in SAR operations (Giusti, et al., 2016), (Bejiga, et al., 2017). In (Bejiga, et al., 2017), the authors propose machine learning techniques applied to images captured by UAVs equipped with vision cameras. In their study, a pre-trained Convolutional Neural Network (CNN) with trained linear Support Vector Machine (SVM) is used to determine the exact image/video frame at which a lost person is potentially detected. Figure 2.21 illustrates a block diagram of a SAR system utilizing UAVs in conjunction with machine learning technology (Bejiga, et al., 2017). SAR operations that employ UAVs with machine learning technologies face many challenges. Since most UAVs are battery powered, there is a significant limitation on its on-board processing capability. Moreover, protection against adversarial attacks on the employed machine learning techniques poses another important challenge. Furthermore, reliable and real-time communications with the GCS given QoS and energy constraints is another challenge (Bejiga, et al., 2017).

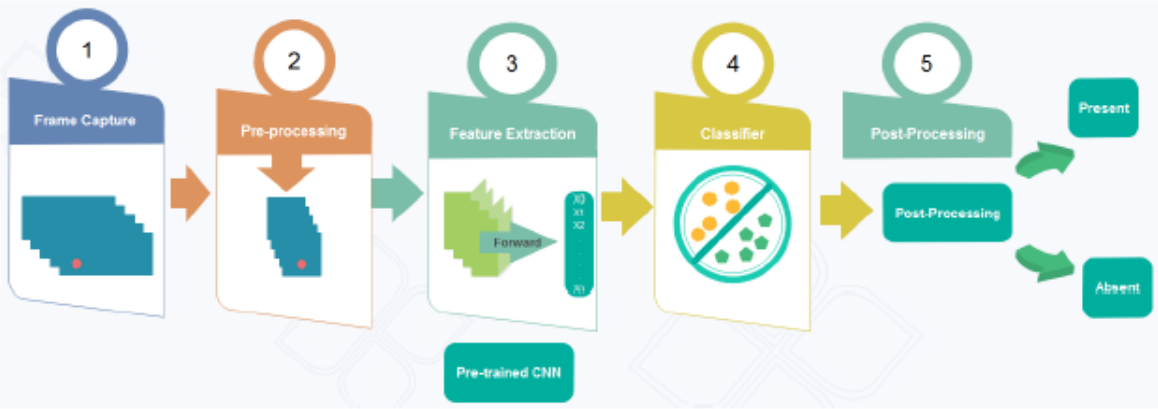


Figure 2.21: Block Diagram of SAR System Using UAVs with Machine Learning Technology (Adapted from (Hazim, et al., 2019))

2.4.1.5 Future Insights in SAR

Considering the reviewed current literature focusing on SAR scenarios using UAVs, we believe more research is needed on the following:

i. Data fusion and decision fusion algorithms that integrate the output of multiple sensors. For example, GPS can be integrated with Forward looking infrared (FLIR)

sensors and thermal sensors to realize more accurate detection solution (Rudol & Doherty, 2008).

ii. While traditional machine learning techniques have demonstrated their success on UAVs, deep learning techniques are currently off limits because of the limitations on the on-board processing capabilities and power resources on UAVs. Therefore, there is a need to design and implement on-board, low power, and efficient deep learning solutions in support of SAR operations using UAVs (Carrio, et al., 2017).

iii. Design and implementation of power-efficient distributed algorithms for the real-time processing of UAV swarm captured videos, images, and sensing data (Bejiga, et al., 2017).

iv. New lighter materials, efficient batteries and energy harvesting solutions can contribute to the potential use of UAVs in long duration missions (Vergouw, et al., 2016).

v. Algorithms that support UAV autonomy and swarm coordination are needed. These algorithms include: flight route determination, path planning, collision avoidance and swarm coordination (Alexopoulos, et al., 2013).

vi. In Multi-UAV systems, there are many coordination and communications challenges that need to be overcome. These challenges include QoS communications between the swarm of UAVs over multi-hop communications links and with the GCS (Scherer, et al., 2015).

vii. More accurate localization and mapping systems and algorithms are required in support of SAR operations. In recent times, GPS is utilised in UAVs to locate the coordinates of UAVs and target objects but it is widely known that GPS has coverage and accuracy issues. Therefore, new algorithms are needed for data fusion of the data received from multiple sensors to achieve more precise localization and mapping without coverage disruptions.

viii. The use of UAVs as aerial base stations is still nascent. During disasters, UAVs can act as aerial base stations to help trapped people and rescue teams during SAR missions. Therefore, more research is needed to study the use of such systems in providing communications coverage when the public communications network is disrupted or operates at its maximum capacity (Al-Hourani, et al., 2014).

2.4.2 Precision Agriculture

UAVs can be utilized in Precision Agriculture (PA) for crop management and monitoring (Huang, et al., 2013), (Muchiri & Kimathi, 2016), weed detection (Kazmi, et al., 2011), irrigation scheduling (Gonzalez-Dugo, et al., 2013), disease detection (Garcia-Ruiz, et al., 2013), pesticide spraying (Huang, et al., 2013) and gathering data from ground sensors (moisture, soil properties, etc.,) (Mathur, et al., 2016). The deployment of UAVs in PA is cost-effective and time conserving. This technology can help improve crop yields, farms productivity and profitability in farming systems. Moreover, UAVs facilitate agricultural management, weed monitoring, and pest damage; thereby they help to meet these challenges quickly (Primicerio, et al., 2012).

UAVs can be efficiently used for small crop fields at low altitudes with higher precision and low-cost compared with traditional manned aircraft. The use of UAVs for crop management can provide precise, accurate and reliable real time data about specific location. Furthermore, UAVs can offer high resolution images of crops to help in crop management activities such as monitoring agriculture, disease detection, detecting the variability in crop response to irrigation, reduce the amount of herbicides and weed management (Huang, et al., 2013), (Muchiri & Kimathi, 2016), (Jensen, et al., 2003). In Table 2.2, a comparison between UAVs, satellite based system and traditional manned aircraft is presented in terms of system cost, endurance, deployment time, availability, coverage area, weather and working conditions, operational complexity, applications usage and finally we present some examples from reviewed literatures.

Table 2.2: a comparison between UAVs, traditional manned aircraft and satellite based system for PA

Issues	UAVs	Manned Aircraft	Satellite System
Cost	Low	High	Very High
Endurance	Short-time	Long-time	All the times
Availability	When required	Some times	All the times
Deployment time	Easy	Need runway	Complex
Coverage area	Small	Large	Very large
Weather and working conditions	Sensitive	Low sensitivity	Require clear sky for imaging
Payload	Low	Large	Large
Operational complexity	Simple	Simple	Very complicated
Applications and usage	Carry small digital, thermal cameras & sensors	Spraying UAV system pesticide spraying	high resolution images for specific-area
Examples	(Muchiri & Kimathi, 2016), (Jensen, et al., 2003)	(Akesson & Yates, 1974)	(Reed, et al., 1994)

Table 2.3 summarizes some of the precision agriculture applications using UAVs. More specifically, this table presents several types of UAV used in precision agriculture applications, as well as the type of sensors deployed for each application and the corresponding UAV specifications in terms of payload, altitude and endurance.

2.4.2.1 The Deployment of UAVs in Precision Agriculture

In (Khanal, et al., 2017), the authors presented the deployment of UAVs in precision agriculture applications as summarized in Figure 2.21. The deployment of UAVs in precision agriculture is discussed in the following:

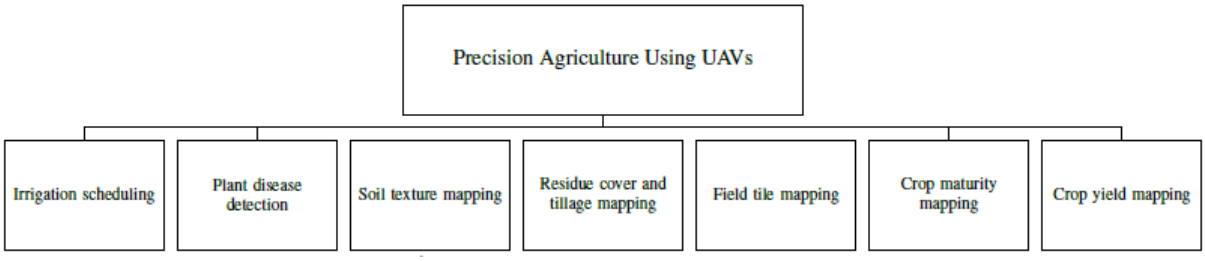


Figure 2.22: The Deployment of UAVs in Precision Agriculture Applications (Adapted from (Hazim, et al., 2019))

i. Irrigation scheduling

There are four factors that need to be monitored, in order to determine a need for irrigation:

1. Availability of soil water;
2. The amount of water needed by the various crops to grow optimally which is known as crop water need;
3. Rainfall amount;
4. Efficiency of the irrigation system (Rhoads & Yonts, 2000).

These factors can be quantified by utilizing UAVs to measure soil moisture, plant-based temperature, and evapotranspiration. For instance, the spatial distribution of surface soil moisture can be estimated using high-resolution multi-spectral imagery captured by a UAV, in combination with ground sampling (Hassan-Esfahani, et al., 2015). The crop water stress index can also be estimated, in order to determine water stressed areas by utilizing thermal UAV images (Gonzalez-Dugo, et al., 2013).

ii. Plant disease detection

In the U.S., it is estimated that crop losses caused by plant diseases result in about $33 billion in lost revenue every year (Pimentel, et al., 2005). UAVs can be used for thermal remote sensing to monitor the spatial and temporal patterns of crop diseases pre-

symptomatically during various disease development phases and hence farmers may reduce the crop losses. For instance, aerial thermal images can be used to detect early stage development of soil-borne fungus (Calder´on, et al., 2013)

iii. Soil texture mapping

Some soil properties, such as soil texture, can be an indication of soil quality which in turn influences crop productivity. Thus, UAV thermal images can be utilized to quantify soil texture at a regional scale by measuring the differences in land surface temperature under a relatively homogeneous climatic condition (De-Cai, et al., 2012), (Wang, et al., 2015).

iv. Residue cover and tillage mapping

Crop residues are essential in soil conservation by providing a protective layer on agricultural fields that shields soil from wind and water. Accurate assessment of crop residue is necessary for proper implementation of conservation tillage practices (U. D. of Interior, 2018). In (Sullivan, et al., 2004), the authors demonstrated that aerial thermal images can explain more than 95% of the variability in crop residue cover amount compared to 77% using visible and near IR images.

v. Field tile mapping

Tile drainage systems remove excess water from the fields and hence it provides ecological and economic benefits (Hofstrand, 2015). An efficient monitoring of tile drains can help farmers and natural resource managers to better mitigate any adverse environmental and economic impacts. By measuring temperature differences within a field, thermal UAV images can provide additional opportunities in field tile mapping (Steve & Kevin, 2018).

vi. Crop maturity mapping

UAVs can be a practical technology to monitor crop maturity for determining the harvesting time, particularly when the entire area cannot be harvested in the time available. For example, In Lundavra, Australia, UAV visual and infrared images from barley trial areas were used to map two primary growth stages of barley and demonstrated classification accuracy of 83.5% (Jensen, et al., 2003).

vii. Crop yield mapping

Farmers require accurate, early estimation of crop yield for a number of reasons, including crop insurance, planning of harvest and storage requirements, and cash flow budgeting. In (Swain, et al., 2010), UAV images were utilized to estimate yield and total biomass of rice crop in Thailand. In (Geipel, et al., 2014), UAV images were also utilized to predict corn grain yields in the early to midseason crop growth stages in Germany. The authors in (Sankaran, et al., 2015) presented several types of sensors that were used in UAV-based precision agriculture, as summarized in Table 2.4.

Table 2.3: Summary of UAV specifications, applications and technology used in precision agriculture

UAV Type	Applications	Payload/Altitude/Endurance	Sensor Type	References
Yamaha Aero Robot"R-50.	Monitoring Agriculture, spraying UAV systems.	20 kg / LAP / 1 hour.	Azimuth and Differential Azimuth and Differential Azimuth and Differential	(Muchiri & Kimathi, 2016)
Yanmar KG-135, YH300 and AYH3.	Pesticide spraying over crop fields.	22.7 kg / 1500m / 5 hours.	Spray system with GPS sensor system.	(Huang, et al., 2013)
RC model fixed-wing airframe.	Imaging small sorghum fields to assess the attributes of a grain crop.	Less than 1kg / LAP / less than 1 hour.	Image sensor digital camera.	(Huang, et al., 2013), (Jensen, et al., 2003)
Vector-P UAV.	Crop management (e.g. winter wheat) for site-specific agriculture, a correlation is investigated between leaf area index and the green normalized difference vegetation index (GNDVI).	Less than 1kg /105m-210m/1-6 hours deepening on the payload.	Digital color-infrared camera with a red-light-blocking filter.	(Hunt, et al., 2010)
Fixed-wing UAV.	Detection of variability in crop response to irrigation (e.g. cotton).	Lightweight camera/ 90m / Less than 1 hour.	Thermal camera, Thermal Infrared (TIR) imaging sensor.	(Sullivan, et al., 2004)
Multi-rotor micro UAV	Agricultural management, disease detection for citrus (citrus greening, Huanglongbing (HLB)).	Less than 1kg / 100 m / 10-20 min.	Multi-band imaging sensor, 6-channel multispectral camera.	(Garcia-Ruiz, et al., 2013)
Vario XLC helicopter.	Management of weed; reduction of the amount of herbicides using aerial images for crop.	7 kg / LAP / 30 min.	Sophisticated vision sensors for 3D and multispectral imaging.	(Kazmi, et al., 2011)
VIPtero UAV.	Crop management, they used UAV to acquire high resolution multi-spectral mages for vineyard management.	1 Kg / 150 m / 10 min.	Tetracam ADC-lite camera, GPS.	(Primicerio, et al., 2012)

Fieldcopter UAV.	Water assessment. UAVs were used for acquiring high resolution images and to assess vineyard water status, which can help for irrigation processes.	Smaller than 1 Kg / LAP /NA.	Multispectral and thermal cameras on-board UAV.	(Baluja, et al., 2012)
Multi-rotor hexacopter. ESAFLY A2500-WH	Cultivations analysis, processing multi spectral data of the surveyed sites to create tri-band ortho-images used to extract some Vegetation Indices	Up to 2.5 kg Kg/LAP/12-20 min.	Tetracam camera on-board UAV.	(Candiago, et al., 2015)

2.4.2.2 Challenges of UAV Utilisation in PA

There are several challenges in the deployment of UAVs in PA:

i. Thermal cameras have poor resolution and they are expensive. The price ranges from $2000-$50,000 depending on the quality and functionality, and the majority of thermal cameras have resolution of 640 pixels by 480 pixels (Khanal, et al., 2017).

ii. Thermal aerial images can be affected by many factors, such as the moisture in the atmosphere, shooting distance, and other sources of emitted and reflected thermal radiation. Therefore, calibration of aerial sensors is critical to extract scientifically reliable surface temperatures of objects (Khanal, et al., 2017).

iii. Temperature readings through aerial sensors can be affected by crop growth stages. At the beginning of the growing season, when plants are small and sparse, temperature measurements can be influenced by reflectance from the soil surface (Khanal, et al., 2017).

iv. In the event of adverse weather, such as extreme wind, rain and storms, there is a big challenge of UAVs deployment in PA applications. In conditions such as these, UAVs may fail in their missions. Therefore, small UAVs cannot operate in extreme weather conditions and even cannot take readings during these conditions.

v. One of the key challenges is the ability of lightweight UAVs to carry a high-weight payload, which will limit the ability of UAVs to carry an integrated system that includes multiple sensors, high resolution and thermal cameras (Anderson & Gaston, 2013).

vi. UAVs have short battery life time, usually less than 1 hour. Therefore, the power limitations of UAVs are one of the challenges of using UAVs in PA. Moreover, when UAVs are used to cover large areas, they need to return many times to the charging station for recharging. (Huang, et al., 2013), (Garcia-Ruiz, et al., 2013), (Jensen, et al., 2003).

2.4.2.3 Research Trends in Utilisation of UAVs for PA

i. Machine Learning:

The next generation of UAVs will utilize the new technologies in precision agriculture, such as machine learning. Hummingbird is a UAV-enabled data and imagery analytics business for precision agriculture (hummingbirdtech, 2018). It utilizes machine learning to deliver actionable insights on crop health directly to the field. The process flow begins by performing UAV surveys on the agricultural land at critical decision-making points in the growing season. Then, UAV images are uploaded to the cloud, before being processed with machine learning techniques. In the final step, the mobile app and web based platform provide farmers with necessary information and actionable insights on crop health. The advantages of utilizing UAVs with machine learning technology in precision agriculture are:

- Early detection of crop diseases;
- Precision weed mapping;
- Accurate yield forecasting;
- Nutrient optimization and planting;
- Plant growth monitoring (hummingbirdtech, 2018).

ii. Image Processing:

UAV-based systems can be used in PA to acquire high-resolution images for farms, crops and rangeland. It can also be used as an alternative to satellite and manned aircraft imaging system. Processing of these images is one of the most rapidly developing fields in PA applications. The Vegetation Indices (VI) can be produced using image processing techniques for the prediction of the agricultural crop yield, agricultural analysis, crop and weed management and in diseases detection. Moreover, the Vis can be used to create vigor

maps of the specific-site and for vegetative covers evaluation using spectral measurements (Candiago, et al., 2015), (Bannari, et al., 1995).

In (Hunt, et al., 2010), (Candiago, et al., 2015), (Bannari, et al., 1995), (Glenn, et al., 2008), most of the VIs found in the literature have been summarized and discussed. Some of these VIs are:

- Green Vegetation Index (GVI).
- Normalized Difference Vegetation Index (NDVI).
- Green Normalized Difference Vegetation index (GNDVI).
- Soil Adjusted Vegetation Index (SAVI).
- Perpendicular Vegetation Index (PVI).
- Enhanced Vegetation Index (EVI).

Many researchers made use of VIs that are derived from image processing techniques in PA. The authors in (Candiago, et al., 2015) presented agricultural analysis for vineyards and tomatoes crops. A UAV with Tetracam multi-spectral camera was deployed to take aerial image for crop. These images were processed using PixelWrench2 (PW2) software which came with the camera and it will be exported in a tri-band TIFF image. Then from the contents of this images VIs such as NDVI (Rouse Jr, et al., 1974), GNDVI (Gitelson, et al., 1996), SAVI (Huete, 1988) can be extracted. In (Laliberte, et al., 2010), the authors used UAVs to take aerial images for rangeland to identify rangeland VI for different types of plant in Southwestern Idaho. In the study, image processing and analysis was performed in three steps as shown in Figure 2.22.

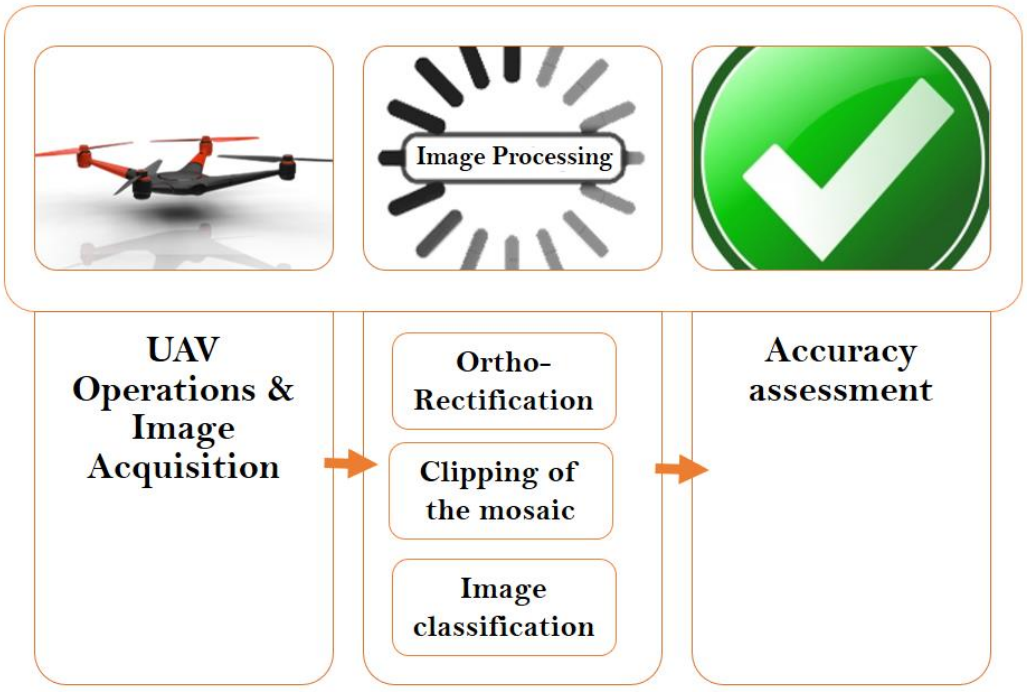


Figure 2.22: Steps of Image processing and analysis for identifying rangeland VI (Adapted from (Hazim, et al., 2019))

More specifically, the three steps were:

i. Ortho-Rectification and mosaicing of UAV imagery (Laliberte, et al., 2010). A semi-automated ortho-rectification approach were developed using PreSync procedure (Laliberte, et al., 2008).
ii. Clipping of the mosaic to the 50m x 50m plot areas measured on the ground. In this step, image classification and segmentation was performed using an object-based image analysis (OBIA) program with Definiens Developer 7.0 (Definiens, 2007), where the acquisition image was segmented into homogeneous areas (Laliberte, et al., 2010).
iii. Image classification: In this step, hierarchical classification scheme along with a rule based masking approach were used (Laliberte, et al., 2010).

2.4.2.4 Future Insights in PA

Considering the reviewed articles focusing on PA using UAVs, The following future possible directions are suggested:

i. With relaxed flight regulations and improvement in image processing, geo-referencing, mosaicing, and classification algorithms, UAV can provide a great potential for soil and crop monitoring (Khanal, et al., 2017), (Zhang & Kovacs, 2012).
ii. The next generation of UAV sensors, such as 3p sensor (SLANTRANGE, 2018), can provide on-board image processing and in-field analytic capabilities, which can give farmers instant insights in the field, without the need for cellular connectivity and cloud connection (Burwood-Taylor, 2018).

Table 2.4: UAV sensors in precision agriculture applications

Type of sensor	Operating frequency	Applications	Disadvantages
Digital camera	Visible region	Visible properties, outer defects, greenness, growth.	-Limited to visual spectral bands and properties.
Multispectral camera	Visible-infrared region	Multiple plant responses to nutrient deficiency, water stress, diseases among others.	-Limited to few spectral bands.
Hyperspectral camera	Visible-infrared region	Plant stress, produce quality, and safety control.	-Image processing is challenging. - High cost sensors.
Thermal camera	Thermal infrared region	Stomatal conductance, plant responses to water stress and diseases.	- Environmental conditions affect the performance. -Very small temperature differences are not detectable.

			- High resolution cameras are heavier.
3D camera	Infrared laser region	Physical attributes such as plant height and canopy density.	- Lower accuracies. - Limited field applications.
LiDAR	Laser region	Accurate estimates of plant/tree height and volume.	- Sensitive to small variations in path length.
SONAR	Sound propagation	Mapping and quantification of the canopy volumes, digital control of application rates in sprayers or fertilizer spreader.	- Sensitivity is limited by acoustic absorption, background noise, etc. -Lower sampling rate than laser-based sensing.

iii. More precision agricultural researches are required towards designing and implementing special types of cameras and sensors on- board UAVs, which have the ability of remote crop monitoring and detection of soil and other agricultural characteristics in real time scenarios (Primicerio, et al., 2012).
iv. UAVs can be used for obtaining high-resolution images for plants to study plant diseases and traits using image processing techniques (Patil & Kumar, 2011).

2.4.3 Remote Sensing Systems

There are two basic types of remote sensing systems: active and passive remote sensing systems (NASA, 2017). In active remote sensing system, the sensors provide the source of energy required to detect the objects. The sensor transmits radiation toward the object to be investigated, and then the sensor detects and measures the radiation that is reflected from the object. In the Majority of active remote systems used in remote sensing applications operate in the microwave portion of the electromagnetic spectrum. Thus this makes them easily propagate through the atmosphere under most conditions (NASA, 2017). The active remote sensing systems include LiDAR, laser altimeter, radar, ranging instrument, sounder and scatterometer. In passive remote sensing system, the sensor detects natural radiation that is emitted or reflected by the object as shown in Figure 2.23. The majority of passive sensors operate in the visible, infrared, thermal infrared, and microwave portions of the electromagnetic spectrum (NASA, 2017). The passive remote sensing systems include accelerometer, hyperspectral radiometer, imaging radiometer, radiometer, sounder, spectrometer and spectroradiometer. In Figure 2.24, we show the classifications of UAV aerial sensing systems. In Table 2.5, we make a comparison among the UAV remote sensing systems based on the operating frequency and applications. The common active sensors in remote sensing are LiDAR and radar. A LiDAR sensor propagates a laser beam onto the earth's surface and calculates the distance to the object by measuring the time

between transmitted and backscattered light pulses. A radar sensor creates a two-dimensional image of the surface by computing the range and magnitude of the energy reflected from all objects. The common passive sensor in remote sensing is the spectrometer. A spectrometer sensor detects, measures, and analyses the spectral content of incident electromagnetic radiation.

2.4.3.1 Image Processing and Analysis

The image processing steps for a typical UAV mission are described in details by the authors in (Hugenholtz, et al., 2013) and (Whitehead, et al., 2013). The process flow is the same for most remotely sensed imagery processing algorithms. First, the algorithm utilizes the log file from the UAV autopilot to provide initial estimates for the position and orientation of each image. Then the algorithm applies aerial triangulation process. In the course of this process the algorithm re-establishes the true positions and orientations of the images from an aerial mission. During this process, the algorithm generates a large number of automated tie points for conjugate points identified across multiple images. A bundleblock adjustment further utilises these automated tie points to optimize the photo positions and orientations. This is done by generating a high number of redundant observations. These observations are used to derive an efficient solution through a rigorous least-squares adjustment. In order to provide an independent check on the accuracy of the adjustment, a number of check points is included by the algorithm. The oriented images are then used to develop a digital surface model. This model provides a detailed representation of the terrain surface, including the elevations of raised objects, such as trees and buildings. The digital surface model generates a dense point cloud by matching features across multiple image pairs (Whitehead, et al., 2013). At this level, a digital terrain model can be generated. This model is referred to as a bare-earth model. Compared to a surface model, a digital terrain model is a more useful product. This is because the high frequency noise associated with vegetation cover is removed. After the algorithm generates a digital terrain model, orthorectification process can then be performed to remove the distortion in the original images. After orthorectification process, the algorithm combines the individual images into a mosaic, to provide a seamless image of the mission area at the desired resolution (Whitehead & Hugenholtz, 2014). Figure 2.25 summarizes the image processing steps for remotely sensed imagery.

2.4.3.2. Flight Planning

The same steps and processes are normally followed, although each UAV mission is unique in nature. Typically, a UAV mission starts with flight planning (Hugenholtz, et al., 2013). This step depends on specific flight-planning algorithm and uses a background map or satellite

image as a reference to define the flight area. Extra data is then included, for example, the desired flying altitude, the focal length and orientation of the camera, and the desired flight path. The flight-planning algorithm will then find an efficient way to obtain overlapping stereo imagery covering the area of interest. During the flight-planning process, the algorithm can adjust various parameters until the operator is satisfied with the flight plan. Part of the mission planning process requires that the camera shutter speed settings must satisfy the different lighting conditions. If exposure time is too short, the imagery might be too dark to discriminate among all key features of interest, but if it is too long, the imagery will be blurred or will be bright. Next, the generated flight plan is uploaded to the UAV autopilot. The autopilot uses the instructions contained in the flight plan to find climb rates and positional adjustments that enable the UAV to follow the planned path as closely as possible. The autopilot reads the adjustments from the global navigation and satellite system and the initial measurement unit several times per second throughout the flight. After the flight completion, the autopilot download a log file, this file contains information about the recorded UAV 3D placements throughout the flight, as well as information about when the camera was triggered. The information in log file is used to provide initial estimates for image centre positions and camera orientations, which are then used as inputs to recover the exact positions of surface points (Whitehead & Hugenholtz, 2014).

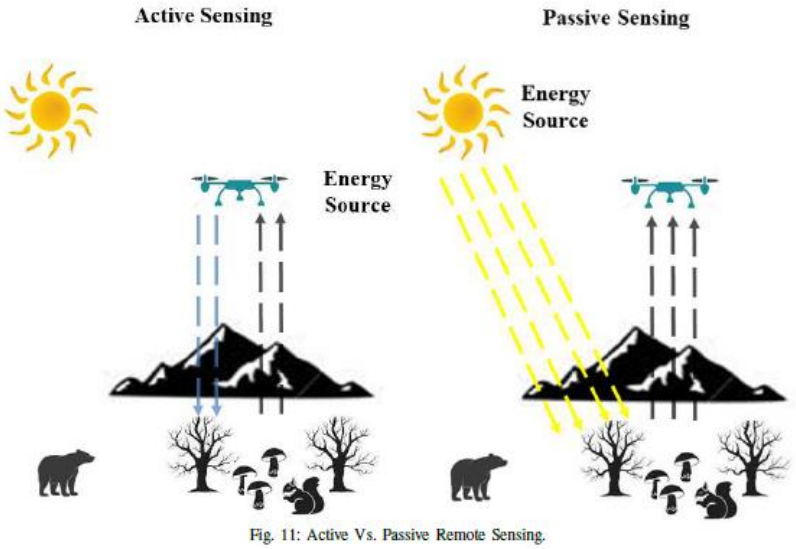


Figure 2.23: Active vs. Passive Remote Sensing
(Adapted from (Hazim, et al., 2019))

Table 2.5: UAV aerial sensing systems

Type of remote sensing	Operating Frequency	Type of sensor	Applications
Active	Microwave portion of the electromagnetic spectrum	Laser altimeter	It measures the height of a UAV with respect to the mean Earth's surface to determine the topography of the underlying surface.
		LiDAR	It determines the distance to the object by recording the time between transmitted and backscattered light pulses.
		Radar	It produces a two-dimensional image of the surface by recording the range and magnitude of the energy reflected from all objects.
		Ranging Instrument	It determines the distance between identical microwave instruments on a pair of platforms.
		Scatterometer	It derives maps of surface wind speed and direction by measuring backscattered radiation in the microwave spectral region.
		Sounder	It measures vertical distribution of precipitation, temperature, humidity, and cloud composition.
Passive	Visible, infrared, thermal infrared, and microwave portions of the electromagnetic spectrum	Accelerometer	It measures two general types of acceleration: 1) The translational acceleration (changes in linear motions); 2) The angular acceleration (changes in angular velocity per unit time).
		Hyperspectral radiometer	It discriminates between different targets based on their spectral response in each of the narrow bands.
		Imaging radiometer	It provides a two-dimensional array of pixels from which an image may be produced.
		Radiometer	It measures the intensity of

			electromagnetic radiation in some bands within the spectrum.
		Sounder	It measures vertical distributions of atmospheric parameters such as temperature, pressure, and composition from multispectral information.
		Spectrometer	It designs to detect, measure, and analyze the spectral content of incident electromagnetic radiation.
		Spectroradiometer	It measures the intensity of radiation in multiple wavelength bands. It designs for remotely sensing specific geophysical parameters.

2.4.3.3 Challenges of UAVs in Remote Sensing

i. Hostile Natural Environment:

UAVs can be utilized to study the atmospheric composition, air quality and climate parameters, because of their ability to access hazardous environments, such as thunderstorms, hurricanes and volcanic plumes (Austin, 2011). The researchers used UAVs for conducting environmental sampling and ocean surface temperature studies in the Arctic (Villa, et al., 2016). The authors in (Curry, et al., 2004) modify and test the Aerosonde UAV in extreme weather conditions, at very low temperatures (less than 20^{o}C) to ensure a safe flight in the Arctic. The aim of the work was to modify and integrate sensors on-board an Aerosonde UAV to improve the UAVs capability for its mission under extreme weather conditions such as in the Arctic. The steps to customize the UAV for the extreme weather conditions were:

- The avionics were isolated;
- A fuel-injection engine was utilised to avoid carburettor icing;
- A servo-system was used to induce ice breaking over the leading edge of the air-foil.

In Barrow, Alaska, the modified UAVs successfully demonstrated their capabilities to collect data for 48 hours along a 30 km^2 rectangular geographical area. When a UAV collects data from a volcano plume, a rotary wing UAV was particularly beneficial to hover inside the plume (McGonigle, et al., 2008).

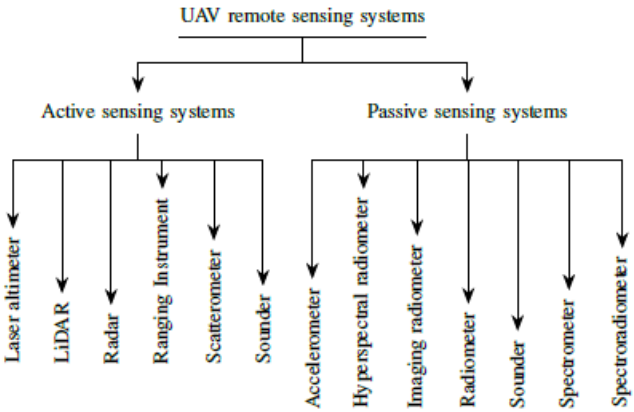


Figure 2.24: Classification of UAV Aerial Sensing Systems

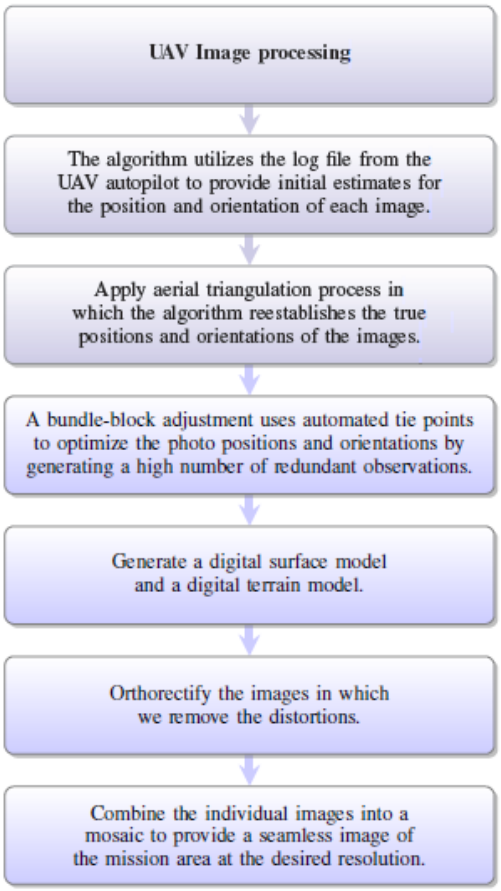


Figure 2.25: Image processing for a UAV remote sensing image

On the other hand, a fixed wing UAV was suitable to cover longer distances and higher altitudes to sense different atmospheric layers (Saggiani, et al., 2007). In (Lin & Lee, 2008), the authors presented a successful eye-penetration reconnaissance flight by Aerosonde UAV into Typhoon Longwang (2005). The 10 hours flight was split into four flight legs. In these

flight legs, the UAV measured the wind field and provided the tangential and radial wind profiles from the outer perimeter into the eye of the typhoon at the 700hPa layer. The UAV also took a vertical sounding in the eye of the typhoon and measured the strongest winds during the whole flight mission (Villa, et al., 2016).

ii. Camera Issues

The radiometric and geometric limitations imposed by the current generation of lightweight digital cameras are outstanding issues that need to be addressed. The current UAV digital cameras are designed for the general market and are not optimized for remote sensing applications. Current commercial instruments are mostly too bulky to be used with current lightweight UAVs. For the few that do exist, there is still a question of calibration process with conventional sensors. Spectral drawbacks include the fact that spectral response curves from cameras are usually poorly calibrated, which makes it difficult to convert brightness values to radiance. However, even cameras designed specifically for UAVs may not meet the required scientific benchmarks (Whitehead & Hugenholtz, 2014). Another setback is that the camera detectors may also become saturated when there are high contrasts, for instance when an image covers both a dark forest and a snow covered field. Another drawback is that many cameras are prone to vignette, where the centres of images appear brighter than the edges. This is due to the rays of light in the centres of the image having to pass through a less optical thickness of the camera lens. Thus they become more lowly attenuated than rays at the edges of the image. There are a number of techniques that can be taken into account to improve the quality of image:

1. micro-four-thirds cameras with fixed interchangeable lenses can be used instead of having a retractable lens, which allows for much improved calibrations and image quality;
2. A simple step that can make a big difference in the processing stage is to remove images that are blurred, under or overexposed, or saturated (Whitehead & Hugenholtz, 2014).
3. Illumination Issues: The shadows on a sunny day are clear and well defined. These weather conditions can cause critical problems for the automated image matching algorithms used in both triangulation process and digital elevation model generation (Whitehead & Hugenholtz, 2014). When clouds move rapidly, shaded areas can vary between images obtained during the same mission; therefore the aerial triangulation process will fail for some images, and also will result in errors for automatically generated digital elevation models. Moreover, the automated color balancing algorithms utilised in the creation of image mosaics may be affected by the patterns of light and shade across images. This can result in mosaics with poor visual quality. A similar generally observed illumination effect is the presence of image hotspots,

where a bright point appears in the image. Hotspots occur at the antisolar point due to the effects of bidirectional reflectance, which is dependent on the relative placement of the image sensor and the sun (Whitehead & Hugenholtz, 2014).

2.4.3.4. Research Trends in Utilisation of UAVs for Remote Sensing

i. Machine Learning

In remote sensing, the machine learning process begins with data collection using UAVs. The next step of machine learning is data cleansing, which includes cleansing up image and/or textual-based data and making the data manageable. This step sometimes might include reducing the number of variables associated with a record. The third step is selecting the right algorithm, which includes getting acquainted with the problem we are trying to solve. There are three famous algorithms being used in remote sensing:

- Random forest;
- Support vector machines;
- Artificial neural networks.

An algorithm is selected depending on the type of problem being solved. In some scenarios, where there are multiple features but limited records, support vector machines might work better. If there are a lot of records but fewer features, neural networks might yield better prediction/classification accuracy. Normally, several algorithms will be applied on a dataset and the one that works best is selected. A higher accuracy of the machine learning results can be achieved by using a combination of multiple algorithms, which is referred to as ensemble. Similarly, multiple ensembles will need to be applied on a dataset, in order to select the ensemble that works the best. It is practical to choose a subset of candidate algorithms based on the type of problem and then uses the narrowed down algorithms on a part of the dataset and see which one performs best. The first major challenge in machine learning is that the training segment of the dataset. This should have an unbiased representation of the whole dataset and should not be too little compared to the testing segment of the dataset. The second challenge is overfitting which can happen when the dataset that has been used for algorithm training is used for evaluating the model. This will result in very high prediction/classification accuracy. However, if a simple modification is performed, then the prediction/classification accuracy takes a dip (Python_Tips, 2017). The machine learning steps utilized by UAV remote sensing are shown in Figure 2.26.

ii. Combining Remote Sensing and Cloud Technology:

The use of digital maps in risk management as well as improving visualization of data and decision making process has become a standard for businesses and insurance companies. For instance, the insurance company can utilize UAV to generate a normalized difference vegetation index (NDVI) map in order to have an overview of the hail damage in corn. The

geographic information system (GIS) technology in the cloud use the NDVI map created from UAV images to provide an accurate and advanced tool for providing assistance with crop hail damage insurance settlements in minimal time and without conflict, while keeping expenses low (GisCloud, 2018).

iii. Free Space Optical

FSO technology over an UAV can be utilized in armed forces, where military wireless communications demand for secure transmission of information on the battlefield. Remote sensing UAVs can utilize this technology to disseminate large amount of images and videos to the fighting forces, mostly in a real time. UAVs that are Near Earth observing can be utilized to provide high resolution images of surface contours using synthetic aperture radar as well as light detection and ranging. Using FSO technology, aerial sensors can also transmit the collected data to the command center via on-board satellite communication sub-system (Kaushal & Kaddoum, 2017).

2.4.3.5. Future Insights in Utilization of UAVs for Remote Sensing

Some of the future possible directions for this application are:

i. Camera stabilization during flight (Whitehead & Hugenholtz, 2014) is one of the issues that need to be addressed, in the employment of UAV for remote sensing.
ii. Battery weight and charging time are critical issues that affect the duration of UAV missions (Madden, et al., 2015). The development and incorporation of lightweight solar powered components of UAV can improve the duration of UAV missions and hence it reduces the complexity of flight planning.
iii. The temporal digital surface models produced from aerial imagery using UAV as the platform, can become practical solution in mass balance studies for example mass balance of any particular metal in sanitary landfills, chloride in groundwater and sediment in a river. More specifically, the UAV-based mass balance in a debris covered glacier was estimated at high accuracy, when using high-resolution digital surface models differencing. Thus, the employment of UAV save time and money when compared with the traditional method of stake drilling into the glaciers which was labor-intensive and time consuming (Immerzeel, et al., 2014), (Bhardwaj, et al., 2016).
iv. The tracking methods utilized on high-resolution images can be practical to find an accurate surface velocity and glacier dynamics estimates. In (Sam, et al., 2016), the authors suggested a differential band method for estimating velocities of debris and non-debris parts of the glaciers. The on demand deployment of UAV to obtain high-resolution images of a glacier has resulted in efficient tracking methods when

compared with satellite imagery which depends on the satellite overpass (Bhardwaj, et al., 2016).

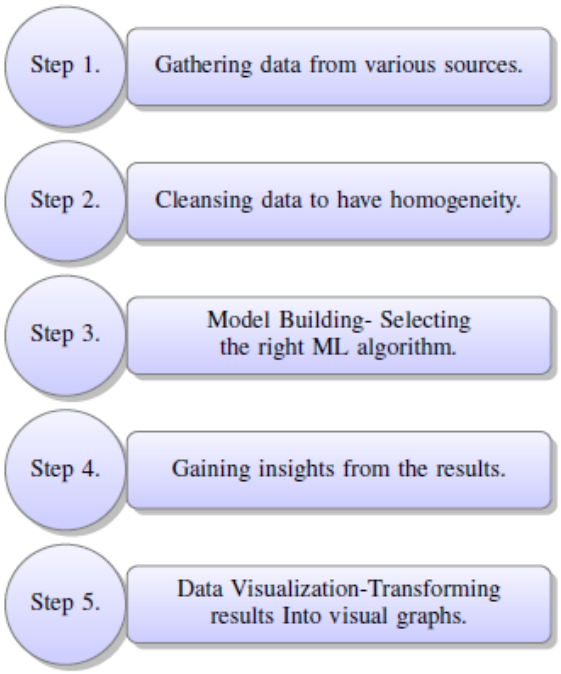


Figure 2.26: Machine learning in UAV remote sensing

v. UAV remote sensing can be as a powerful technique for field-based phenotyping with the advantages of high efficiency, low cost and suitability for complex environments. The adoption of multi-sensors coupled with advanced data analysis techniques for retrieving crop phenotypic traits have attracted great attention in recent years (Yang, et al., 2017).
vi. It is expected that with the advancement of UAVs with larger payload, longer flight time, low-cost sensors, improved image processing algorithms for Big data, and effective UAV regulations, there is potential for wider applications of the UAV-based field crop phenotyping (Yang, et al., 2017).
vii. UAV remote sensing for field-based crop phenotyping provides data at high resolutions, which is needed for accurate crop parameter estimations. The derivation of crop phenotypic traits based on the spectral reflection information using UAV as the platform has shown good accuracy under certain conditions. However, it showed a low accuracy in the research on the non-destructive acquisition of complex traits that were indirectly related to the spectral information (Yang, et al., 2017);
viii. Image processing of UAV imagery faces a number of challenges, such as variable scales, high amounts of overlap, variable image orientations, and high amounts of relief displacement arising from the low flying altitudes relative to the variation in

topographic relief (Whitehead & Hugenholtz, 2014). Researchers need to find efficient ways to overcome these challenges in future studies.

2.5 Homogenous Transformation Modelling Convention

2.5.1 Forward Kinematics

A manipulator comprises of serial links which are connected to each other's revolute or prismatic joints from the base frame through the end-effector. Forward kinematics refers to the calculation of the position and orientation of the end-effector in terms of the joint variables. A suitable kinematics model is needed in order to have forward kinematics for a robot mechanism in a systematic manner. The most common method for describing robot kinematics is the Denavit-Hartenberg method that uses four parameters. These parameters α_{i-1}, a_{i-1}, d_i and θ_i are the link twist, link length, link offset and joint angle, respectively. In order to determine the DH parameters, a coordinate frame is attached to each joint. Z_i axis of the coordinate frame points along the sliding or rotary direction of the joints. Figure 2.28 shows the coordinate frame assignment for a general manipulator.

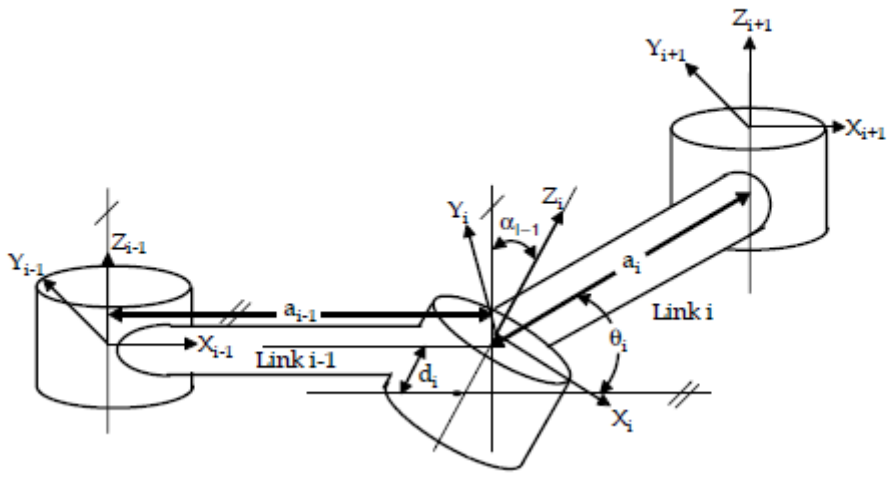


Figure 2.27: Coordinate frame assignment for a general manipulator

As shown in Figure 2.27, the distance from Z_{i-1} to Z_i measured along X_{i-1} is assigned as a_{i-1}, the angle between Z_{i-1} and Z_i measured along X_i is assigned as α_{i-1}, the distance from X_{i-1} to X_i measured along Z_i is assigned as d_i and the angle between X_{i-1} to X_i measured about Z_i is assigned as θ_i (Craig, 1989).

The general transformation matrix ${}^{i-1}_{i}T$ for a single link can be derived as follows;

$$
{}^{i-1}_{i}T = R_x(\alpha_{i-1})D_x(a_{i-1})R_z(\theta_i)Q_i(d_i)
$$

$$
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Equation 2.3

Where R_x and R_z represent rotation, D_x and Q_i represent translation, and $c\theta_i$ and $s\theta_i$ represent $\cos\theta_i$ and $\sin\theta_i$, respectively. The forward kinematics of the end-effector with respect to the base frame is derived by multiplying all of the ${}^{i-1}_{i}T$ matrices.

$$
{}^{base}_{end\_effector}T = {}^{0}_{1}T\ {}^{1}_{2}T \ldots {}^{n-1}_{n}T
$$

Equation 2.4

An alternative representation of ${}^{base}_{end\_effector}T$ can be written as

$$
{}^{base}_{end\_effector}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Equation 2.5

Where r_{kj}'s denote the rotational elements of transformation matrix (k and j=1, 2 and 3). p_x, p_y and p_z represent the elements of the position vector. For a manipulator with six joints, the position and orientation of the end-effector with respect to the base is determined by

$$
{}^{0}_{6}T = {}^{0}_{1}T(q_1){}^{1}_{2}T(q_2){}^{2}_{3}T(q_3){}^{3}_{4}T(q_4){}^{4}_{5}T(q_5){}^{5}_{6}T(q_6)
$$

Equation 2.6

Where q_i is the joint variable (revolute or prismatic joint) for joint i, (i=1, 2, ...6).

For an example, consider a 6-DOF manipulator (Stanford Manipulator) whose coordinate frame assignment and rigid body are illustrated in Figure 2.28. Note that the three axes of the manipulator's Euler wrist intersect at a common point. The last (RRR) and first three (RRP) joints are spherical in shape. R and P denote revolute and prismatic joints, respectively. The DH parameters corresponding to this manipulator are shown in Table 2.6.

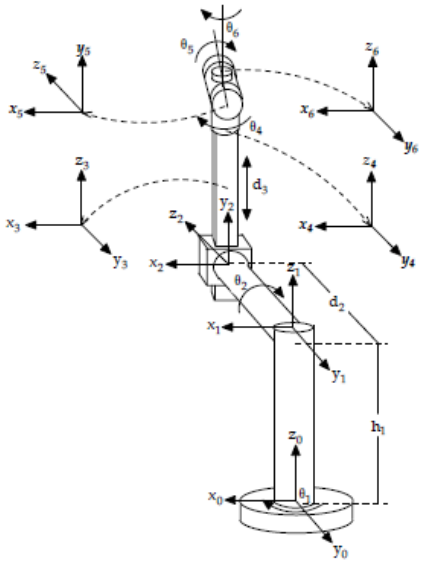


Figure 2.28: Rigid body and coordinate frame assignment for the Stanford Manipulator

Table 2.6: DH parameters for the Stanford Manipulator

I	**θ_i**	**α_{i-1}**	**a_{i-1}**	**d_i**
1	θ_1	0	0	h_1
2	θ_2	90	0	d_2
3	0	-90	0	d_3
4	θ_4	0	0	0
5	θ_5	90	0	0
6	θ_6	-90	0	0

It is straightforward to compute each of the link transformation matrices using equation 2.3, as follows.

$$ {}^{0}_{1}\mathrm{T} = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & h_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Equation 2.7

$$ {}^{1}_{2}\mathrm{T} = \begin{bmatrix} \mathrm{c}\theta_2 & -\mathrm{s}\theta_2 & 0 & 0 \\ 0 & 0 & -1 & -\mathrm{d}_2 \\ \mathrm{s}\theta_2 & \mathrm{c}\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Equation 2.8

$$ {}^{2}_{3}\mathrm{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & \mathrm{d}_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Equation 2.9

$$ {}^{3}_{4}\mathrm{T} = \begin{bmatrix} \mathrm{c}\theta_4 & -\mathrm{s}\theta_4 & 0 & 0 \\ \mathrm{s}\theta_4 & \mathrm{c}\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Equation 2.10

$$ {}^{4}_{5}\mathrm{T} = \begin{bmatrix} \mathrm{c}\theta_5 & -\mathrm{s}\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \mathrm{s}\theta_5 & \mathrm{c}\theta_5 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Equation 2.11

$$ {}^{5}_{6}\mathrm{T} = \begin{bmatrix} \mathrm{c}\theta_6 & -\mathrm{s}\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \mathrm{s}\theta_6 & \mathrm{c}\theta_6 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Equation 2.12

The forward kinematics of the Stanford Manipulator can be derived in the form of equation 3 multiplying all of the ${}^{\mathrm{i}-1}_{\mathrm{i}}\mathrm{T}$ matrices, where i=1,2, …, 6. In this case, ${}^{0}_{6}\mathrm{T}$ is given by

$$ {}^{0}_{6}\mathrm{T} = \begin{bmatrix} \mathrm{r}_{11} & \mathrm{r}_{12} & \mathrm{r}_{13} & \mathrm{p}_x \\ \mathrm{r}_{21} & \mathrm{r}_{22} & \mathrm{r}_{23} & \mathrm{p}_y \\ \mathrm{r}_{31} & \mathrm{r}_{32} & \mathrm{r}_{33} & \mathrm{p}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Equation 2.13

Where

$$r_{11} = -s\theta_6(c\theta_4 s\theta_1 + c\theta_1 c\theta_2 s\theta_4) - c\theta_6(c\theta_5(s\theta_1 s\theta_4 - c\theta_1 c\theta_2 c\theta_4) + c\theta_1 s\theta_2 s\theta_5)$$

$$r_{12} = s\theta_6(c\theta_5(s\theta_1 s\theta_4 - c\theta_1 c\theta_2 c\theta_4) + c\theta_1 s\theta_2 s\theta_5) - c\theta_6(c\theta_4 s\theta_1 + c\theta_1 c\theta_2 s\theta_4)$$

$$r_{13} = s\theta_5(s\theta_1 s\theta_4 - c\theta_1 c\theta_2 c\theta_4) - c\theta_1 c\theta_5 s\theta_2$$

$$r_{21} = s\theta_6(c\theta_1 c\theta_4 - c\theta_2 s\theta_1 s\theta_4) + c\theta_6(c\theta_5(c\theta_1 s\theta_4 + c\theta_2 c\theta_4 s\theta_1) - s\theta_1 s\theta_2 s\theta_5)$$

$$r_{22} = c\theta_6(c\theta_1 c\theta_4 - c\theta_2 s\theta_1 s\theta_4) + s\theta_6(c\theta_5(c\theta_1 s\theta_4 + c\theta_2 c\theta_4 s\theta_1) - s\theta_1 s\theta_2 s\theta_5)$$

$$r_{23} = -s\theta_5(c\theta_1 s\theta_4 + c\theta_2 c\theta_4 s\theta_1) - c\theta_5 s\theta_1 s\theta_2$$

$$r_{31} = c\theta_6(c\theta_2 s\theta_5 + c\theta_4 c\theta_5 s\theta_2) - s\theta_2 s\theta_4 s\theta_6$$

$$r_{32} = -s\theta_6(c\theta_2 s\theta_5 + c\theta_4 c\theta_5 s\theta_2) - c\theta_6 s\theta_2 s\theta_4$$

$$r_{33} = c\theta_2 c\theta_5 + c\theta_4 s\theta_2 s\theta_5$$

$$p_x = ds\theta_1 + d_4 c\theta_1 s\theta_2$$

$$p_y = -d_2 c\theta_1 + d_3 s\theta_1 s\theta_2$$

$$p_z = h_1 c\theta_1 + d_3 c\theta_2$$

2.5.2 Verification of Mathematical model

In order to check the accuracy of the mathematical model of the Stanford Manipulator shown in Figure 2.29, the following steps should be taken. The general position vector in equation 2.13 should be compared with the zero position vector in Figure 2.28.

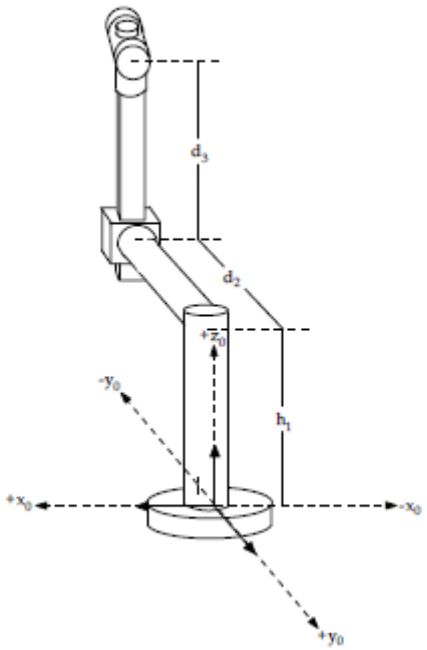


Figure 2.29: Zero position for the Stanford Manipulator

The general position vector of the Stanford Manipulator is given by

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} d_2 s\theta_1 - d_3 c\theta_1 s\theta_2 \\ -d_2 c\theta_1 - d_3 s\theta_1 s\theta_2 \\ h_1 + d_3 c\theta_2 \end{bmatrix}$$

Equation 2.14

In order to obtain the zero position in terms of link parameters, let's set $\theta_1=\theta_2=0°$ in equation 2.14.

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} d_2 s(0^\circ) - d_3 c(0^\circ) s(0^\circ) \\ -d_2 c(0^\circ) - d_3 s(0^\circ) s(0^\circ) \\ h_1 + d_3 c(0^\circ) \end{bmatrix} = \begin{bmatrix} 0 \\ -d_2 \\ h_1 + d_3 \end{bmatrix}$$

Equation 2.15

All coordinate frames in Figure 2.28 are neglected except the base which is the reference coordinate frame for deriving the link parameters in zero position as in Figure 2.29. Since there is not any link parameters observed in the direction of $+x_0$ and $-x_0$ in Figure 2.29, $p_x=0$. py equals $-d_2$ because there is only d_2 parameter in $-y_0$ direction. The parameters d_3 and h_1

are the $+z_0$ direction, so p_z equals h_1+d_3. In this scenario, the zero position vector of Stanford Manipulator are obtained as following

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} 0 \\ -d_2 \\ h_1 + d_3 \end{bmatrix}$$

Equation 2.16

It is explained above that the results of the position vector in equation 2.15 are identical to those obtained by equation 2.16. Hence, the mathematical model of the Stanford Manipulator is can be said to be driven correctly.

2.5.3 Inverse Kinematics

For many decades, the inverse kinematics problem of the serial manipulators has been studied. It is needed in the control of manipulators. Solving the inverse kinematics requires lots of computationally resources and generally uses up much time in the real time control of manipulators. Actuators work in joint space whereas tasks to be performed by a manipulator are in the Cartesian space. Cartesian space includes position vector and orientation matrix. However, joint angles are used to represent joint space. The conversion of the orientation and position of a manipulator end-effector from Cartesian space to joint space is referred to as inverse kinematics problem. There are two approaches to the solutions namely, geometric and algebraic. Algebraic solution is used for deriving the inverse kinematics solution analytically.

2.5.3.1 Geometric Solution Approach

In the Geometric solution approach, the spatial geometry of the manipulator is decomposed into several plane geometry problems. It is then applied to the simple robot structures, such as, 2-DOF planer manipulator whose joints are both revolute. The link lengths are l_1 and l_2 shown in Figure 2.28. Consider Figure 2.29 in order to determine the kinematics equations for the planar manipulator. The components of the point P (p_x and p_y) are derived as follows.

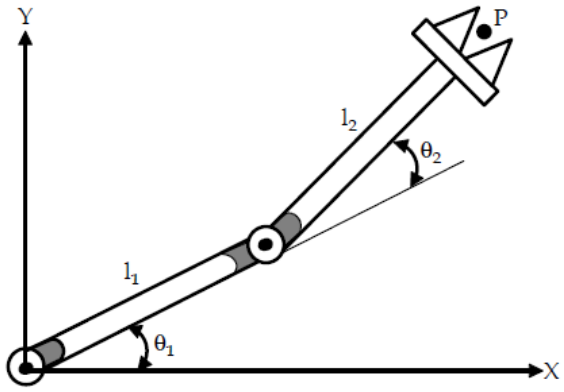


Figure 2.30: Planer manipulator

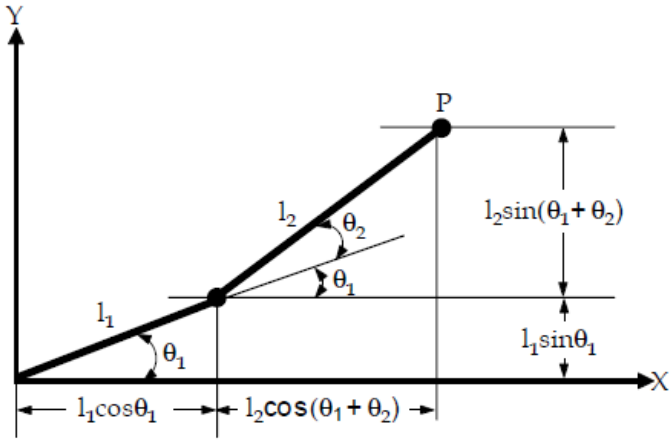


Figure 2.31s: Solving the inverse kinematics based on trigonometry

$P_x = l_1c\theta_1 + l_2c\theta_{12}$

Equation 2.17

$P_y = l_1s\theta_1 + l_2s\theta_{12}$

Equation 2.18

Where $c\theta_{12} = c\theta_1c\theta_2 - s\theta_1s\theta_{12}$ and $s\theta_{12} = s\theta_1c\theta_2 + c\theta_1s\theta_2$. The solution of θ_2 can be computed from summation of squaring both equations 15 and 16.

$$P_x^2 = I_1^2c^2\theta_1 + I_2^2c^2\theta_{12} + 2I_1I_2c\theta_1c\theta_{12}$$
$$P_y^2 = I_1^2s^2\theta_1 + I_2^2s^2\theta_{12} + 2I_1I_2s\theta_1s\theta_{12}$$
$$P_x^2 + P_y^2 = I_1^2(c^2\theta_1 + s^2\theta_2) + I_2^2(c^2\theta_{12} + s^2\theta_{12}) + 2I_1I_2(c\theta_1c\theta_{12} + s\theta_1s\theta_{12})$$

Since

$c^2\theta_1 + s^2\theta_2 = 1$, the equation given above is simplified as follows;

$$P_x^2 + P_y^2 = I_1^2 + I_2^2 + 2I_1I_2(c\theta_1[c\theta_1 c\theta_2 - s\theta_1 s\theta_2] + s\theta_1[s\theta_1 c\theta_2 + c\theta_1 s\theta_2])$$

$$P_x^2 + P_y^2 = I_1^2 + I_2^2 + 2I_1I_2(c^2\theta_1 c\theta_2 - c\theta_1 s\theta_1 s\theta_2 + s^2\theta_1 c\theta_2 + c\theta_1 s\theta_1 s\theta_2$$

$$P_x^2 + P_y^2 = I_1^2 + I_2^2 + 2I_1I_2(c\theta_2[c^2\theta_1 + s^2\theta_1])$$

$$P_x^2 + P_y^2 = I_1^2 + I_2^2 + 2I_1I_2 c\theta_2$$

Therefore

$$c\theta_2 = \frac{P_x^2 + P_y^2 - I_1^2 - I_2^2}{2I_1I_2}$$

Equation 2.19

Since, $c^2\theta_i + s^2\theta_i = 1$ $(i = 1,2,3, \dots\dots\dots)$, $s\theta_2$ is obtained as

$$s\theta_2 = \pm\sqrt{1 - \left(\frac{P_x^2 + P_y^2 - I_1^2 - I_2^2}{2I_1I_2}\right)^2}$$

Equation 2.20

Finally, two possible solutions for θ_2 can be written as

$$\theta_2 = \text{Atan2}\left(\pm\sqrt{1 - \left(\frac{P_x^2 + P_y^2 - I_1^2 - I_2^2}{2I_1I_2}\right)^2}, \frac{P_x^2 + P_y^2 - I_1^2 - I_2^2}{2I_1I_2}\right)$$

Equation 2.21

Let's first, multiply each side of equation 2.17 by $c\theta_1$ and equation 2.18 by $s\theta_1$ and add the resulting equations in order to find the solution of θ_1 in terms of link parameters and the known variable θ_2.

$$c\theta_1 p_x = I_1 c^2\theta_1 + I_2 c^2\theta_1 c\theta_2 - I_1 c\theta_1 s\theta_1 s\theta_2$$

$$s\theta_1 p_y = I_1 s^2\theta_1 + I_2 s^2\theta_1 c\theta_2 - I_2 s\theta_1 c\theta_1 s\theta_2$$

$$c\theta_1 p_x + s\theta_1 p_y = I_1(c^2\theta_1 + s^2\theta_1) + I_2 c^2\theta_2(c^2\theta_1 + s^2\theta_1)$$

The simplified equation obtained as follows;

$$c\theta_1 p_x + s\theta_1 p_y = I_1 + I_2 c\theta_2$$

Equation 2.22

In this step, multiply both sides of equation 2.17 by $-s\theta_1$ and equation 2.18 by $c\theta_1$ and then adding the resulting equations produce

$$-s\theta_1 p_x = -I_1 s\theta_1 c\theta_1 - I_2 s\theta_1 c\theta_1 c\theta_2 + I_2 s^2\theta_1 s\theta_2$$
$$c\theta_1 p_y = I_1 s\theta_1 c\theta_1 + I_2 c\theta_1 s\theta_1 c\theta_2 + I_2 c^2\theta_1 s\theta_2$$
$$-s\theta_1 p_x + c\theta_1 p_y = I_2 s\theta_2 (c^2\theta_1 + s^2\theta_1)$$

The simplified equation is given by

$$-s\theta_1 p_x + c\theta_1 p_y = I_2 s\theta_2$$

Equation 2.23

Now, multiply each side of equation 2.22 by P_x and equation 2.23 by P_y and add the resulting equations in order to obtain $c\theta_1$.

$$c\theta_1 p_x^2 + s\theta_1 p_x p_y = p_x(I_1 + I_2 c\theta_2)$$
$$-s\theta_1 p_x p_y + c\theta_1 p_y^2 = p_y I_2 s\theta_2$$
$$c\theta_1 \left(p_x^2 + p_y^2\right) = p_x(I_1 + I_2 c\theta_2) + p_y I_2 s\theta_2$$

Therefore

$$c\theta_1 = \frac{p_x(I_1 + I_2 c\theta_2) + p_y I_2 s\theta_2}{p_x^2 + p_y^2}$$

Equation 2.24

$s\theta_1$ is obtained as

$$s\theta_1 = \pm\sqrt{1 - \left(\frac{p_x(I_1 + I_2 c\theta_2) + p_y I_2 s\theta_2}{p_x^2 + p_y^2}\right)},$$

Equation 2.25

As a result, two possible solutions for θ_1 can be written

$$\theta_1 = \text{Atan2}\left(\pm\sqrt{1 - \left(\frac{p_x(I_1 + I_2 c\theta_2) + p_y I_2 s\theta_2}{p_x^2 + p_y^2}\right)^2}, \frac{p_x(I_1 + I_2 c\theta_2) + p_y I_2 s\theta_2}{p_x^2 + p_y^2}\right)$$

Equation 2.26

As seen above, although the planar manipulator has a simple structure, its inverse kinematics solution based on geometric approach is very cumbersome.

2.5.3.2 Algebraic Solution Approach

For the manipulators with more links and whose arm extends into 3 dimensions the geometry gets much more tedious. Hence, algebraic approach is chosen for the inverse kinematics solution. Recall the equation 2.6 to find the inverse kinematics solution for a six-axis manipulator.

$${}^0_6T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^0_1T(q_1){}^1_2T(q_2){}^2_3T(q_3){}^3_4T(q_4){}^4_5T(q_5){}^5_6T(q_6)$$

To find the inverse kinematics solution for the first joint (q_1) as a function of the known elements of ${}^{base}_{end\_effector}T$, the link transformation inverses are premultiplied as follows.

$$[{}^0_1T(q_1)]^{-1}{}^0_6T = [{}^0_1T(q_1)]^{-1}{}^0_1T(q_1){}^1_2T(q_2){}^2_3T(q_3){}^3_4T(q_4){}^4_5T(q_5){}^5_6T(q_6)$$

Where $[{}^0_1T\ (q_1)]^{-1}\ {}^0_1T\ (q_1) = I$, I is identity matrix. In this case the above equation is given by

$$[{}^0_1T(q_1)]^{-1}{}^0_6T = {}^1_2T(q_2){}^2_3T(q_3){}^3_4T(q_4){}^4_5T(q_5){}^5_6T(q_6)$$

Equation 2.27

To find the other variables, the following equations are obtained as a similar manner.

$$[{}_1^0\mathrm{T}(q_1){}_2^1\mathrm{T}(q_2)]^{-1}{}_6^0\mathrm{T} = {}_3^2\mathrm{T}(q_3){}_4^3\mathrm{T}(q_4){}_5^4\mathrm{T}(q_5){}_6^5\mathrm{T}(q_6)$$

Equation 2.28

$$[{}_1^0\mathrm{T}(q_1){}_2^1\mathrm{T}(q_2){}_3^2\mathrm{T}(q_3)]^{-1}{}_6^0\mathrm{T} = {}_4^3\mathrm{T}(q_4){}_5^4\mathrm{T}(q_5){}_6^5\mathrm{T}(q_6)$$

Equation 2.29

$$[{}_1^0\mathrm{T}(q_1){}_2^1\mathrm{T}(q_2){}_3^2\mathrm{T}(q_3){}_4^3\mathrm{T}(q_4)]^{-1}{}_6^0\mathrm{T} = {}_5^4\mathrm{T}(q_5){}_6^5\mathrm{T}(q_6)$$

Equation 2.30

$$[{}_1^0\mathrm{T}(q_1){}_2^1\mathrm{T}(q_2){}_3^2\mathrm{T}(q_3){}_4^3\mathrm{T}(q_4){}_5^4\mathrm{T}(q_5)]^{-1}{}_6^0\mathrm{T} = {}_6^5\mathrm{T}(q_6)$$

Equation 2.31

There are twelve simultaneous set of nonlinear equations to be solved. The only unknown variable on the LHS of equation 18 is q_1. The twelve nonlinear matrix elements of right hand side are constants, zero or functions of q_2 through q_6. If the elements on the LHS which are the function of q_1 are equated with the elements on the RHS, then the joint variable q_1 can be solved as functions of the fixed link parameters and r_{11}, r_{12},… r_{33}, P_x, P_y, P_z. When q_1 is determined, then the other joint variables are solved by the same way as before. It is not compulsory that the first equation will produce q_1 and the second q_2 etc. To find a suitable equation for the solution of the inverse kinematics problem, any equation defined above (equations 2.27-2.31) can be arbitrarily used. Some trigonometric equations used in the solution of the inverse kinematics problem are given in Table 2.7.

Table 2.7: Some trigonometric equations and solutions used in inverse kinematics

S/N	Equations	solutions
1	a sin θ + b cos θ = c	$\theta = \mathrm{Atan2}(a, b) \pm \mathrm{Atan2}\left(\sqrt{a^2 + b^2 - c^2}, c\right)$
2	a sin θ + b cos θ = 0	$\theta = \mathrm{Atan2}(-b, a)$ or $\theta = \mathrm{A\,tan2}(b, -a)$
3	Cos θ = a and sin θ = b	$\theta = \mathrm{Atan2}(b, a)$
4	Cos θ = a	$\theta = \mathrm{Atan2}\left(\pm\sqrt{1 - a^2}, a\right)$
5	Sin θ = a	$\theta = \mathrm{Atan2}\left(a, \pm\sqrt{1 - a^2}\right)$

CHAPTER THREE: METHODOLOGY

This chapter begins by laying out the docking and undocking concept as well as explaining it. The next section describes the handshake docking algorithm. The model frame work follows next and the docking system demonstration is explained. Afterwards, the hardware and software used are displayed and mentioned respectively. The software and programming section which follows shows the 7 step object detection and tracking algorithm for the mother drone as well as the LabVIEW subVIs for autonomous movement. Finally, the python based tracking program for the micro drone is presented.

3.1 Docking system concept

The docking and undocking system fits at the top part of the guardian 5 VTOL as illustrated in Figure 3.1. This position was considered primarily because of the space configuration on the mother drone. The arm was designed to extend during docking and undocking as illustrated in Figure 3.2. This will improve stability of the micro drone by minimising the effect of the turbulence created by the propellers of the mother drone in hover mode. The micro drone needs to be as far away from the mother drone propeller as possible to ensure that it does not get blown away or overly affected during docking or undocking. The arm system makes use of a rack and pinion mechanism for its extension and retraction. A bevel gear system is also used to control the grippers on the arm as illustrated in Figure 3.4. The grippers serve to hold the micro drone in position during docking and to release it during undocking.

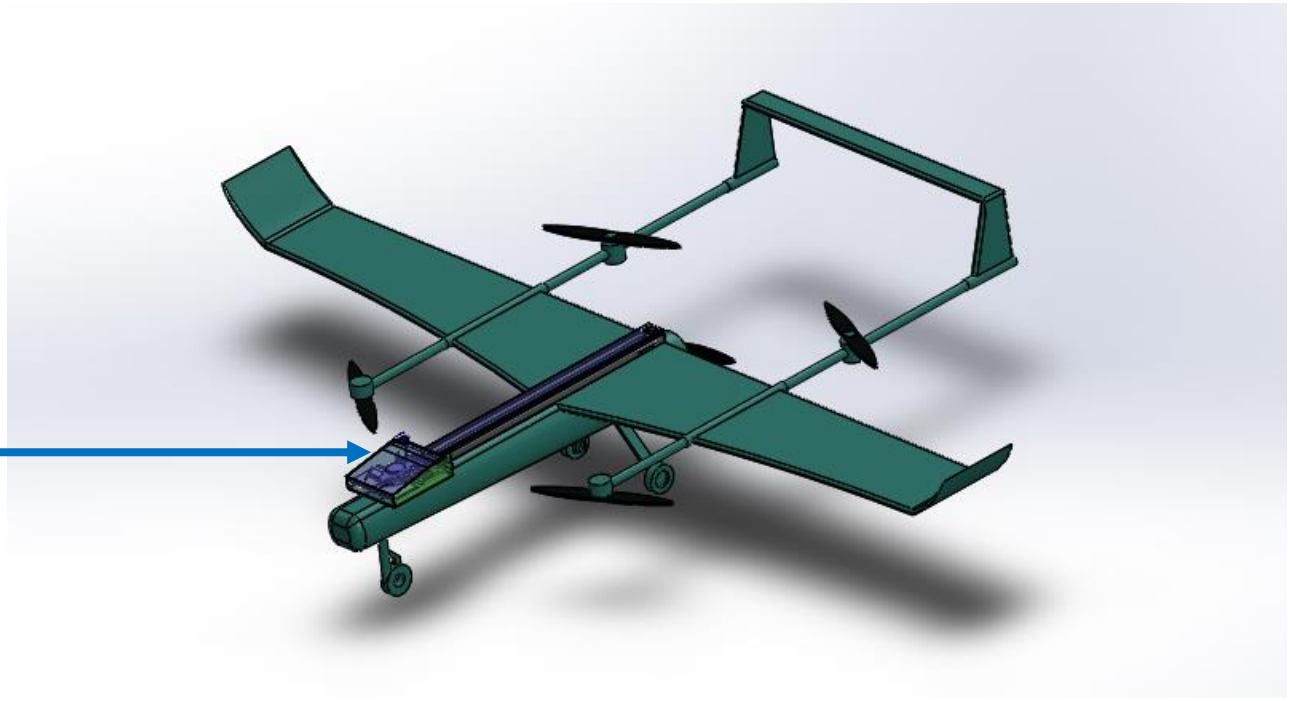


Figure 3.1: Guardian 5 docking concept

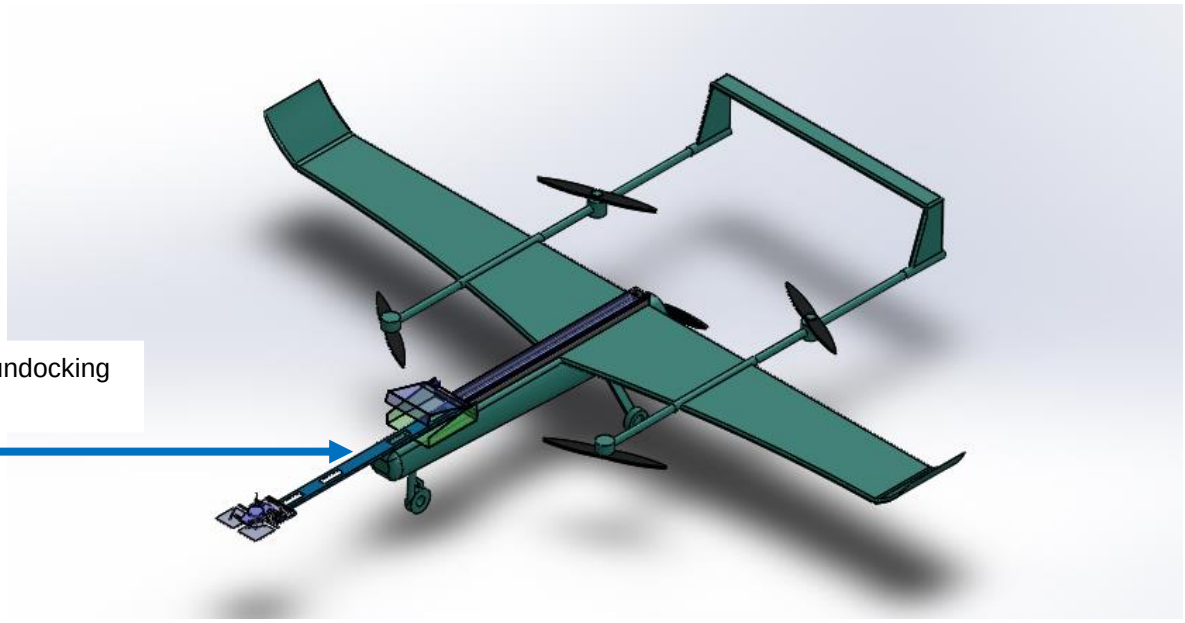


Figure 3.2: Guardian 5 docking concept (during undocking)

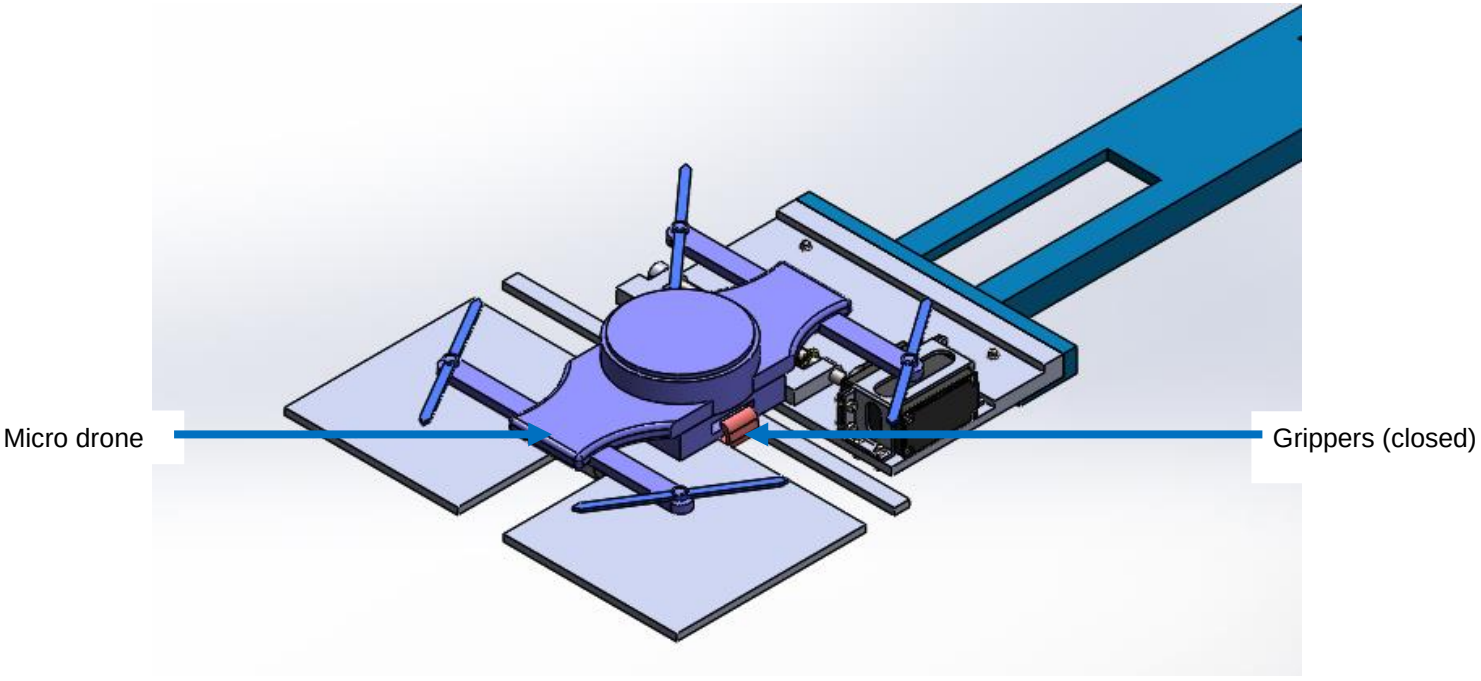


Figure 3.3: Guardian 5 docking concept (extended docking arm with micro drone)

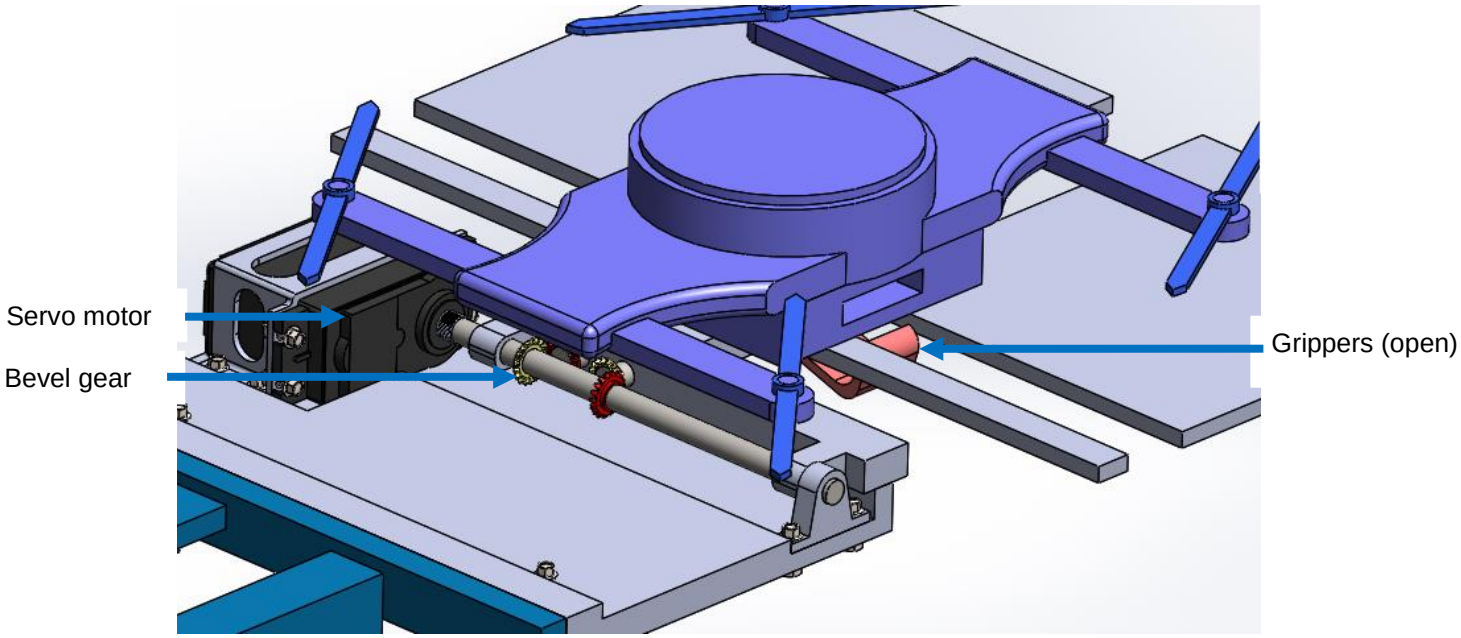


Figure 3.4: Guardian 5 concept showing bevel gears controlling grippers

3.2 Handshake Docking Algorithm

The docking algorithm is derived from the way humans give and receive handshakes. In a typical handshake between two humans, the following steps take place:

1. At a particular instance of time (t), the giver and the receiver assume a specific approximate location and orientation where there arms can possibly make contact.
2. At a particular instance of time(t), the giver makes a gesture for the handshake
3. The receiver, having seen the gesture, guides his arm hand towards the hand of the giver.
4. A handshake takes place.

In the airborne docking case, the micro drone plays the role of the giver while the mother drone plays the role of the receiver. The algorithm follows the following steps:

1. At a particular instance of time (t), the mother drone and micro drone assume a specific approximate location and orientation where docking can possibly take place. This done by moving the micro drone and mother drone to approximate location then initialising a line of sight tracking algorithm. The tracking algorithm is implemented via cameras on both platforms (micro drone and mother drone). It ensures that the mother drone is in alignment with the micro drone and vice versa. A yellow ball is placed on the micro drone and the mother drone as tracking objects in this particular setup.
2. At a particular instance of time (t), the micro drone gives a signal to the mother drone by flying upwards for 40cm.
3. The mother drone, having detected the gesture from the micro drone, moves to a docking position and extends it arm towards the micro drone. The arm extension is carried out very precisely so that the micro drone can land accurately on the platform. The Camera on the mother drone works with an IR sensor on the arm to ensure precise movement of the arm.
4. Docking takes places as the micro drone lands on the arm. The Camera on the mother drone tracks the coordinates of the micro drone in order to detect when it has successfully landed. After which the grippers grab the micro drone and the arm is retracted.

3.3 Model framework

In order to actualise the hardware, software and programming algorithms required to achieve an autonomous airborne docking and undocking system, a more simplified system was designed. The system comprised of a micro drone (Dji-Tello) and mother drone (ground station) built from Pitsco Education Tetrix robotics kit and NI myRIO micro controller. The Dji Tello has a Python based Software Development Kit (SDK), which was used to develop customised programs in Python programming language to control the drone. Since Python has many libraries for machine learning and Artificial intelligence applications, we found it suitable for developing machine vision algorithms such as object detection and tracking algorithms for the docking system. Furthermore Python has a very large and active online community with many repositories on GitHub. Python OpenCV Library was used extensively to develop the algorithms that control the drone. The ground station control algorithms which simulate the Mother drone was developed using LabVIEW. LabVIEW has various VIs for Machine vision algorithms such as Vision Acquisition and Image Processing. These VIs can be deployed on NI myRIO to achieve decent object detection and tracking functionalities.
In order to demonstrate the docking algorithm, a flight mission path was created as shown in figure 3.4

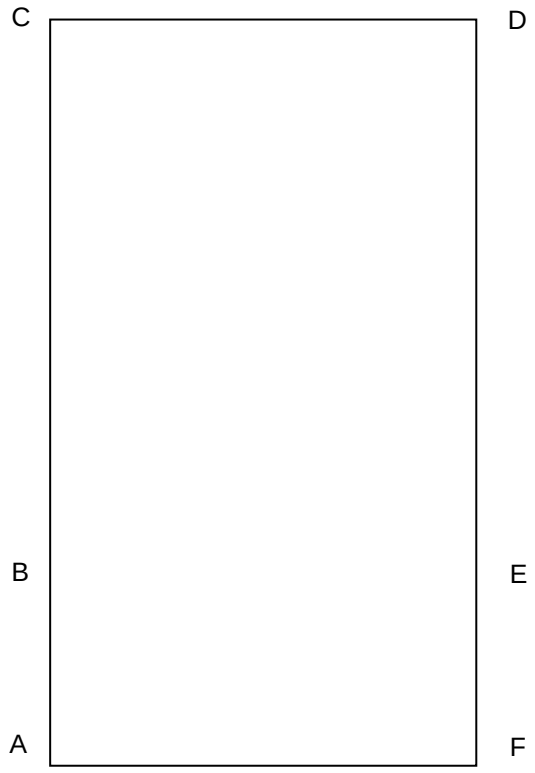


Figure 3.5: Flight Path

The docking system demonstration autonomously implemented the following steps;

1. The mother drone starts off from F with the micro drone docked.
2. The mother drone moves from F to A and then B
3. At B, the mother drones prepares to undock by extending the docking arm and the micro drone takes off. Immediately after the take-off, the docking arm retracts to its initial position.
4. The mother drone heads back to t F while the micro drone flies from B to C and then to D.
5. At D the micro drone drops its altitude and prepares for docking. While the Mother drone moves from F to E and initiates a search mode in preparation for docking.
6. Micro drone flies from D to E and initiates tracking algorithm. The mother drone detects the micro drone and also initiates tracking algorithm.
7. At a particular instance of time (t), the micro drone gives a signal by flying upwards and the mother drone extends it arm for docking.
8. The micro drone lands and is grabbed by the docking arm after which the arm retracts to its original position.
9. The Mother drone moves from E to F with the micro drone docked.

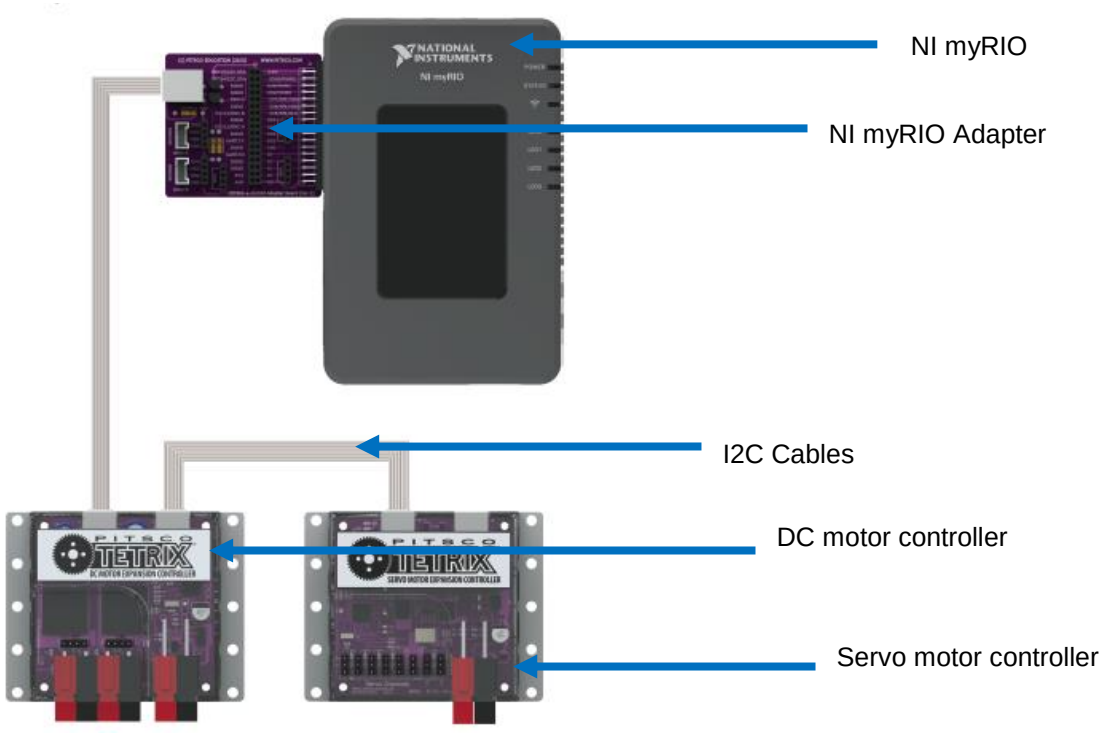


Figure 3.6: Basic connection diagram for ground station

Servo port 1: Camera swivel

Servo port 2: Gripper

Servo port 3: Side movements for ground station

Servo port 4: Arm extension (rack and pinion mechanism)

Left DC motor: Motor 2

Right DC motor: Motor 1

DI03: Initiate button

AI0: IR sensor

Figure 3.7: DJI Tello drone

The DJI Tello drone dimension is 98mm x 92.5mm x 41mm

Figure 3.8: DJI tello drone without propeller guards
Propeller guards were taken off to reduce the instability while hovering at low heights due to ground effect.

Figure 3.9: DJI Tello with batteries removed to show how it fits

Figure 3.10: DJI Tello with yellow tracking ball attached

The yellow ball was chosen to make the tracking system more consistent and accurate. The spherical shape remains consistent irrespective of the orientation of the drone during flight. The yellow colour is also relatively easy to track.

Figure 3.11: DJI Tello with ball attached, top view

The ball was attached in a way that the propellers can still rotate freely

Figure 3.12: DJI Tello with ball attached, side view

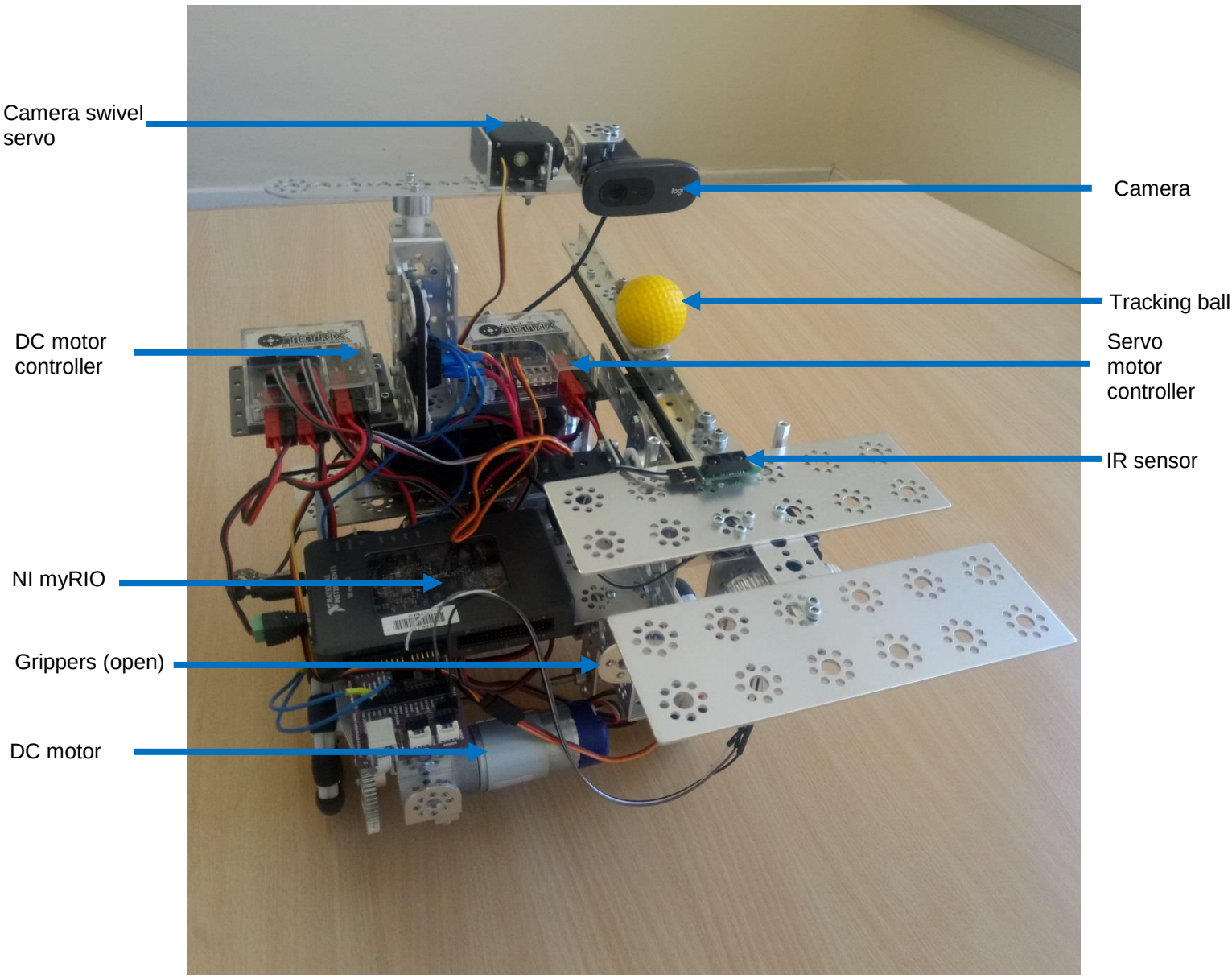


Figure 3.13: Ground station

The ground station is 440mm x 340mm x 350mm when the arm is retracted and 500mm x 340mm x 350mm when the arm is extended

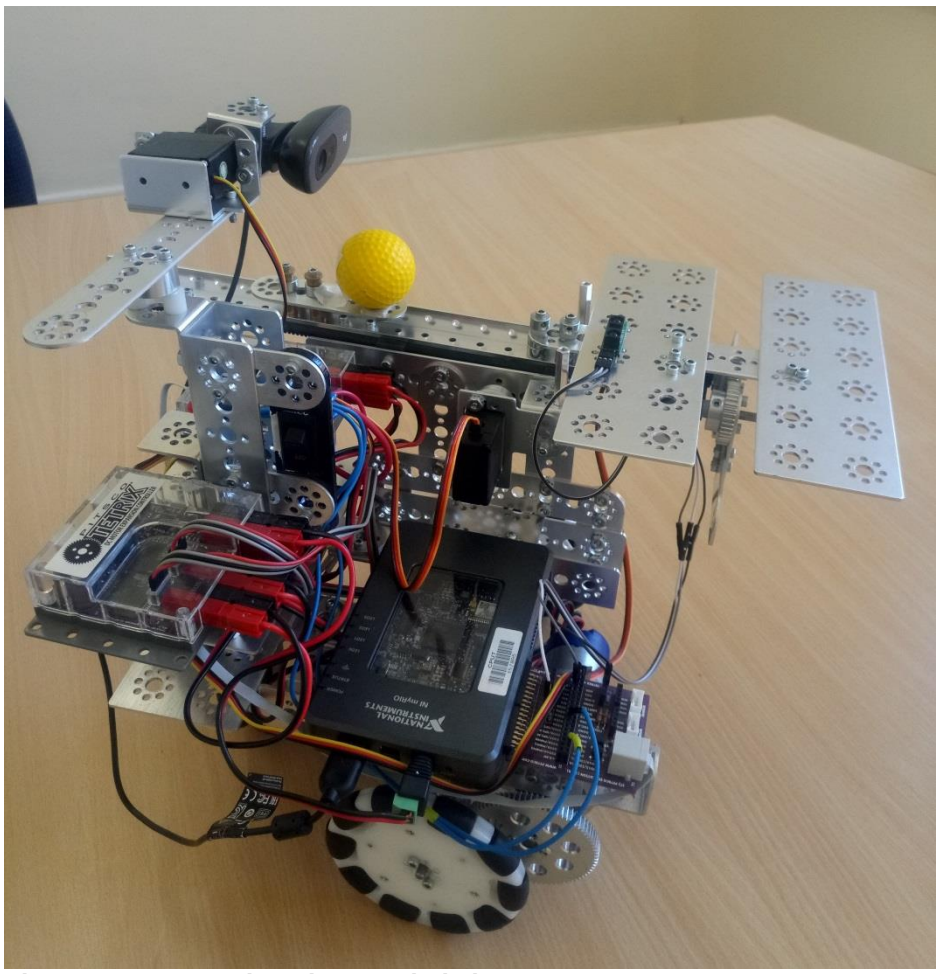


Figure 3.14: ground station, angled view

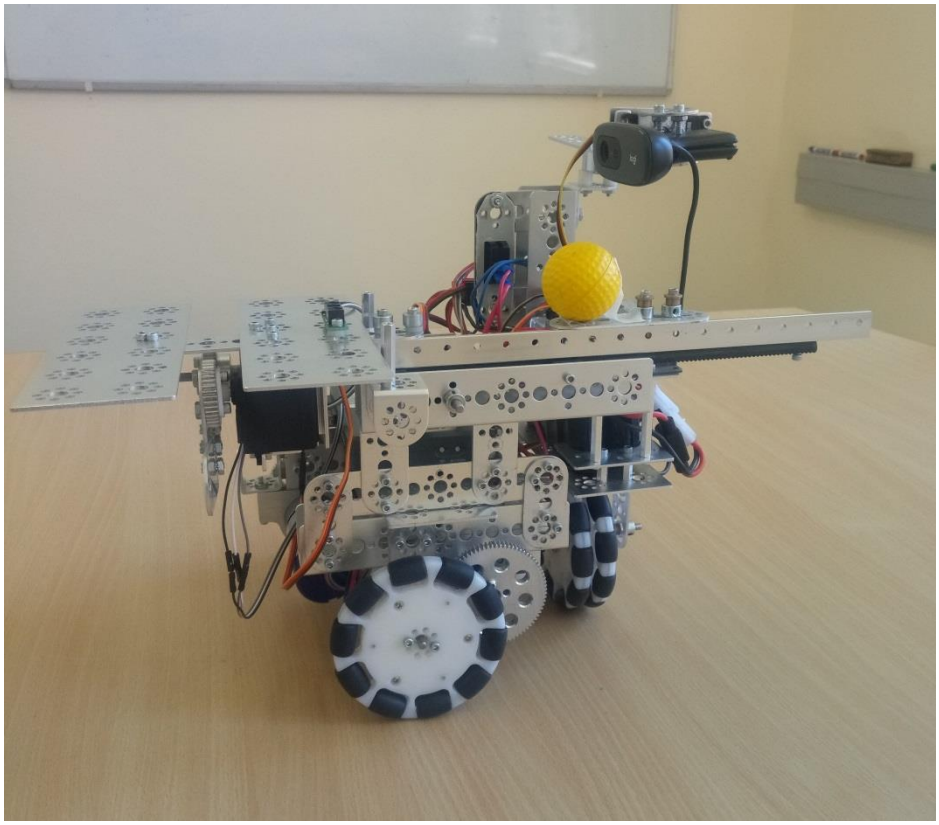


Figure 3.15: ground station left side view, retracted arm

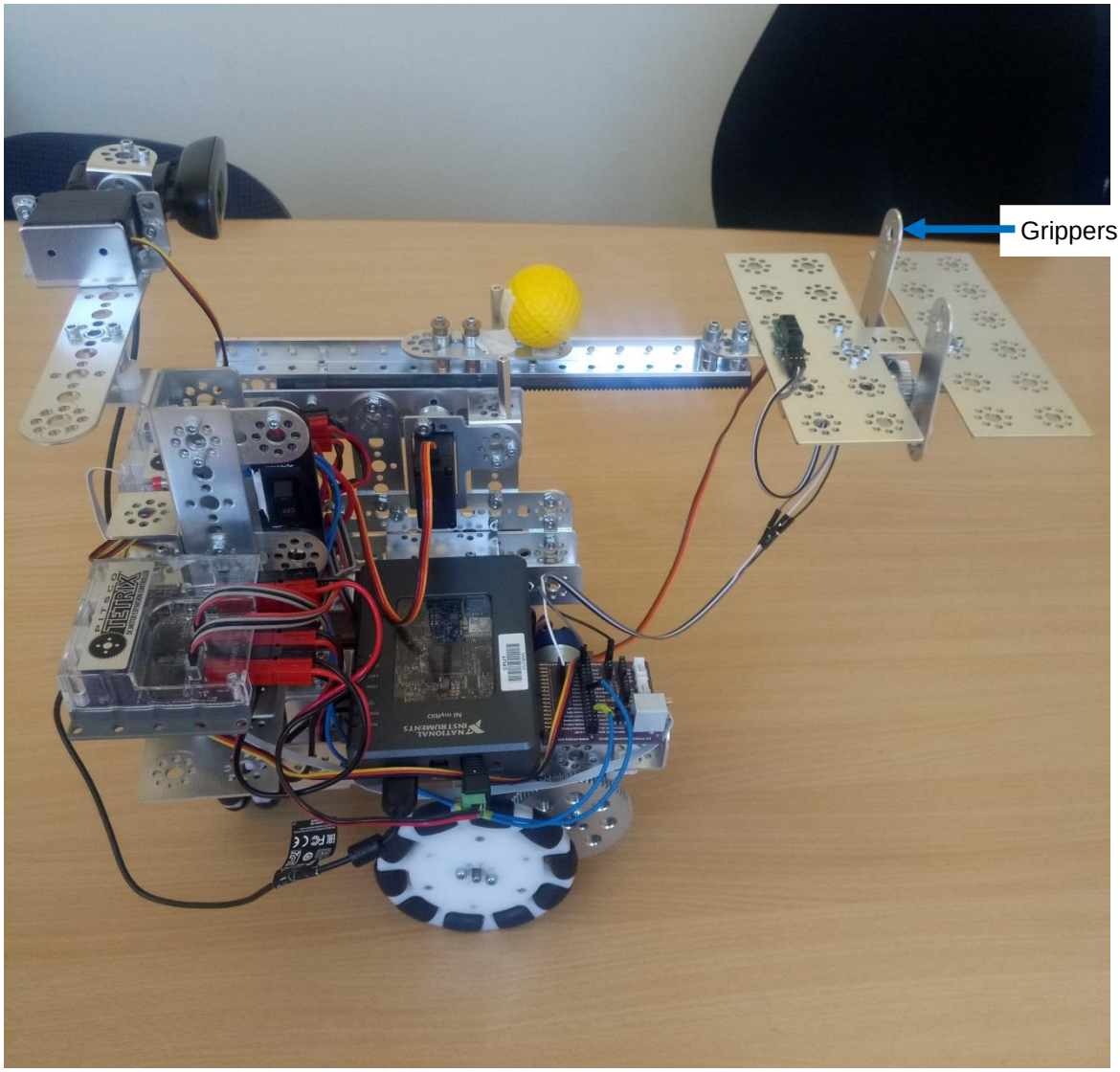


Figure 3.16: Ground station, right side view with extended docking arm and closed grippers

3.4 Software and Programming

The Ground station was programmed in LabVIEW, while the micro drone was programmed in Python.

3.4.1 Ground Station Object Detection

The ground station object detection algorithm follows a 7 step image processing algorithm

Step 1: The original image is fed into the Vision Assistant platform

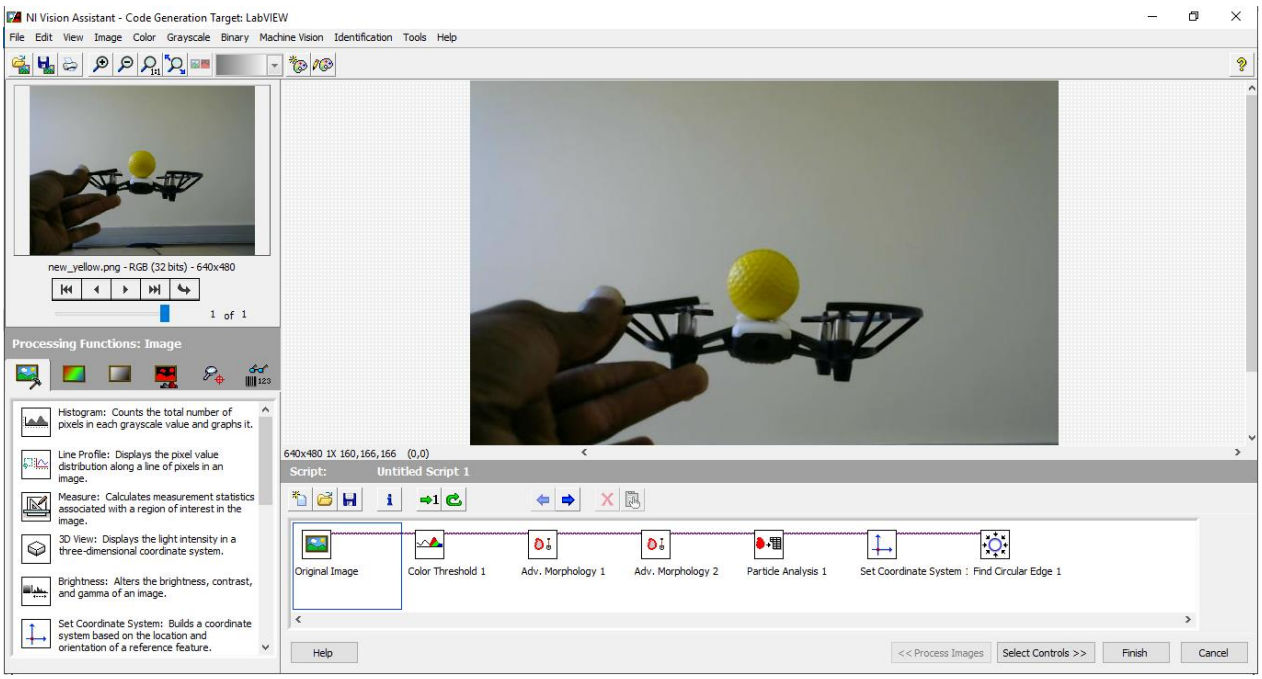


Figure 3.17: Original image

Step 2: A colour threshold setup is done to extract only yellow colour from the original image. The RGB colour threshold setup has the following values:

R_{min} =98 R_{max}=245 G_{min}=66 G_{max}=241 B_{min}=0 B_{max}= 69

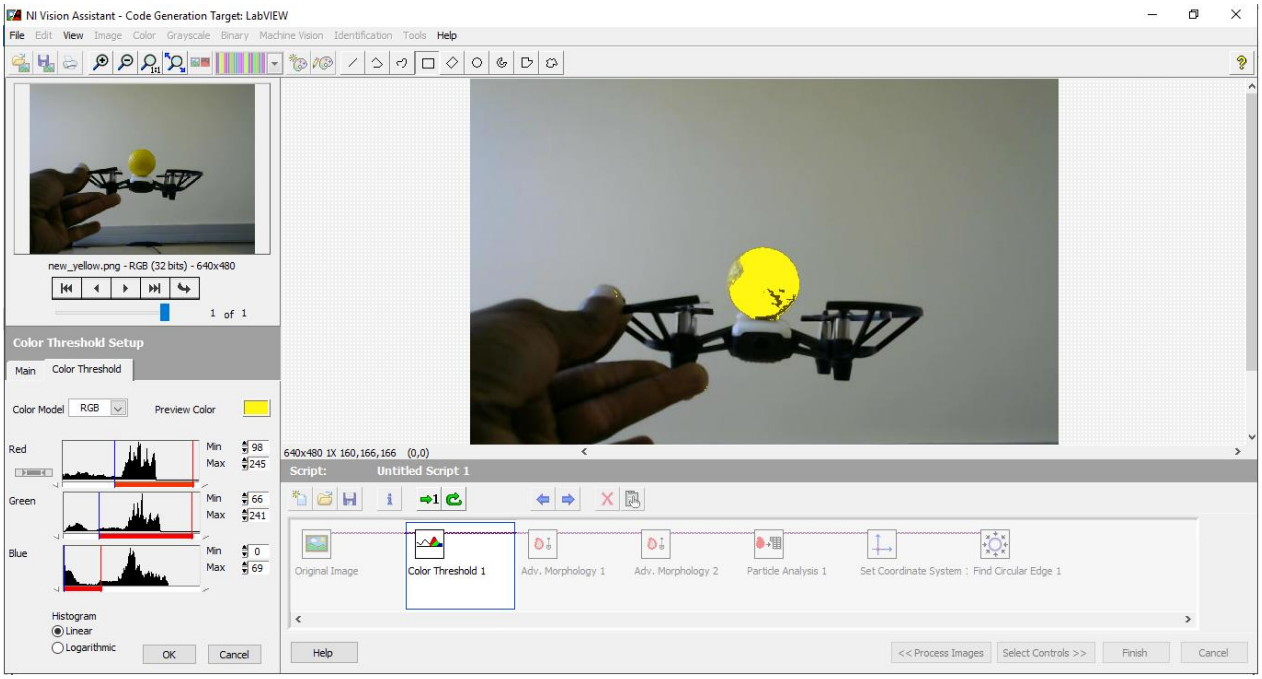


Figure 3.18: Colour threshold extraction

Step3: Advanced Morphology setting is used to remove small objects in order to leave behind only one mass

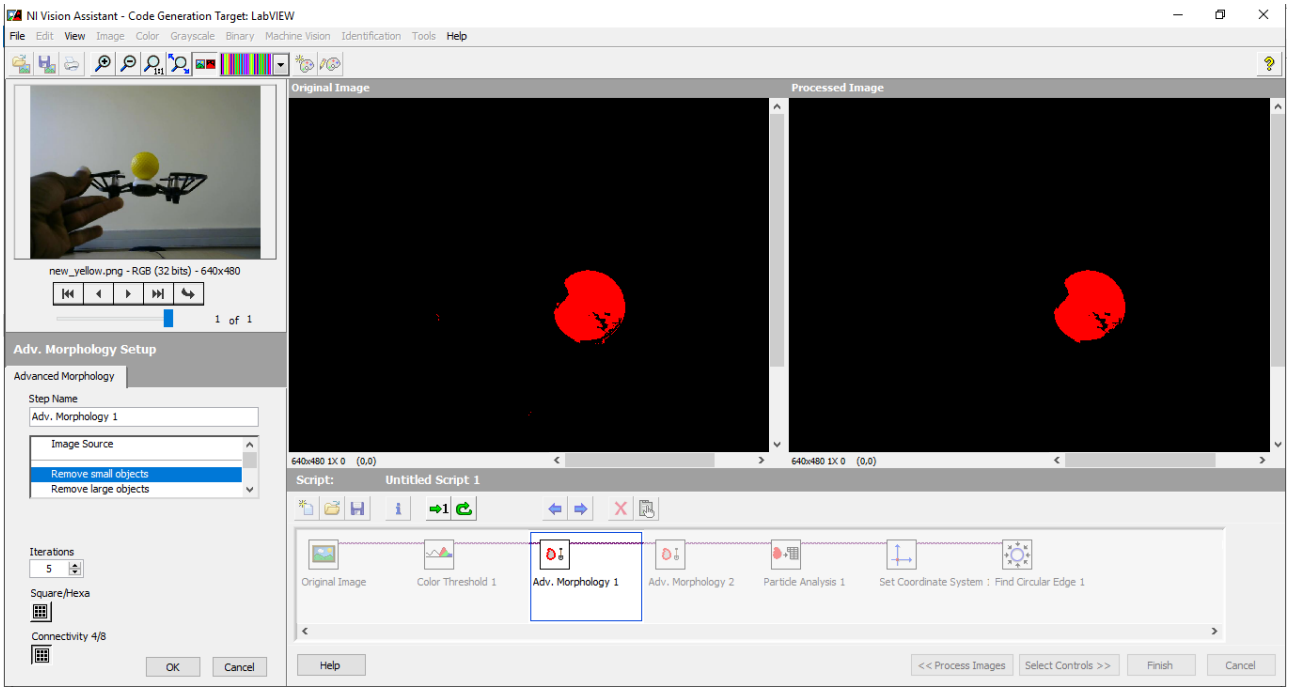


Figure 3.19: Remove small objects

Step 4: Advanced morphology setting is used to fill up holes

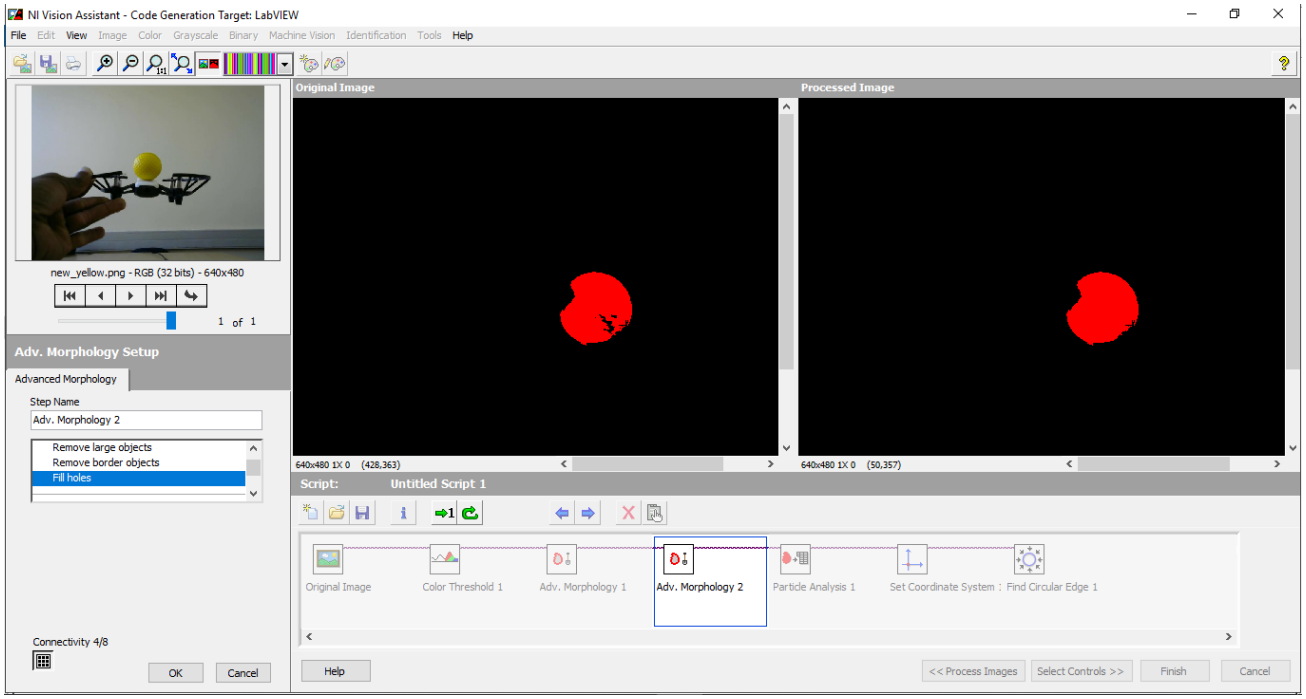


Figure 3.20: Fill holes

Step 5: Particle analysis is used to calculate the number of detected objects

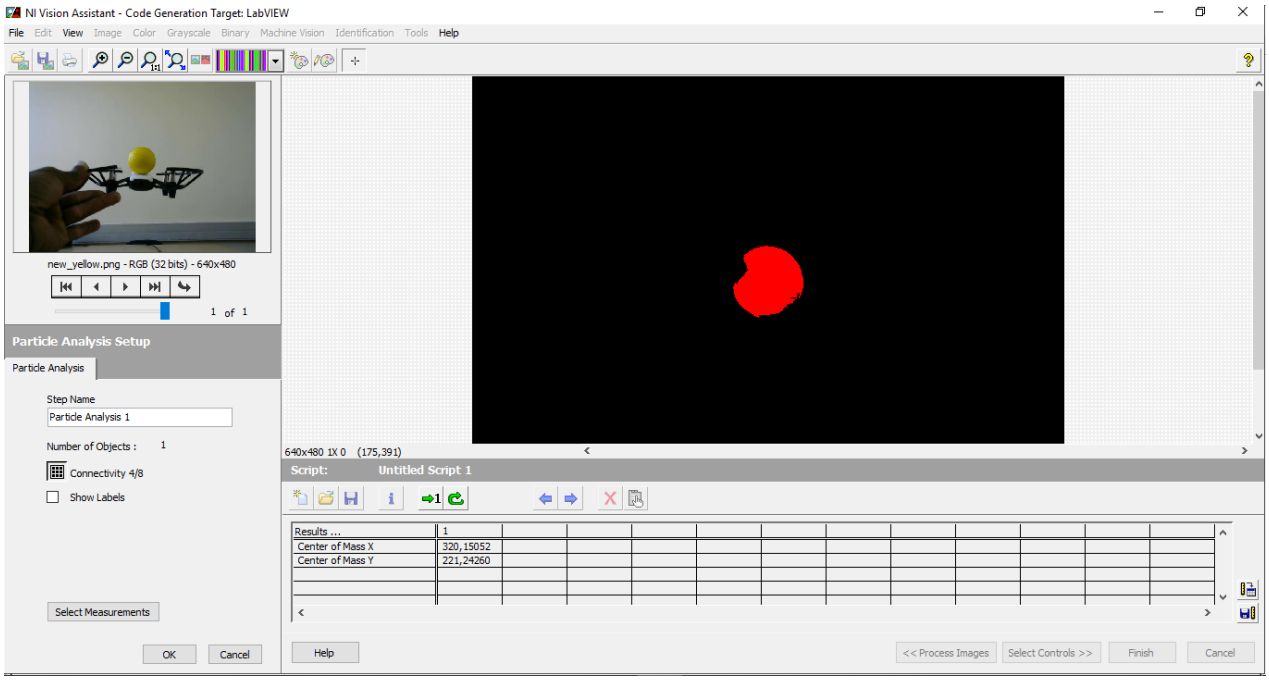


Figure 3.21: particle analysis

Step 6: A coordinate system is assigned to the detected particles

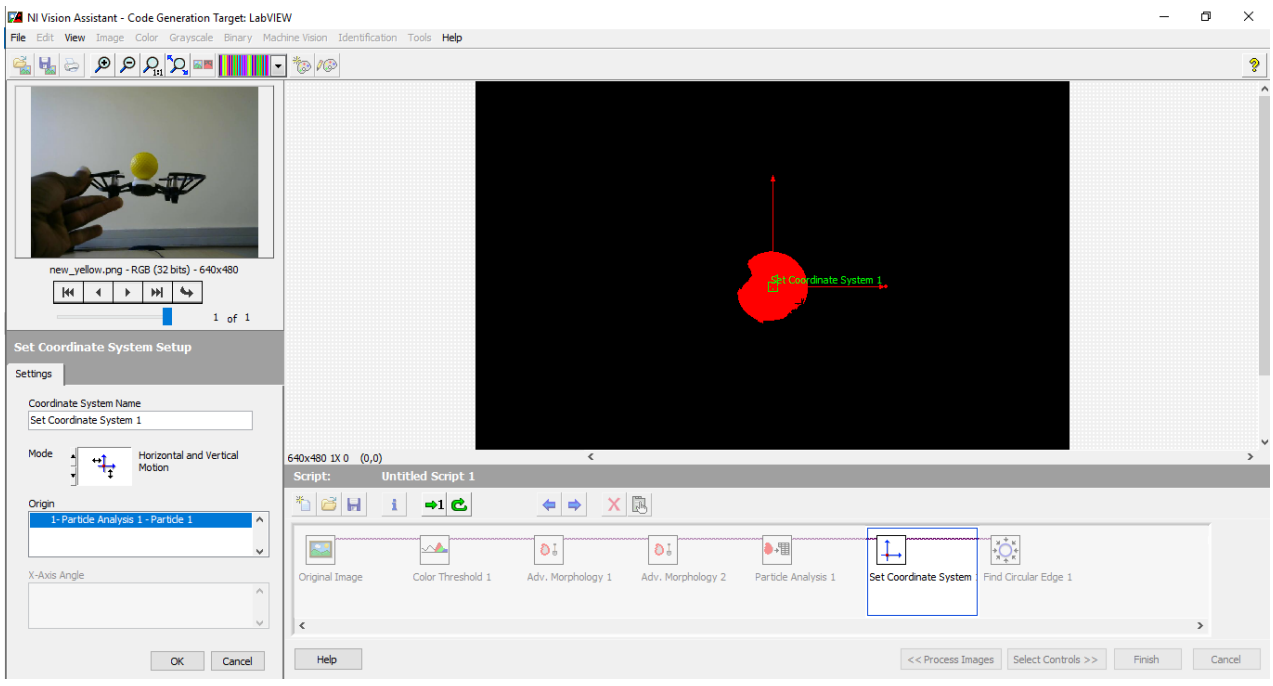


Figure 3.22: Set coordinate system

Step 7: An edge detection setup is used to find circular edges around the detected particle. A circle is drawn around the detected particle and the approximate radius is calculated.

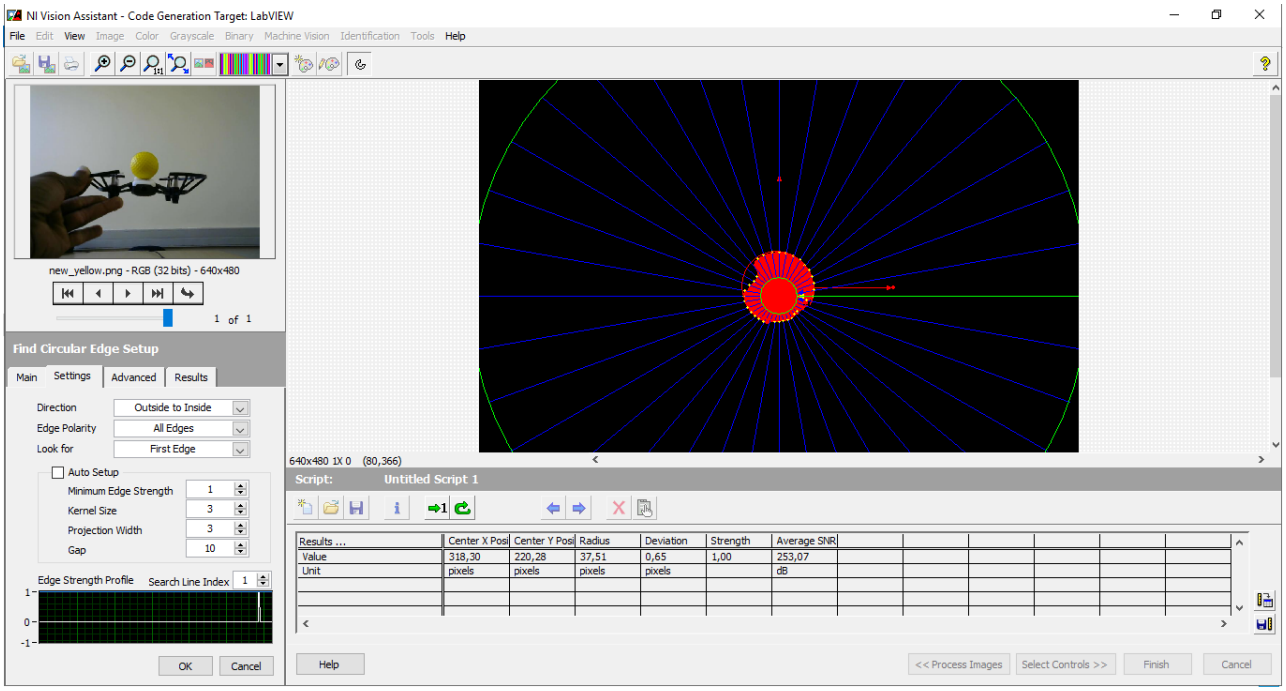


Figure 3.23: Circular edge detection

3.4.2 Ground Station Tracking Algorithm

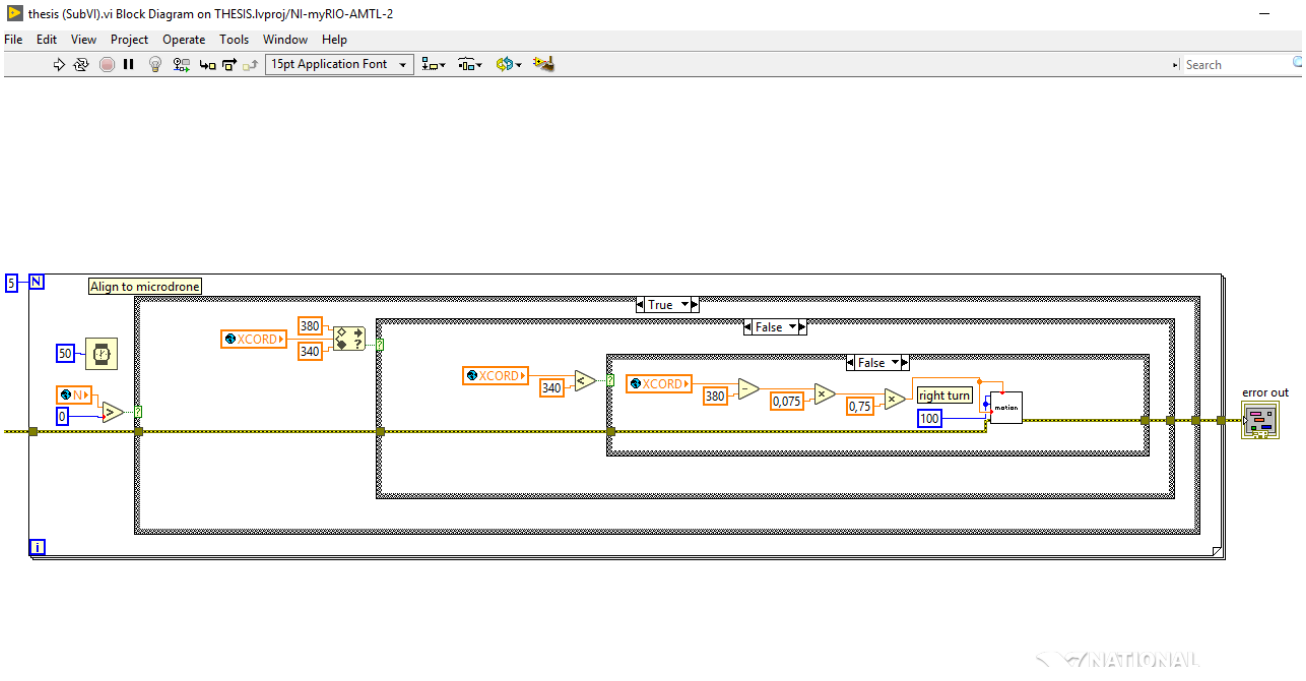


Figure 3.24: Ground station tracking algorithm a

N represents the number of particles found. Where the particles refer to the yellow tracking ball on the drone. XCORD is the x-coordinate of the detected particle. The motion subVI controls the DC motors. The ground station turns left or right to maintain the x-coordinate between 340-380 units.

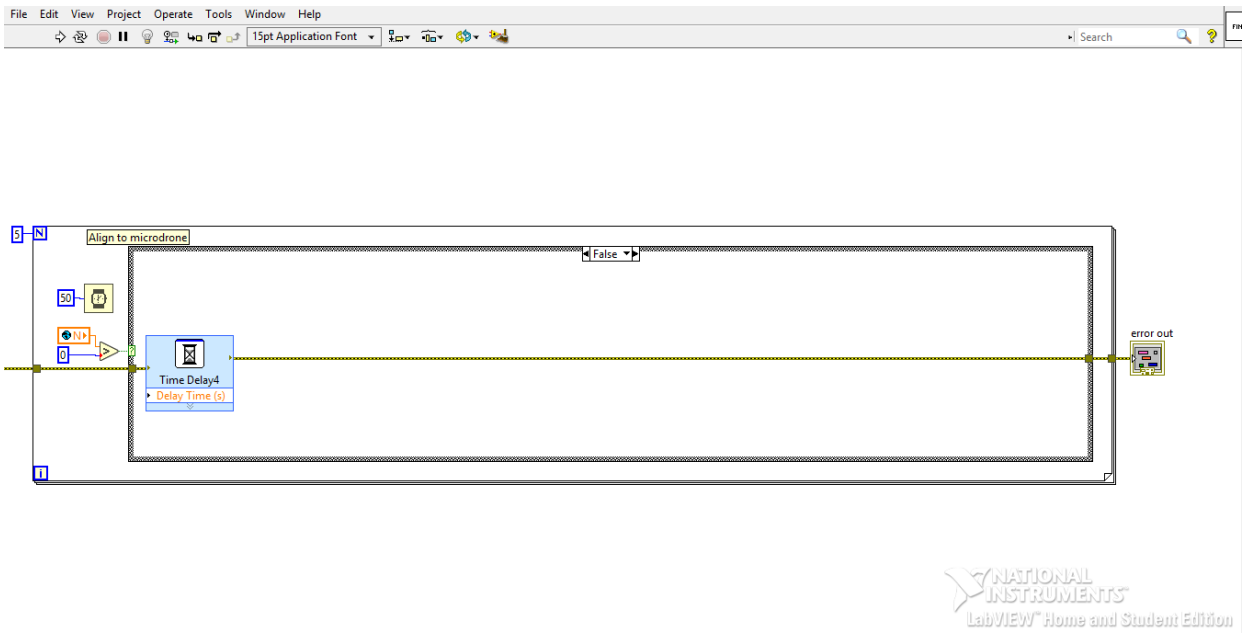


Figure 3.25: Ground station tracking algorithm b

When no particle is found the ground station does not move

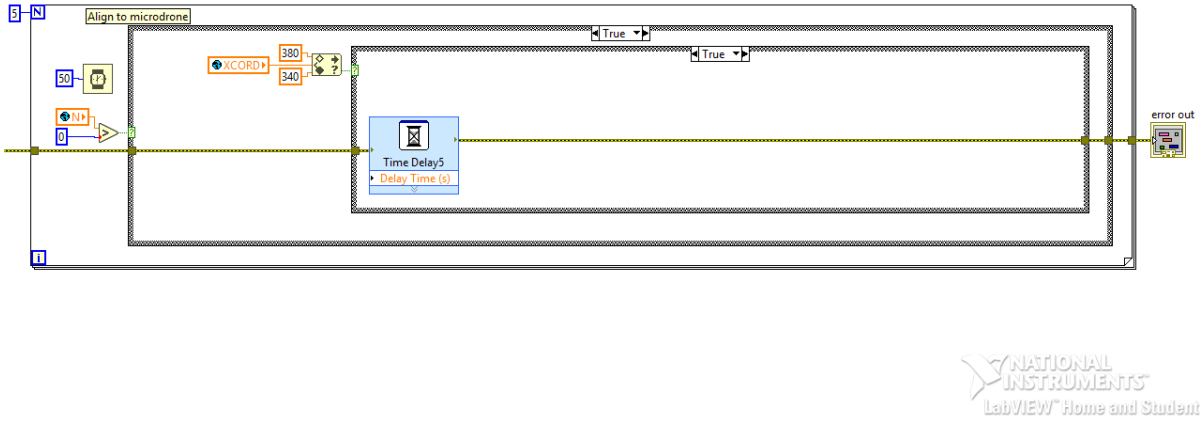


Figure 3.26: Ground station tracking algorithm c

Figure 3.25When particles are found to be within the defined limits of the x-coordinate tracker (340-380), the ground station remains stationary

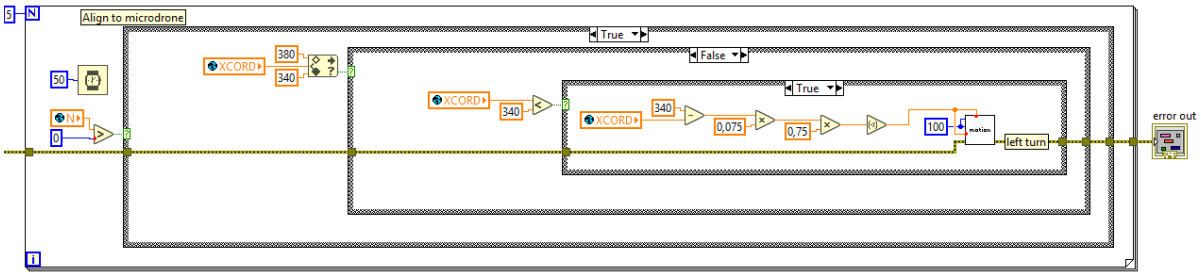


Figure 3.27: Ground station tracking algorithm d

When x-cordinate is less than the lower margin (340), the ground station turns to the left for calculated angle to bring the x-coordinate back to the limits (340-380)

3.4.3 Ground station Movement SubVi's

The ground station utilises DC motors and Servo motors for movements. Two DC motors are used for the steering wheels and 4 servos are used for other functions.

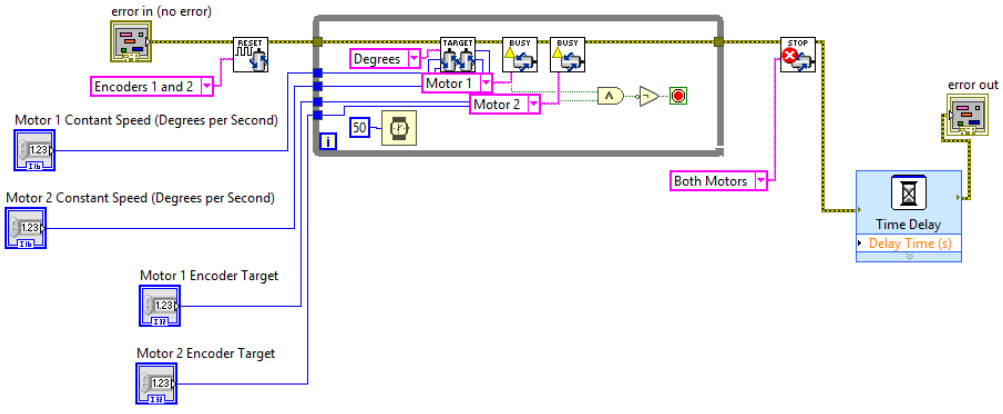


Figure 3.28: Ground station subVI for DC motor control

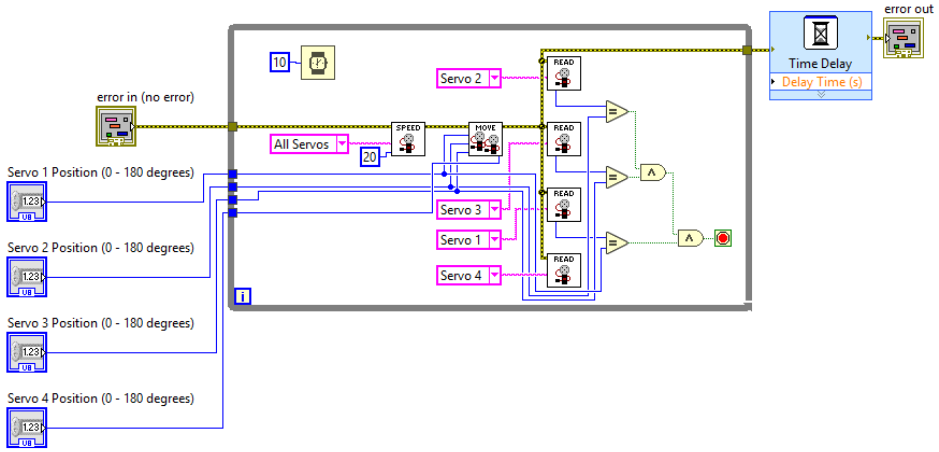


Figure 3.29: Ground station SubVI for servo motor control

The full LabVIEW program for the micro drone is presented in appendix C

3.4.4 Micro drone Tracking Algorithm

```
midx = int(width / 2)
midy = int(height / 2)
xoffset = 0
yoffset = 0
frame_read = tello.get_frame_read()

turn = 0
radius = 1
decide = 0



#video feed

while True:

    frame = frame_read.frame
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    l_b = np.array([15, 146, 150])
    u_b = np.array([68, 255, 255])
```

```
cv2.arrowedLine(frame, (midx, midy),
                (midx + xoffset, midy - yoffset),
                (0, 0, 255), 5)

"""Simple HSV color space tracking"""
   # resize the frame, blur it, and convert it to the HSV
   # color space
blurred = cv2.GaussianBlur(frame, (11, 11), 0)
hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

# construct a mask for the color then perform
# a series of dilations and erosions to remove any small
# blobs left in the mask
mask = cv2.inRange(hsv, l_b, u_b)
mask = cv2.erode(mask, None, iterations=2)
mask = cv2.dilate(mask, None, iterations=2)

   # find contours in the mask and initialize the current
   # (x, y) center of the ball
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
                        cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0]
center = None

# only proceed if at least one contour was found
if len(cnts) > 0:
   # find the largest contour in the mask, then use
   # it to compute the minimum enclosing circle and
   # centroid
   c = max(cnts, key=cv2.contourArea)
   ((x, y), radius) = cv2.minEnclosingCircle(c)
   M = cv2.moments(c)
   center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

   # only proceed if the radius meets a minimum size
   if radius > 10:
      # draw the circle and centroid on the frame,
      # then update the list of tracked points
```

```
            cv2.circle(frame, (int(x), int(y)), int(radius),
                       (0, 255, 255), 2)
            cv2.circle(frame, center, 5, (0, 0, 255), -1)

            xoffset = int(center[0] - midx)
            yoffset = int( midy - center[1])
            decide = 1
            #if microdrone is skewed to the right rotate left
            if xoffset < -distance:
                tello.rotate_counter_clockwise(8)
            #if microdrone is skewed to the left rotate right
            elif xoffset > distance:
                tello.rotate_clockwise(8)
        #if microdrone is within range, do nothing
        else:
            xoffset = 0
            yoffset = 0
            tello.get_battery()

    #if microdrone has not been found keep searching to the left and right
    else:
        xoffset = 0
        yoffset = 0
        tello.get_height()
        if turn == 0 and decide == 0:
            tello.rotate_clockwise(40)
            turn = 1
            time.sleep(1)
        elif turn == 1 and decide == 0:
            tello.rotate_counter_clockwise(40)
            turn = 2
            time.sleep(1)
        elif turn == 2 and decide == 0:
            tello.rotate_counter_clockwise(40)
            turn = 3
            time.sleep(1)
        elif turn == 3 and decide == 0:
            tello.rotate_clockwise(40)
```

```
            turn = 0
            time.sleep(1)
        elif decide == 1:
            time.sleep(0.5)

    #show processed image
    mask = cv2.inRange(hsv, l_b, u_b)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow("mask", mask)

    cv2.imshow("Video", frame)
    key = cv2.waitKey(1)
    #stop tracking if the ground station is close enough
    if radius > 37 and xoffset > -distance and xoffset < distance :
        print("exit tracking mode 1, end tracking radius is", radius)
        break
```

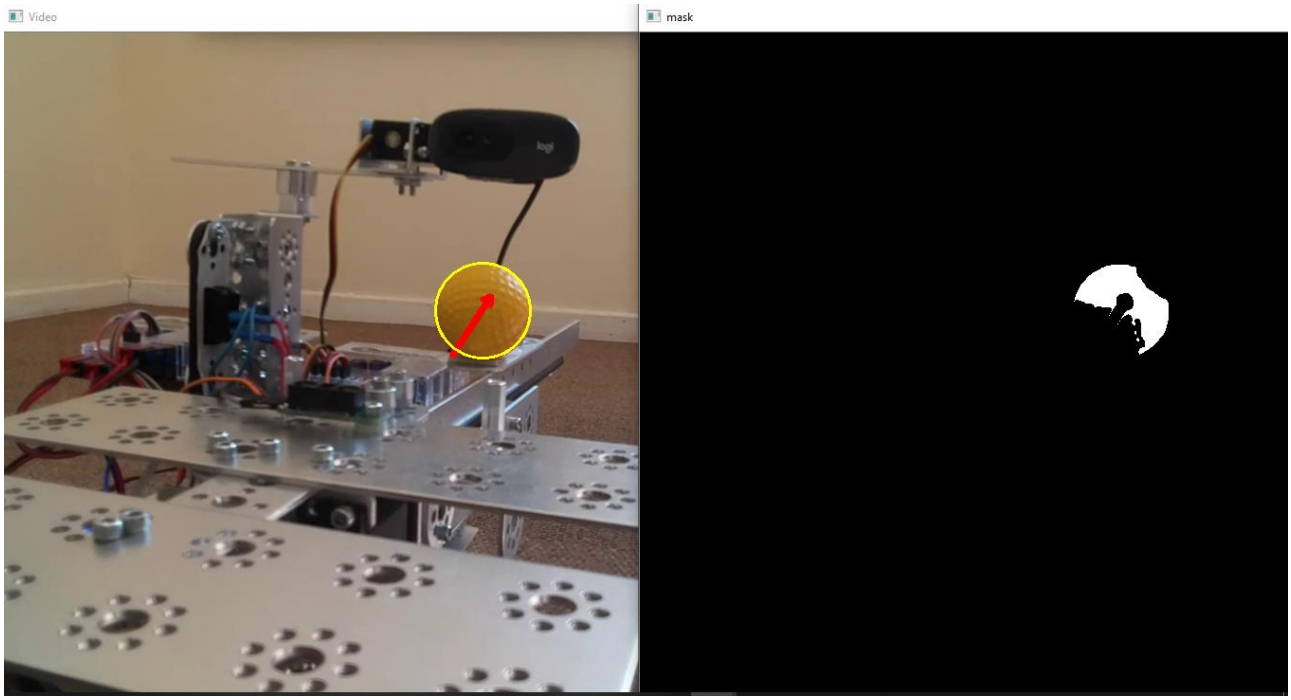


Figure 3.30: Micro drone video feed when running tracking algorithm

The full program for the micro drone is presented in appendix A

CHAPTER FOUR: ANALYSIS AND RESULTS

This chapter presents the narrative of the autonomous docking system demonstration. It shows the mother drone and docked micro drone taking off as a single unit from an initial position. Afterwards, the micro drone is deployed (undocking). The mother drone moves along its mission path while the micro drone flies along its mission path. Then both units arrive at the docking area and align to each other. The micro drone then gives a signal by flying upwards and the mother drone extends its arm for docking to take place. Docking successfully takes place and the mother drone moves with the docked micro drone back to its initial take-off point.

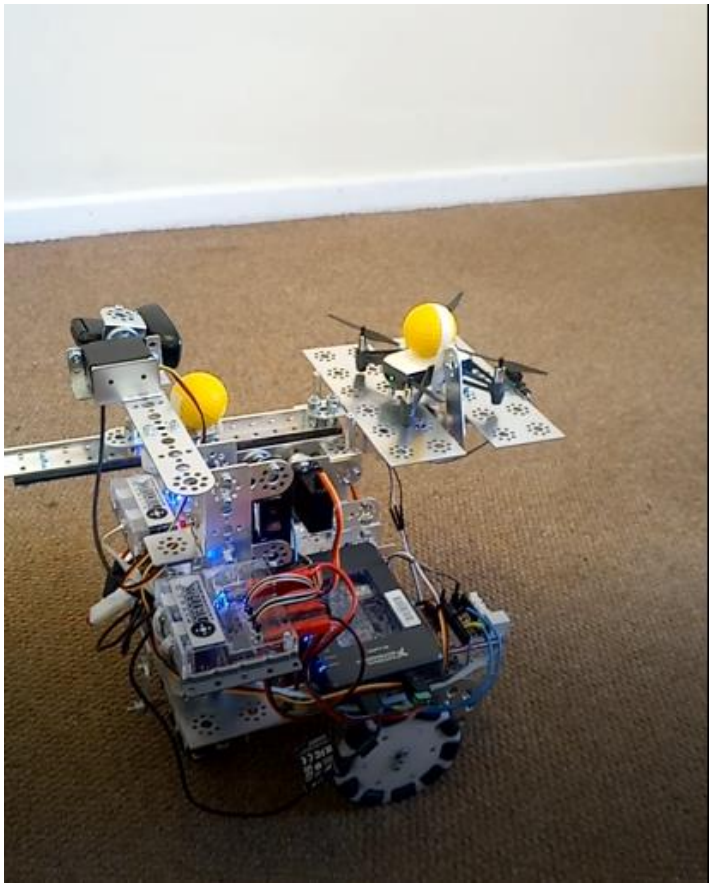


Figure 4.1: Docking system just before take-off
Note that the docking arm is at its initial position which is not extended. The gripper is also closed.

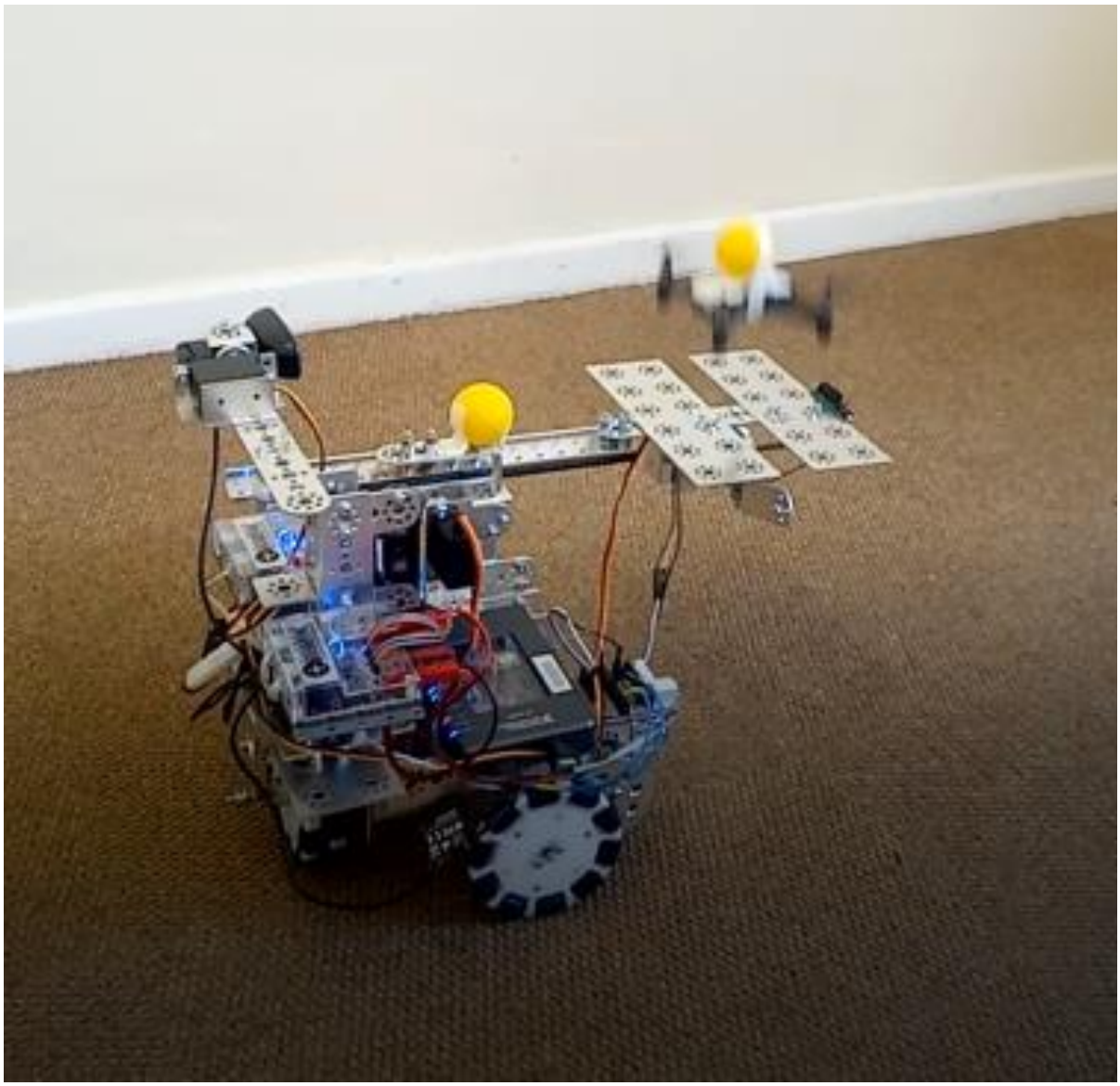


Figure 4.2: docking system during take off
Note that the docking arm is extended

Figure 4.3: Docking system just after take-off
Note that the docking arm retracts back to its initial position

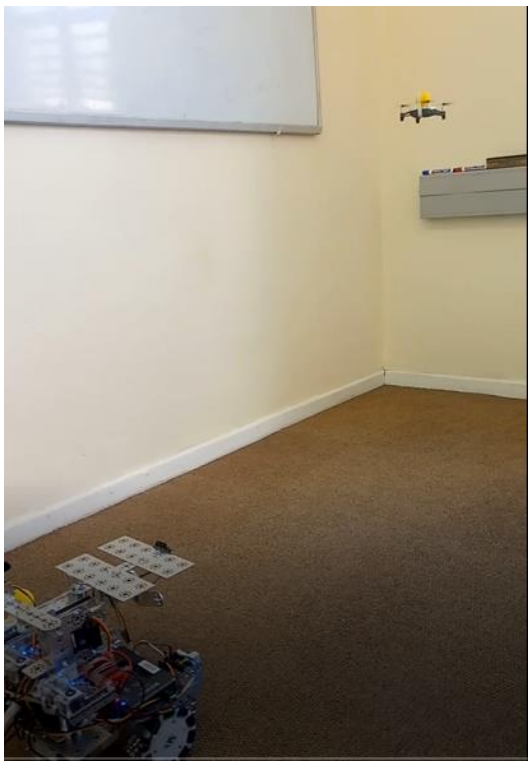


Figure 4.4: Docking system during the flight mission of the micro drone

Figure 4.5: docking system just before landing

Note that the drone is aligned to the yellow ball on the ground station and vice versa

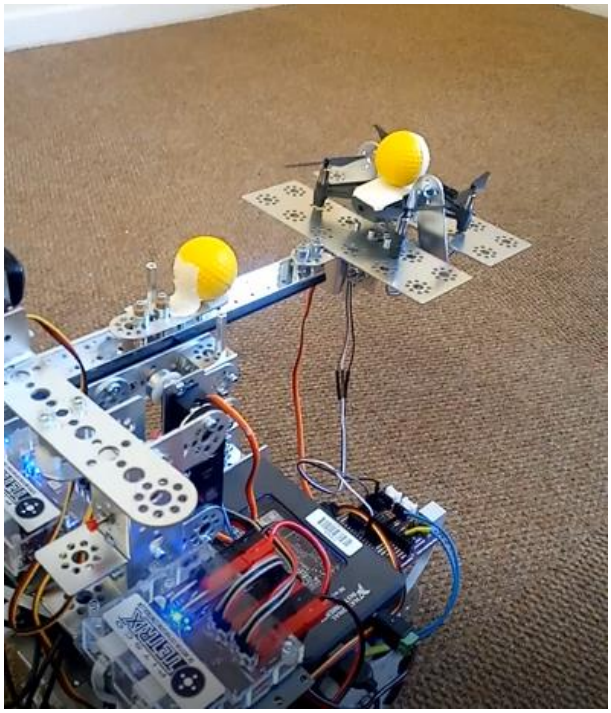


Figure 4.6: Docking system just after landing

Note that the arm is still extended and the grippers are closed

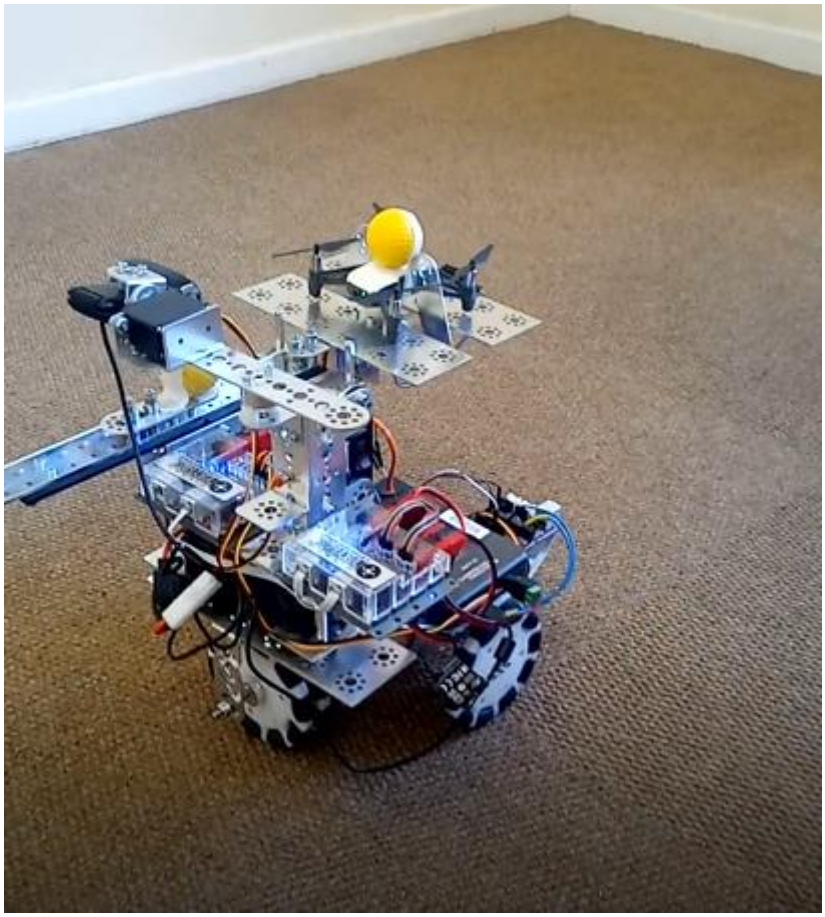


Figure 4.7: Docking system after landing
Note that the arm is back to its initial position and the grippers are closed

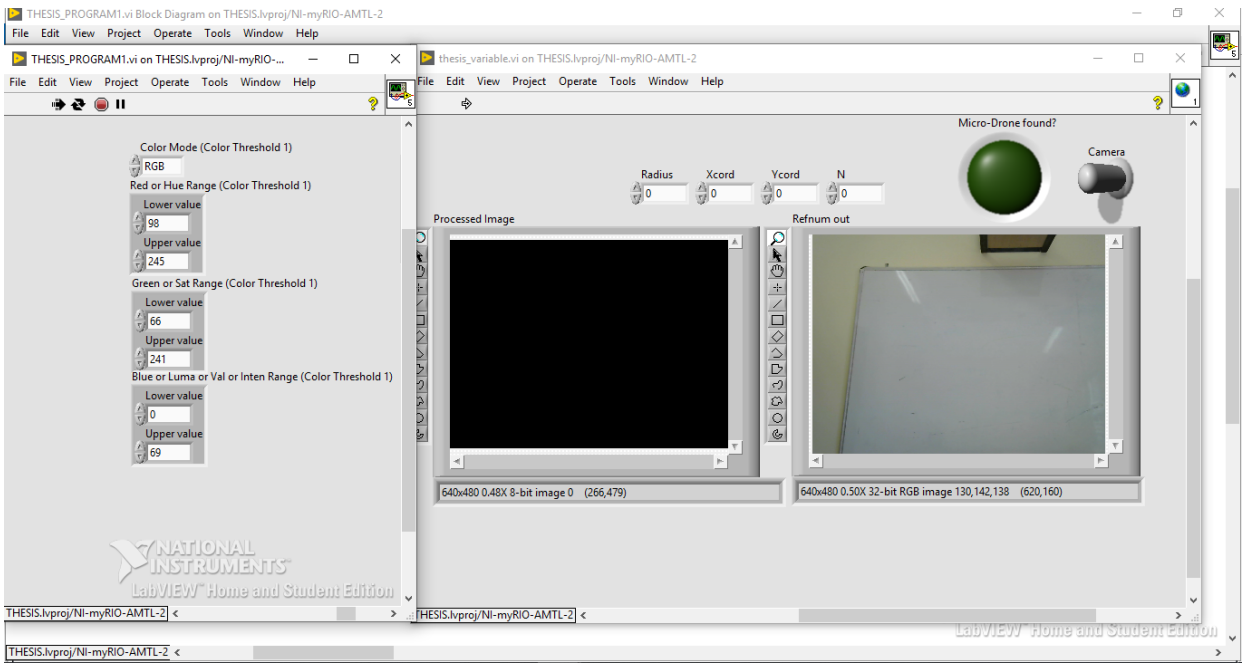


Figure 4.8: Camera peripheral view when no object is detected

A global variable is used to display the output on the two video displays. Where the left display is the processed image and the right display is the original image display.

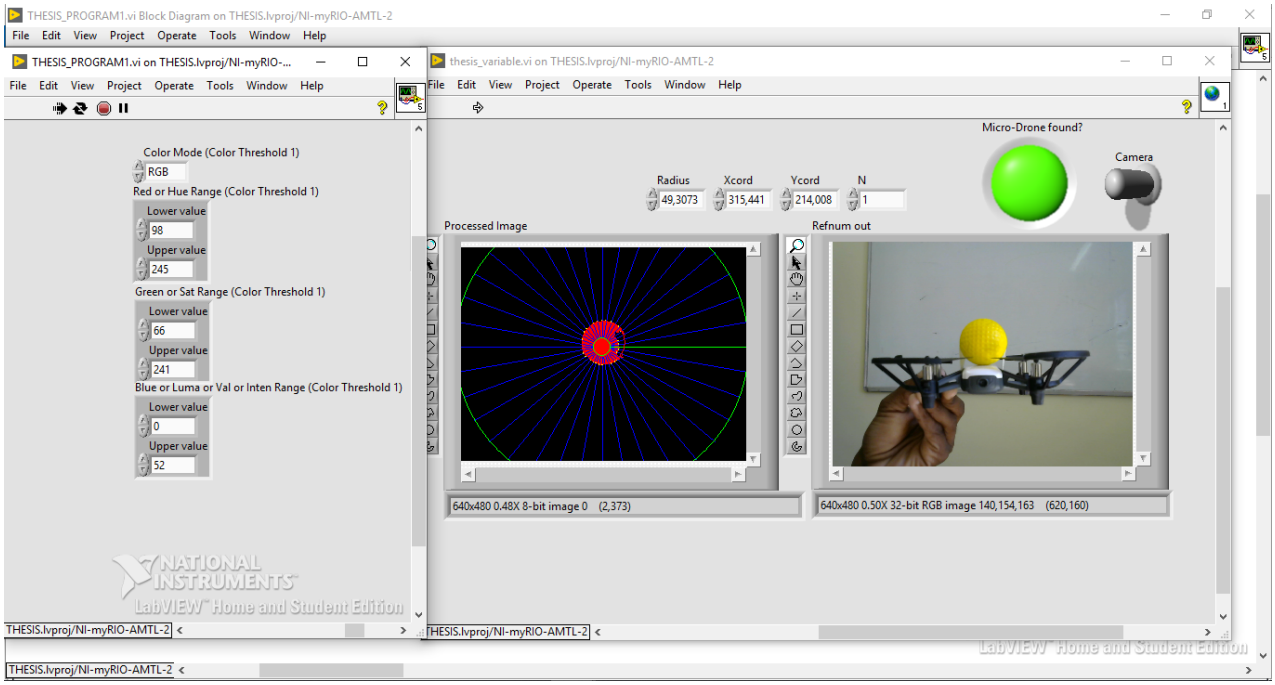


Figure 4.9: Camera peripheral view when Micro-drone is found

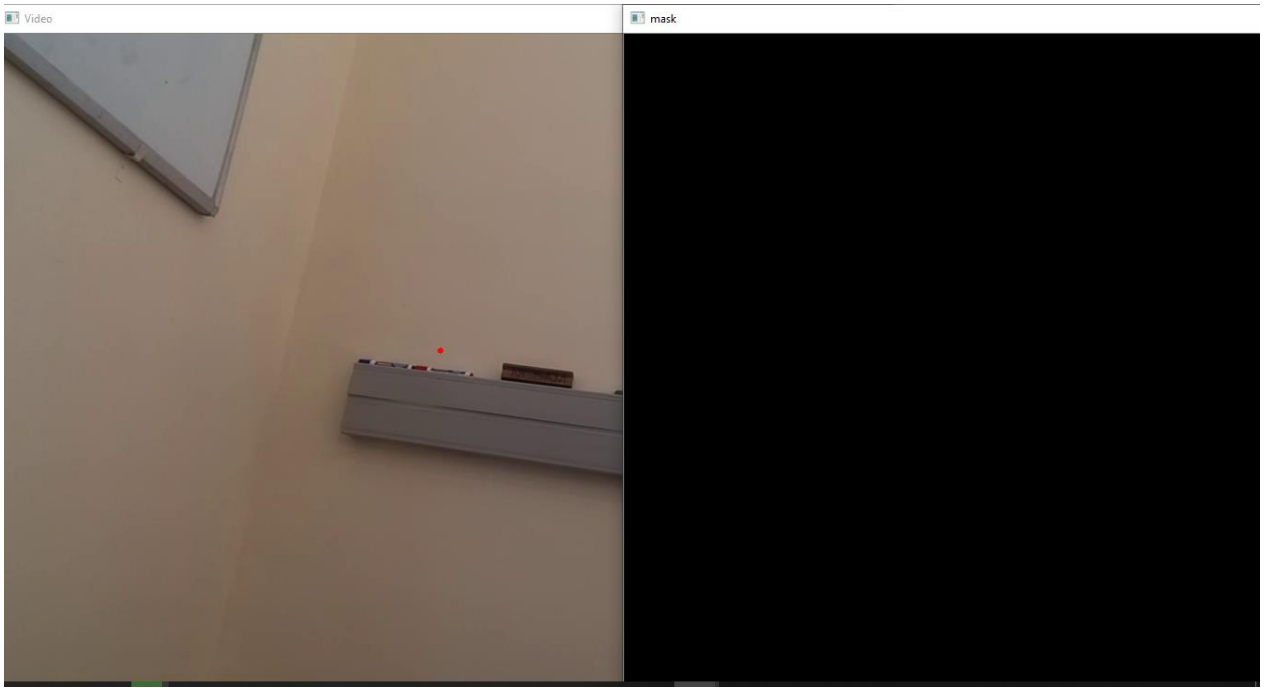


Figure 4.10: Drone Peripheral view when ground station is not found

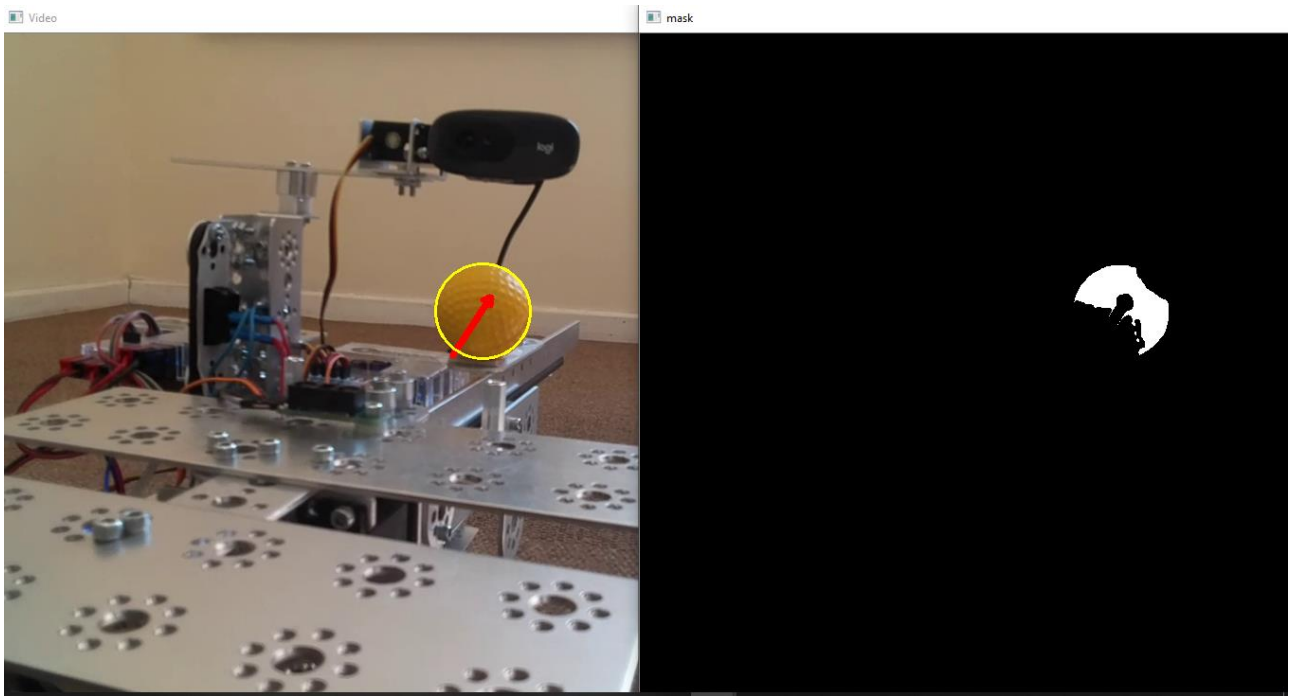


Figure 4.11: Drone Peripheral view when ground station is found.

Note: real display by the left and processed image by the right

CHAPTER FIVE: DISCUSSION

This chapter briefly discusses the tracking ball object and why it was chosen. Furthermore the ground station (mother drone) and micro drone tracking algorithms are briefly discussed.

5.1 Tracking ball

Figure 5.1: Tracking ball

The tracking ball was gotten from the Pitsco tetrix robotics expansion kit. It is a small and light yellow golf ball of 20mm radius. The tracking ball was attached by means of a masking tape. Many tracking objects were tested during the course of developing a reliable object tracking algorithm using colour and shape detection. However, the accuracy of the algorithms were not optimal due to the changing orientation of the tracking objects during flight. The solution to this problem was to use a spherical shape which remains largely consistent irrespective of the orientation of the ball during flight. This improved the ease of identifying the tracking object as well as a centre coordinate position for the tracking algorithms. The LabVIEW program made use of a RGB colour spectrum to identify the yellow colour. Furthermore, a shape detection algorithm was used to estimate the radius of the ball and indirectly infer the distance between the ball and the camera. On the other hand, the

python program used a HSV colour spectrum to identify the yellow ball. When lighting conditions, it was noticed that slight changes needed to be made on the colour spectrum variables to keep the docking system working correctly. It was also necessary to attach the ball in such a way that the propellers can still rotate freely. If the ball was not attached firmly on the micro drone it resulted in a distortion of the micro drone flight path or a crash. The system would fail if there were other yellow or “near yellow” objects in the environment where the docking takes places. This is because the tracking algorithm may not be able to differentiate other yellow objects from its designed target.

5.2 Ground station tracking algorithm

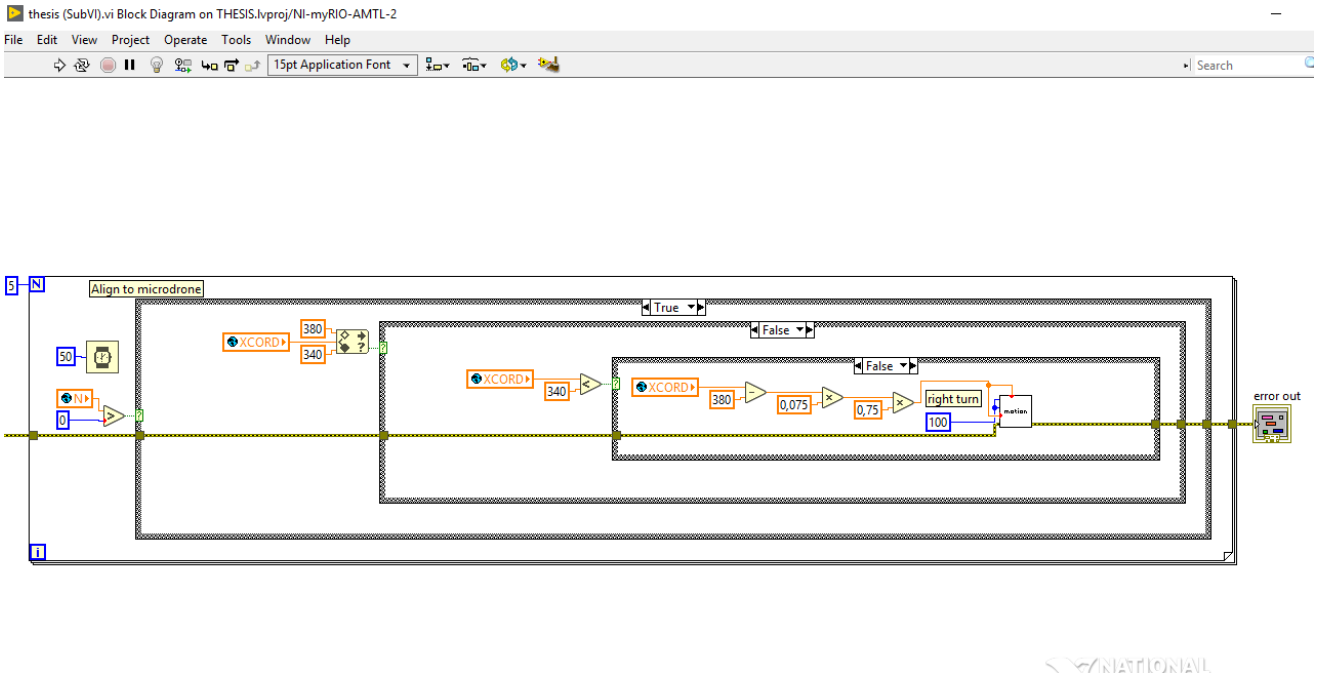


Figure 5.2: Ground station tracking algorithm

The ground station tracking algorithm controls the tracking movements of the ground station after the micro drone has been found. Before the micro drone is found, the ground station carries out a “search mode” in which it basically turns slightly left and right in search of the micro drone. When the micro drone is found, an X-cordinate is assigned to the tracking object based on the position of the object in the peripheral view of the ground station. The ground station then makes a series of turns to ensure that the X-cordinate is within the alignment limit (340 – 380 units). When the X-cordinate from the position of the detected drone is fed in through the XCORD variable, the difference between the measured coordinate and the desired limit is calculated. There are two constants in the program as seen in Figure 5.2 (0.075 and 0.75). The first constant deals with the difference between the measured X-coordinate and the desired limit as well as the turn angle for the ground station. The second constant deals with the number on degrees the wheels need to turn to turn in

order to achieve the turn angle. The algorithm runs on a for-loop which is initiated 5 times. Initially, the algorithm was designed with a while-loop but it proved more prone to duration miscalculation compared to the for-loop. Using a for-loop with a 50milisecond iteration interval proved to be more reliable in predicting exactly when the loop will be completed.

5.3 Drone Tracking

The DJI Tello SDK was used extensively to develop the drone tracking program in Python. The python repository found on Github is shown in appendix B. We used the Anaconda Navigator and Spyder IDE to develop the python program. The drone is rated to have a maximum flight time of 13 minutes but in practice we found it to fly for about 10 minutes before the batteries run out. During the tracking process, the processing intensity makes the drone to heat up and require a few minutes to cool off after landing. The drone does not have a GPS sensor so it uses a vision positioning system for position holding. This vision system is very stable under normal conditions but fails when the lighting condition is poor. During take-off, the ground station has to remain stationary until the micro drone is stable enough to fly forward. If the ground station moves while under the micro drone during take-off, the vision positioning system may detect it as a failure in position hold and this might result in unpredictable deviation from the flight path. Since the tello can be programmed to fly at a certain approximate height, the tracking algorithm excludes the Y-cordinate. The drone only rotates along the yaw-axis during tracking. The docking system worked with a 75% success rate during the course of testing. The failures were mostly due to missed timing calculations between the ground station and the micro drone.

CHAPTER SIX: CONCLUSION AND RECOMMENDATIONS

This chapter briefly presents the research conclusions, possible prospects, applications for the technology and further areas of research.

This research successfully demonstrates the possibility and feasibility of realising an autonomous airborne docking and undocking system for micro drones to a mother drone. The system developed is projected to be a key factor in the advancement of autonomous drone technology. Considering the exponential growth of computer vision, machine learning and computing technologies, autonomous airborne docking and undocking can become more simple and easy to implement. These technological developments can also help reduce the probability of airborne docking failures thereby making the process safe and reliable. As a result it will find applications in areas such as;

6.1 Search and Rescue (SAR)

According to (Vergouw, et al., 2016), as highlighted in the literature review section, future insights for UAV/drone applications in SAR include efficient batteries and energy harvesting solutions for UAVs in long distance missions. This research contributes to achieving this by employing the mother drone as a recharging station, hence allowing for a much longer airborne time for the quadcopter micro drone. (Bejiga, et al., 2017) also pointed out future insights such as power-efficient distributed algorithms for the real-time processing of UAV swarm captured videos, images and sensing data. This research contributes in this aspect by extending the endurance in terms of airborne time for the micro drone, hence allowing for more flexibility in image processing functionalities.

6.2 Precision Agriculture

(Primicerio, et al., 2012) highlighted some future insights in UAV applications towards precision agriculture in the literature review section of this research. This included designing and implementing special types of cameras and sensors on- board UAVs, which have the ability of remote crop monitoring and detection of soil and other agricultural characteristics in

real time scenarios. This research is relevant to the mentioned future insight because special types of cameras and sensors require more power to function. Utilising the mother drone as a recharging station solves this problem.

6.3 Remote sensing

(Yang, et al., 2017) predicts future insights for UAV applications in remote sensing as seen in the literature review section of this research. This included the advancement of UAVs with larger payload, longer flight time, low-cost sensors, improved image processing algorithms for Big data, and effective UAV regulations. This research contributes to realising this by creating a platform for extending the airborne time for drones especially quadcopter.

Other areas of application could include surveillance, fire fighting, military use, videography and much more. Furthermore, this research can be a systematic solution to the energy limitations facing drone technology.

However, under harsh weather conditions or poor lighting conditions the system may not perform optimally. More research can be done to implement deep learning, neural network algorithms and more advanced sensors to make the vision tracking and positioning system more accurate in a variety of environments. In this research, the yellow ball was chosen as the tracking object because of its ease of tracking through a programming algorithm and its shape consistency irrespective of the orientation of the airborne micro drone. More research can be done to explore the possibility of new tracking objects or algorithms. Using an upward facing camera on the micro drone and a downward facing camera on the mother drone can also allow for a more efficient docking and undocking system. Further research can also be done to design a suitable robot arm for such purpose. It is necessary to note that the system presented in this work is presented at a conceptual level and was tested under controlled indoor environments. However, it can be further developed physically on the guardian 4 VTOL drone. If that is to be done, further work will be required on material selection, aerodynamics of the frame and structural analysis.

In conclusion, this research establishes the possibility of autonomous airborne docking in a controlled environment using the handshake docking algorithm as described in Chapter 3 of this research. This algorithm uses computer vision and image processing techniques to allow both mother drone and micro drone track themselves and implement airborne docking as well as undocking. The mother drone utilises a robot docking arm in creating a platform for the micro drone to dock and undock. This robot arm requires a distance sensor to complement the tracking camera on the mother drone. The camera and distance sensor work together to make the robot arm move accurately and precisely. We project that this research will find use in the further development of autonomous drone technology.

REFERENCES

1. AIRBORNEDRONES, 2018. *Sentinel+ drone.* [Online]
   Available at: http://www.airbornedrones.co/
   [Accessed 6 December 2019].
2. Akesson, N. B. & Yates, W. E., 1974. *The use of aircraft in agriculture.* California, Food & Agriculture Org.
3. Alcedo, T., 2018. *Alcedo.* [Online]
   Available at: http://www.alcedo.ethz.ch/
   [Accessed 6 December 2019].
4. Alexopoulos, A., Kandil, A., Orzechowski, P. & Badreddin, E., 2013. *A comparative study of collision avoidance techniques for unmanned aerial vehicles.* Manchester, International Conference on Systems, Man, and Cybernetics (SMC), IEEE, p. 1969–1974.
5. Al-Hourani, A., Kandeepan & Jamalipour, A., 2014. *Modeling air-to- ground path loss for low altitude platforms in urban environments.* Texas, Global Communications Conference (GLOBECOM), IEEE, p. 2898–2904.
6. Alisa, K., 2018. *BUSINESS POTENTIAL ANALYSIS OF UAV APPLICATIONS,* Lappeenranta: Global Management of Innovation and Technology.
7. Anderson, K. & Gaston, K. J., 2013. Lightweight unmanned aerial vehicles will revolutionize spatial ecology. *Frontiers in Ecology and the Environment,* 11(3), p. 138–146.
8. Andreas, V. H., 2015. *Model, Design and Control of a Quadcopter,* Trondheim: Norwegian University of Science and Technology.
9. Anudeep, M., Diwakar, G. & Ravi, K., 2014. Design of A Quad Copter and Fabrication. *International Journal of Innovations in Engineering and Technology (IJIET),* 4(1), pp. 59-65.
10. Austin, R., 2011. Unmanned aircraft systems: UAVS design, development and deployment. *John Wiley & Sons,* 54(1), pp. 1-372.
11. Baluja, J. et al., 2012. Assessment of vineyard water status variability by thermal and multispectral imagery using an unmanned aerial vehicle (UAV). *Irrigation Science,* 30(6), p. 511–522.
12. Bannari, A., Morin, D., Bonn, F. & Huete, A., 1995. A review of vegetation indices. *Remote sensing reviews,* 13(1-2), p. 95–120.
13. Bas, V., Huub, N., Geert, B. & Bart, C., 2016. Drone Technology: Types, Payloads, Applications, Frequency Spectrum Issues and Future Developments. In: B. Custers, ed. *The Future of Drone Use, Opportunities and Threats from Ethical and Legal Perspectives.* The Hague: T.M.C Asser Press, pp. 21-42.
14. Bejiga, M. B., Zeggada, A., Nouffidj, A. & Melgani, F., 2017. A convolutional neural network approach for assisting avalanche search and rescue operations with UAV imagery. *Remote Sensing,* 9(2), p. 100.
15. Bhardwaj, A., Sam, L., Mart´ın-Torres, F. J. & Kumar, R., 2016. UAVs as remote sensing platform in glaciology: Present applications and future prospects. *Remote Sensing of Environment,* 175(1), p. 196–204.
16. Burwood-Taylor, L., 2018. *The next generation of drone technologies for agriculture.* [Online]
   Available at: https://agfundernews.com/the-next-generation-of-drone-technologies-for-agriculture.html
   [Accessed 4 December 2019].
17. Calder´on, R., Navas-Cort´es, J. A., Lucena, C. & Zarco-Tejada, P. J., 2013. High-resolution airborne hyperspectral and thermal imagery for early detection of verticillium wilt of olive using fluorescence, temperatureand narrow-band spectral indices. *Remote Sensing of Environment,* 139(1), p. 231–245.

18. Candiago, S. et al., 2015. Evaluating multispectral images and vegetation indices for precision farming applications from UAV images. *Remote Sensing,* 7(4), p. 4026–4047.
19. Carrio, A., Sampedro, C., Rodriguez-Ramos, A. & Campoy, P., 2017. A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors,* 2017(1), pp. 1-13.
20. Chandra, P. S. et al., 2016. *Farmer's Handbook on Basic Agriculture.* 2nd ed. Navsari: Desai Fruits & Vegetables Pvt. Ltd..
21. Chris , H., 2008. *Basic Engineering Design Process,* Virginia: ICE Training.
22. Curry, J., Maslanik, J., Holland, G. & Pinto, J., 2004. Applications of aerosondes in the arctic. *Bulletin of the American Meteorological Society,* 85(12), p. 1855–1861.
23. Dai, L. S. et al., 2016. *3D Printed Quadcopters,* New Jersey: New Jersey Governor's School of Engineering and Technology.
24. De-Cai, W. et al., 2012. Mapping soil texture of a plain area using fuzzyc-means clustering method based on land surface diurnal temperature difference. *Pedosphere,* 22(3), p. 394–403.
25. Definiens, A., 2007. Definiens developer 7 user guide. *Document version,* 7(5), p. 968.
26. Digital Transformation monitor, 2018. *Drones in agriculture ,* New York: Digital Transformation Monitor.
27. Dirman, H. et al., 2013. *Simple GUI Wireless Controller of Quadcopter,* Malaysia: Department of Mechatronic and Robotic Engineering, University Tun Hussein Onn.
28. Doherty, P. & Rudol, P., 2007. *A UAV search and rescue scenario with human body detection and geolocalization.* Australia, Springer, pp. 1-13.
29. Endrowednes, K., Dan, C. & Radu, T., May, 2016. *QUADCOPTER BODY FRAME MODEL,* Romania: ANNALS OF THE UNIVERSITY OF ORADEA, FASCICLE OF MANAGEMENT AND TECHNOLOGICAL ENGINEERING.
30. Erginer, B. & Altu, E., 2007. *Modeling and PD Control of a Quadrotor VTOL Vehicle.* Itanbul, IEEE Intelligent Vehicles Symposium, pp. 894-899.
31. Erico, P. F., 2017. Seed Plant Drone for Reforestation. *The Graduate Review,* 2(7), pp. 13-26.
32. Garcia-Ruiz, F. et al., 2013. Comparison of two aerial imaging platforms for identification of huanglongbing-infected citrus trees. *Computers and Electronics in Agriculture,* 91(1), p. 106–115.
33. Geipel, J., Link, J. & Claupein, W., 2014. Combined spectral and spatial modeling of corn yield based on aerial images and crop surface models acquired with an unmanned aircraft system. *Remote Sensing,* 6(11), pp. 10 335–10 355,.
34. Gibiansky, A., 2012. *Quadcopter Dynamics and Simulation.* [Online] Available at: http://andrew.gibiansky.com/downloads/pdf/Quadcopter%20Dynamics,%20Simulation,%20and%20Control.pdf [Accessed 1 June 2018].
35. GisCloud, 2018. *Combining remote sensing and cloud technology is the future of farming.* [Online] Available at: https://www.giscloud.com/blog/agriculture-risk-management-use-case/ [Accessed 6 December 2019].
36. Gitelson, A. A., Kaufman, Y. J. & Merzlyak, M. N., 1996. Use of a green channel in remote sensing of global vegetation from eos-modis. *Remote sensing of Environment,* 58(3), p. 289–298.
37. Giusti, A. et al., 2016. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters,* 1(2), pp. 661-667.
38. Glenn, E. P., Huete, A. R., Nagler, P. L. & Nelson, S. G., 2008. Relationship between remotely-sensed vegetation indices, canopy attributes and plant physiological processes: what vegetation indices can and cannot tell us about the landscape. *Sensors,* 8(4), p. 2136–2160.

39. Gonzalez-Dugo, V. et al., 2013. Using high resolution UAV thermal imagery to assess the variability in the water status of five fruit tree species within a commercial orchard. *Precision Agriculture,* 14(6), p. 660–678.
40. Gordana, O., Branislav, T., Stevan, S. & Nikola, Đ., 2015. Design, control and application of quadcopter. *International Journal of Industrial Engineering and Management,* 6(1), p. 46.
41. Gupta, L., Jain, R. & Vaszkun, G., 2016. Survey of important issues in UAV communication networks. *IEEE Communications Surveys & Tutorials,* 18(2), p. 1123–1152.
42. Haomiao, H., Hoffmann, G. M., Waslander, S. L. & Tomlin, C. J., 2009. *Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering,* Kobe: 2009 IEEE International Conference on Robotics and Automation.
43. Hassan-Esfahani, L., Torres-Rua, A., Jensen, A. & McKee, M., 2015. Assessment of surface soil moisture using high-resolution multi-spectral imagery and artificial neural networks. *Remote Sensing,* 7(3), p. 2627–2646.
44. Hayat, S., Yanmaz, E. & Muzaffar, R., 2016. Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint. *IEEE Communications Surveys & Tutorials,* 18(4), p. 2624–2661.
45. Hazim, S. et al., 2019. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Acess,* 7(1), pp. 48572-48634.
46. Henry, B., 2018. *Business Insider.* [Online]
Available at: http://www.businessinsider.com/drones-report-market-forecast-2015-3?IR=T
[Accessed 30 May 2018].
47. Heong Ang, K., Chong, G. & Yun, L., 2005. PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology,* 13(4), pp. 559-576.
48. Hernandez-Lopez, J.-J.et al., 2012. Detecting objects using color and depth segmentation with kinect sensor. *Procedia Technology,* 3(1), p. 196–204.
49. Hofstrand, D., 2015. Economics of tile drainage. *Ag Decision Maker Newsletter,* 14(9), p. 3.
50. Huang, Y. et al., 2013. Development and prospect of unmanned aerial vehicle technologies for agricultural production management. *International Journal of Agricultural and Biological Engineering,* 6(3), p. 1–10.
51. Huete, A. R., 1988. A soil-adjusted vegetation index (savi). *Remote sensing of environment,* 25(3), p. 295–309.
52. Hugenholtz, C. H. et al., 2013. Geomorphological mapping with a small unmanned aircraft system (sUAS): Feature detection and accuracy assessment of a photogrammetrically-derived digital terrain model. *Geomorphology,* 194(1), p. 16–24.
53. hummingbirdtech, 2018. *Advanced crop analytics and artificial intelligence for farmers.* [Online]
Available at: https://hummingbirdtech.com/
[Accessed 2 December 2019].
54. Hunt, E. R. et al., 2010. Acquisition of nir-green-blue digital photographs from unmanned aircraft for crop monitoring. *Remote Sensing,* 2(1), p. 290–305.
55. Immerzeel, W. et al., 2014. High-resolution monitoring of himalayan glacier dynamics using unmanned aerial vehicles. *Remote Sensing of Environment,* 150(1), p. 93–103.
56. James, D. & Ryan, B., 2014. *Quadcopter Design Project,* Pennsylvania: Pennsylvania State University.
57. Jensen, T. et al., 2003. *Assessing grain crop attributes using digital imagery acquired from a low-altitude remote controlled aircraft.* Denmark, Proceedings of the 2003 Spatial Sciences Institute Conference: Spatial Knowledge Without Boundaries (SSC2003).
58. Jensen, T., Apan, A. & Zeller, L., 2009. *Crop maturity mapping using a low-cost low-altitude remote sensing system.* Korea, Proceedings of the 2009 Surveying and Spatial Sciences Institute Biennial International Conference (SSC 2009), p. 1231–1243.

59. Jo, D. & Kwon, Y., 2017. Development of rescue material transport UAV (unmanned aerial vehicle). *World Journal of Engineering and Technology,* 5(4), p. 720.
60. Joern, J., 2015. *Examining the use of unmanned aerial systems and thermal infrared imaging for search and rescue efforts beneath snowpack* [Interview] (04 04 2015).
61. John, C., Valery, A., Thomas, H. & Wes, B., 2018. *ISS Interface Mechanisms and their Heritage,* Houston: The Boeing Company, 13100 Space Center Boulevard.
62. Johnson, W., November, 2000. *Calculation of Tilt Rotor Aeroacoustic Model (TRAM DNW) Performance, Airloads, and Structural Loads," American Helicopter SocietyLoads," American Helicopter Society Loads,,* Atlanta, Georgia: American Helicopter Society Aeromechanics Specialist Meeting.
63. Jordan, B. R., 2015. A birds-eye view of geology: The use of microdrones/UAVs in geologic fieldwork and education. *GSA Today,* 25(7), pp. 50-52.
64. Jun, L. & Yutang, L., 2011. *Dynamic Analysis And Pid Control For A Quadrotor,* Beijing: International Conference On Mechatronics And Automation.
65. Justin, W., Moble, B. & Vikram, H. I. C., 2016. Design, development, and flight testing of a high endurance micro quadrotor helicopter. *International Journal of Micro Air Vehicles,* 8(3), pp. 155-169.
66. Kahn, M., 2014. Quadcopter Flight Dynamics. *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH,* 3(8), pp. 130-135.
67. Karapantazis, S. & Pavlidou, F., 2005. Broadband communications via high-altitude platforms: A survey. *IEEE Communications Surveys & Tutorials,* 7(1), p. 2–31.
68. Kaushal, H. & Kaddoum, G., 2017. Optical communication in space: Challenges and mitigation techniques. *IEEE Communications Surveys & Tutorials,* 19(1), p. 57–96.
69. Kazmi, W. et al., 2011. *Adaptive surveying and early treatment of crops with a team of autonomous vehicles..* Denmark, ECMR.
70. Khanal, S., Fulton, J. & Shearer, S., 2017. An overview of current and potential applications of thermal remote sensing in precision agriculture. *Computers and Electronics in Agriculture,* 139(1), p. 22–32.
71. Khanal, S., Fulton, J. & Shearer, S., 2017. An overview of current and potential applications of thermal remote sensing in precision agriculture. *Computers and Electronics in Agriculture,* 139(1), p. 22–32.
72. Korchenko, A. & Illyash, O., 2013. *The generalized classification of unmanned air vehicles.* Kiev, IEEE 2nd International Conference on Actual Problems of Unmanned Air Vehicles Developments Proceedings (APUAVD), pp. 28-34.
73. Laliberte, A. S., Herrick, J. E., Rango, A. & Winters, C., 2010. Acquisition, orthorectification, and object-based classification of unmanned aerial vehicle (uav) imagery for rangeland monitoring. *Photogrammetric Engineering & Remote Sensing,* 76(6), p. 661–672.
74. Laliberte, A. S., Winters, C. & Rango, A., 2008. *A procedure for orthorectification of sub-decimeter resolution imagery obtained with an unmanned aerial vehicle (UAV).* Florida, Proc. ASPRS Annual Conf, pp. 8-47.
75. Larry, C., Rebecca, A. A., Chris, C. & Frederic, T., 2015. *SMART Fire Fighting: The Use of Unmanned Aircraft Systems in the Fire Service,* Alabama: NFPA Responder Forum,.
76. Lauren, F., 2018. *BioCarbon Engineering.* [Online] Available at: https://www.biocarbonengineering.com/blog/biocarbon-engineering-receives-us-2-5-million-in-investment-to-advance-drone [Accessed 26 June 2018].
77. Lin, P.-H. & Lee, C.-S., 2008. The eyewall-penetration reconnaissance observation of typhoon longwang (2005) with unmanned aerial vehicle, aerosonde. *Journal of Atmospheric and Oceanic Technology,* 25(1), p. 15–25.
78. Macke Jr, D. C., 2013. *Systems and image database resources for UAV search and rescue applications ,* Missouri: Missouri University of Science and Technology.
79. Madden, M. et al., 2015. The future of unmanned aerial systems (UAS) for monitoring natural and cultural resources. *Photogrammetric Week,* 15(1), p. 369–384.

80. Mathur, P., Nielsen, R. H., Prasad, N. R. & Prasad, R., 2016. Data collection using miniature aerial vehicles in wireless sensor networks. *IET Wireless Sensor Systems,* 6(1), p. 17–25.
81. McGonigle, A. et al., 2008. Unmanned aerial vehicle measurements of volcanic carbon dioxide fluxes. *Geophysical research letters,* 35(6).
82. Meng Leong, B., Low, S. & Po-LeenOoi, M., 2012. Low-Cost Microcontroller-based Hover Control Design of a Quadcopter. *ScienceDirect,* 41(1), pp. 458-461.
83. Mikolajczyk, K., Schmid, C. & Zisserman, A., 2004. *Human detection based on a probabilistic assembly of robust part detectors.* Oxford, Computer Vision-ECCV 2004, p. 69–82.
84. Mitra, S., 2013. *Autonomous Quadcopter Docking System,* New York: Cornel University.
85. Mo, M. A. & Saw, A. N. O., 2018. Design and Implementation of Trainable. *International Journal of Science, Engineering and Technology Research (IJSETR),* 7(2), pp. 48-53.
86. Muchiri, N. & Kimathi, S., 2016. *A review of applications and potential applications of UAV.* Kenya, Proceedings of Sustainable Research and Innovation Conference, p. 280–283.
87. NASA, 2017. *Remote sensors.* [Online]
Available at: https://earthdata.nasa.gov/user-resources/remote-sensors
[Accessed 1 December 2019].
88. Nonami, K. et al., 2010. *Autonomous Flying Robots – Unmanned Aerial Vehicles and Micro Aerial Vehicles.* Tokyo, Springer, pp. 48-52.
89. Park, D., Park, M.-S. & Hong, S.-K., 2011. *A Study on the 3-DOF Attitude Control of Free-Flying Vehicle.* Pusan, Proceeding of the IEEE International Symposium on Indusrial Electronics (ISIE).
90. Patil, J. K. & Kumar, R., 2011. Advances in image processing for detection of plant diseases. *Journal of Advanced Bioinformatics Applications and Research,* 2(2), p. 135–141.
91. Pedro, C., Alejandro, D. & Rogelio, L., 2004. Real-time stabilization and tracking of a four-rotor mini rotorcraft. *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY,* 12(4), pp. 510-515.
92. Pimentel, D., Zuniga, R. & Morrison, D., 2005. Update on the environmental and economic costs associated with alien-invasive species in the united states. *Ecological economics,* 52(3), p. 273–288.
93. Prasad, L., Tyagi, B. & Gupta, H., 2011. Optimal control of nonlinear inverted pendulum dynamical system with disturbance input using PID controller & LQR. *International Journal of Automation and Computing,* 11(6), pp. 661-670.
94. Primicerio, J. et al., 2012. A flexible unmanned aerial vehicle for precision agriculture. *Precision Agriculture,* 13(4), p. 517–523.
95. Python_Tips, 2017. *Introduction to machine learning and its usage in remote sensing.* [Online]
Available at: https://pythontips.com/2017/11/11/introduction-to-machine-learning-and-its-usage-in-remote-sensing/
[Accessed 3 December 2019].
96. Ravindra, K. & Divakar Raju, P., 2017. ANALYSIS OF AIRCRAFT WING WITH DIFFERENT MATERIALS USING ANSYS SOFTWARE. *International Research Journal of Engineering and Technology (IRJET),* 4(10), pp. 1280-1285.
97. Reed, B. C. et al., 1994. Measuring phenological variability from satellite imagery. *Journal of vegetation science,* 5(5), p. 703–714.
98. Reynaud, L. & Rasheed, T., 2012. *Deployable aerial communication networks: challenges for futuristic applications.* Cyprus, Proceedings of the 9th ACM symposium on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks.
99. Rhoads, F. M. & Yonts, C. D., 2000. *Irrigation scheduling for corn: why and how.* Florida: National Corn Handbook.
100. Rouse Jr, J., Haas, R., Schell, J. & Deering, D., 1974. *Monitoring vegetation systems in the great plains with erts.* Texas: Texas A&M University College station.

101. Rudol, P. & Doherty, P., 2008. *Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery,* Link¨oping: IEEE Aerospace Conference.
102. Saggiani, G. et al., 2007. *A UAV system for observing volcanoes and natural hazards.* Washington, AGU Fall Meeting Abstracts.
103. Salih, A. L., Moghavvemmil, M. & Gaeid, K. S., 2010. Flight PID Controller Design for a UAV Quad-copter. *Scientific Research and Essays,* 5(23), pp. 3660-3667.
104. Sam, L., Bhardwaj, A., Singh, S. & Kumar, R., 2016. Remote sensing flow velocity of debris-covered glaciers using landsat 8 data. *Progress in Physical Geography,* 40(2), p. 305–321.
105. Sankaran, S. et al., 2015. Low-altitude, high-resolution aerial imaging systems for row and field crop phenotyping: A review. *European Journal of Agronomy,* 70(1), p. 112–123.
106. Scherer, J. et al., 2015. *An autonomous multi-UAV system for search and rescue.* New York, Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use, pp. 35-38.
107. Shireen, B., 2018. *DRONE REPORT,* Sydney: North Sydney Innovation Network (NSIN).
108. Silvagni, M., Tonoli, A., Zenerino, E. & Chiaberge, M., 2017. Multipurpose UAV for search and rescue operations in mountain avalanche events. *Geomatics, Natural Hazards and Risk,* 8(1), pp. 18-33.
109. SLANTRANGE, 2018. *The slantrange 3p multispectral sensor.* [Online] Available at: http://www.slantrange.com/3p-slantview-available/ [Accessed 1 December 2019].
110. Smith, M. L., 2015. Regulating law enforcement's use of drones: The need for state legislation. *Harv. J. on Legis.,* 52(1), p. 423.
111. Steve, H. & Kevin, E., 2018. *Mapping tile drainage systems.* [Online] Available at: https://fyi.uwex.edu/drainage/files/2016/01/1603-Hoffman-System-for-Mapping-Tile.pdf [Accessed 8 December 2019].
112. Sullivan, D. et al., 2004. Evaluation of multispectral data for rapid assessment of wheat straw residue cover. *Soil Science Society of America Journal,* 68(6), p. 2007–2013.
113. Sun, J., Li, B., Jiang, Y. & Wen, C.-y., 2016. A camera-based target detection and positioning UAV system for search and rescue (sar) purposes. *Sensors ,* 16(11), p. 1778.
114. Swain, K. C., Thomson, S. J. & Jayasuriya, H. P., 2010. Adoption of an unmanned helicopter for low-altitude remote sensing to estimate yield and total biomass of a rice crop. *Transactions of the ASABE,* 53(1), p. 21–27.
115. Thornton, J. et al., 2001. Broadband communications from a high-altitude platform: the european helinet programme. *Electronics & Communication Engineering Journal,* 13(3), p. 138–144.
116. Tozer, T. & Grace, D., 2001. High-altitude platforms for wireless communications. *Electronics & Communication Engineering Journal,* 13(3), p. 127–137.
117. U. D. of Interior, 2018. *Mapping crop residue and tillage intensity on chesapeake bay farmland.* [Online] Available at: https://eros.usgs.gov/doi-remote-sensing-activities/2015/mapping-crop-residue-and-tillage-intensity-chesapeake-bay-farmland [Accessed 8 December 2019].
118. Valcarce, A. et al., 2014. *Airborne base stations for emergency and temporary events.* Toulouse, International Conference on Personal Satellite Services, Springer, p. 13–25.
119. Vergouw, B., Nagel, H., Bondt, G. & Custers, B., 2016. *Drone technology: Types, payloads, applications, frequency spectrum issues and future developments.* Heidelberg, Springer, p. 21–45..

120. Villa, T. F. et al., 2016. An overview of small unmanned aerial vehicles for air quality measurements: Present applications and future prospectives. *Sensors,* 16(7), p. 1072.
121. Wang, D.-C.et al., 2015. Retrieval and mapping of soil texture based on land surface diurnal temperature range data from modis. *PloS one,* 10(6), p. 0129977.
122. Whitehead, K. & Hugenholtz, C. H., 2014. Remote sensing of the environment with small unmanned aircraft systems (uass), part 1: a review of progress and challenges. *Journal of Unmanned Vehicle Systems,* 2(3), p. 69–85.
123. Whitehead, K., Moorman, B. & Hugenholtz, C., 2013. Low-cost, on-demand aerial photogrammetry for glaciological measurement. *Cryosphere Discussions,* 7(3), pp. 1879 - 1884.
124. Wikipedia, 2018. *Uncrewed-vehicle Wikipedia, the free encyclopedia.* [Online] Available at: https://en.wikipedia.org/wiki/Uncrewed-vehicle [Accessed 6 December 2019].
125. Yang, G. et al., 2017. Unmanned aerial vehicle remote sensing for field-based crop phenotyping: Current status and perspectives. *Frontiers in plant science,* 8(1), p. 1111.
126. Yasmina, B. S., 2015. *Smart Autonomous Aircraft: Flight Control and Planning for UAV,* Florida: CRC Press.
127. Young, L. A. & Johnson, J. L., 1999. *Tilt Rotor Aeroacoustic Model Project,* Rome: Confederation of European Aerospace Societies (CEAS) Forum on Aeroacoustics of Rotors and Propellers.
128. Young, L. A., Lillie, D., McCluer, M. & Yamauchi, G. K., 2002. *Insights into Airframe Aerodynamics and Rotor-on-Wing Interactions from a 0.25-Scale Tiltrotor Wind Tunnel Model,* California: Army/NASA Rotorcraft Division, NASA Ames Research Center.
129. Zhang, C. & Kovacs, J. M., 2012. The application of small unmanned aerial systems for precision agriculture: a review. *Precision agriculture,* 13(6), p. 693–712.
130. Zul Azfar, A. & Hazry, D., March, 2011. *Simple GUI Design for Moni-toring of a Remotely Operated Quadcopter Unmanned Aerial Vehicle.* Penang, Proceeding of the 7th International Colloquium on Signal Processing and its Applications (CSPA).

APPENDICES

APPENDIX A: PYTHON PROGRAM FOR MICRO DRONE

```
# -*- coding: utf-8 -*-
"""
Created on Tue Nov 19 15:58:25 2019

@author: Inyeni Showers
"""

from djitellopy import Tello
import time
import cv2
import sys
import numpy as np



tello = Tello()


speed = 10
width = 950
height = 680
distance = 110

if not tello.connect():
    print("Tello not connected")
    sys.exit()

if not tello.set_speed(speed):
    print("Not set speed to lowest possible")
    sys.exit()

# In case streaming is on. This happens when we quit this program without the escape key.
if not tello.streamoff():
    print("Could not stop video stream")
    sys.exit()

if not tello.streamon():
    print("Could not start video stream")
    sys.exit()

#tello waits before takeoff

print ("Current battery is " + tello.get_battery())
tello.set_speed(10)
delay=16.2
close_time=time.time() + delay
while True:
    print ("Current battery is " + tello.get_battery())
    time.sleep(3)
    if time.time()>close_time:
```

```
        break

#tello mission path
tello.takeoff()
tello.set_speed(40)
tello.rotate_clockwise(180)
tello.move_forward(90)
time.sleep(0.5)
tello.rotate_clockwise(90)
time.sleep(0.5)
tello.move_forward(59)
time.sleep(0.5)
tello.rotate_clockwise(90)
time.sleep(0.5)
tello.move_down(55)
time.sleep(2.5)
tello.move_forward(78)
time.sleep(0.5)


#tello tracking variables
midx = int(width / 2)
midy = int(height / 2)
xoffset = 0
yoffset = 0
frame_read = tello.get_frame_read()

turn = 0
radius = 1
decide = 0


#tello tracking 1

while True:

    frame = frame_read.frame
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    l_b = np.array([15, 146, 150])
    u_b = np.array([68, 255, 255])

    cv2.arrowedLine(frame, (midx, midy),
                    (midx + xoffset, midy - yoffset),
                    (0, 0, 255), 5)

    """Simple HSV color space tracking"""
        # resize the frame, blur it, and convert it to the HSV
        # color space
    blurred = cv2.GaussianBlur(frame, (11, 11), 0)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

    # construct a mask for the color then perform
    # a series of dilations and erosions to remove any small
    # blobs left in the mask
    mask = cv2.inRange(hsv, l_b, u_b)
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)
```

```
    # find contours in the mask and initialize the current
    # (x, y) center of the ball
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
                        cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0]
center = None

# only proceed if at least one contour was found
if len(cnts) > 0:
    # find the largest contour in the mask, then use
    # it to compute the minimum enclosing circle and
    # centroid
    c = max(cnts, key=cv2.contourArea)
    ((x, y), radius) = cv2.minEnclosingCircle(c)
    M = cv2.moments(c)
    center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

    # only proceed if the radius meets a minimum size
    if radius > 10:
        # draw the circle and centroid on the frame,
        # then update the list of tracked points
        cv2.circle(frame, (int(x), int(y)), int(radius),
                   (0, 255, 255), 2)
        cv2.circle(frame, center, 5, (0, 0, 255), -1)

        xoffset = int(center[0] - midx)
        yoffset = int( midy - center[1])
        decide = 1

        if xoffset < -distance:
            tello.rotate_counter_clockwise(8)

        elif xoffset > distance:
            tello.rotate_clockwise(8)

    else:
        xoffset = 0
        yoffset = 0
        tello.get_battery()


else:
    xoffset = 0
    yoffset = 0
    tello.get_height()
    if turn == 0 and decide == 0:
        tello.rotate_clockwise(40)
        turn = 1
        time.sleep(1)
    elif turn == 1 and decide == 0:
        tello.rotate_counter_clockwise(40)
        turn = 2
        time.sleep(1)
    elif turn == 2 and decide == 0:
        tello.rotate_counter_clockwise(40)
        turn = 3
```

```
            time.sleep(1)
        elif turn == 3 and decide == 0:
            tello.rotate_clockwise(40)
            turn = 0
            time.sleep(1)
        elif decide == 1:
            time.sleep(0.5)


    mask = cv2.inRange(hsv, l_b, u_b)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow("mask", mask)


    cv2.imshow("Video", frame)
    key = cv2.waitKey(1)
    if radius > 37 and xoffset > -distance and xoffset < distance :
        print("exit tracking mode 1, end tracking radius is", radius)
        break


#tello tracking variables

midx = int(width / 2)
midy = int(height / 2)
xoffset = 0
yoffset = 0
frame_read = tello.get_frame_read()

turn = 0
radius = 1
decide = 0
delay1 = 7
close_time1 = time.time()+delay1

#tello tracking 2

while True:

    frame = frame_read.frame
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    l_b = np.array([15, 146, 150])
    u_b = np.array([68, 255, 255])

    cv2.arrowedLine(frame, (midx, midy),
                    (midx + xoffset, midy - yoffset),
                    (0, 0, 255), 5)

    """Simple HSV color space tracking"""
        # resize the frame, blur it, and convert it to the HSV
        # color space
    blurred = cv2.GaussianBlur(frame, (11, 11), 0)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

    # construct a mask for the color then perform
    # a series of dilations and erosions to remove any small
    # blobs left in the mask
```

```
mask = cv2.inRange(hsv, l_b, u_b)
mask = cv2.erode(mask, None, iterations=2)
mask = cv2.dilate(mask, None, iterations=2)

    # find contours in the mask and initialize the current
    # (x, y) center of the ball
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
                        cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0]
center = None

# only proceed if at least one contour was found
if len(cnts) > 0:
    # find the largest contour in the mask, then use
    # it to compute the minimum enclosing circle and
    # centroid
    c = max(cnts, key=cv2.contourArea)
    ((x, y), radius) = cv2.minEnclosingCircle(c)
    M = cv2.moments(c)
    center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

    # only proceed if the radius meets a minimum size
    if radius > 10:
        # draw the circle and centroid on the frame,
        # then update the list of tracked points
        cv2.circle(frame, (int(x), int(y)), int(radius),
                   (0, 255, 255), 2)
        cv2.circle(frame, center, 5, (0, 0, 255), -1)

        xoffset = int(center[0] - midx)
        yoffset = int( midy - center[1])
        decide = 1

        if xoffset < -distance:
            tello.rotate_counter_clockwise(7)

        elif xoffset > distance:
            tello.rotate_clockwise(7)

    else:
        xoffset = 0
        yoffset = 0
        tello.get_battery()


else:
    xoffset = 0
    yoffset = 0
    tello.get_height()
    if turn == 0 and decide == 0:
        tello.rotate_clockwise(40)
        turn = 1
        time.sleep(1)
    elif turn == 1 and decide == 0:
        tello.rotate_counter_clockwise(40)
        turn = 2
        time.sleep(1)
```

```
        elif turn == 2 and decide == 0:
            tello.rotate_counter_clockwise(40)
            turn = 3
            time.sleep(1)
        elif turn == 3 and decide == 0:
            tello.rotate_clockwise(40)
            turn = 0
            time.sleep(1)
        elif decide == 1:
            time.sleep(0.5)


    mask = cv2.inRange(hsv, l_b, u_b)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow("mask", mask)


    cv2.imshow("Video", frame)
    key = cv2.waitKey(1)
    if time.time() > close_time1 and xoffset > -distance and xoffset < distance :
        print("exit tracking mode 2, end tracking radius is", radius)
        break



#preparing to land
delay2 = 2
close_time2 =time.time() + delay2
while True:
    print ("Current battery is " + tello.get_battery())
    time.sleep(1)
    if time.time()>close_time2:
        break

print("ready to land!")
print('landing')
tello.set_speed(20)
tello.move_up(40)
time.sleep(0.5)
tello.move_down(40)
time.sleep(0.5)
delay3 = 6.9
close_time3 =time.time() + delay3
while True:
    print ("Current battery is " + tello.get_battery())
    time.sleep(1)
    if time.time()>close_time3:
        break

#tello land

tello.land()
time.sleep(0.5)
tello.streamoff()
cv2.destroyAllWindows()
```

APPENDIX B: DJI TELLO REPOSITORY

```
# coding=utf-8
import socket
import time
import threading
import cv2
from threading import Thread
from djitellopy.decorators import accepts


class Tello:
    """Python wrapper to interact with the Ryze Tello drone using the official Tello api.
    Tello API documentation:
    https://dl-
cdn.ryzerobotics.com/downloads/tello/20180910/Tello%20SDK%20Documentation%20EN_1
.3.pdf
    """
    # Send and receive commands, client socket
    UDP_IP = '192.168.10.1'
    UDP_PORT = 8889
    RESPONSE_TIMEOUT = 10  # in seconds
    TIME_BTW_COMMANDS = 5  # in seconds
    TIME_BTW_RC_CONTROL_COMMANDS = 2  # in seconds
    last_received_command = time.time()

    # Video stream, server socket
    VS_UDP_IP = '0.0.0.0'
    VS_UDP_PORT = 11111

    # VideoCapture object
    cap = None
    background_frame_read = None

    stream_on = False

    def __init__(self):
        # To send comments
        self.address = (self.UDP_IP, self.UDP_PORT)
        self.clientSocket = socket.socket(socket.AF_INET,  # Internet
                                          socket.SOCK_DGRAM)  # UDP
        self.clientSocket.bind(('', self.UDP_PORT))  # For UDP response (receiving data)
        self.response = None
        self.stream_on = False

        # Run tello udp receiver on background
        thread = threading.Thread(target=self.run_udp_receiver, args=())
        thread.daemon = True
        thread.start()

    def run_udp_receiver(self):
        """Setup drone UDP receiver. This method listens for responses of Tello. Must be run
from a background thread
        in order to not block the main thread."""
        while True:
            try:
                self.response, _ = self.clientSocket.recvfrom(1024)  # buffer size is 1024 bytes
```

```
            except Exception as e:
                print(e)
                break

    def get_udp_video_address(self):
        return 'udp://@' + self.VS_UDP_IP + ':' + str(self.VS_UDP_PORT)  # +
'?overrun_nonfatal=1&fifo_size=5000'

    def get_video_capture(self):
        """Get the VideoCapture object from the camera drone
        Returns:
            VideoCapture
        """

        if self.cap is None:
            self.cap = cv2.VideoCapture(self.get_udp_video_address())

        if not self.cap.isOpened():
            self.cap.open(self.get_udp_video_address())

        return self.cap

    def get_frame_read(self):
        """Get the BackgroundFrameRead object from the camera drone. Then, you just need
to call
        backgroundFrameRead.frame to get the actual frame received by the drone.
        Returns:
            BackgroundFrameRead
        """
        if self.background_frame_read is None:
            self.background_frame_read = BackgroundFrameRead(self,
self.get_udp_video_address()).start()
        return self.background_frame_read

    def stop_video_capture(self):
        return self.streamoff()

    @accepts(command=str)
    def send_command_with_return(self, command):
        """Send command to Tello and wait for its response.
        Return:
            bool: True for successful, False for unsuccessful
        """
        # Commands very consecutive makes the drone not respond to them. So wait at least
self.TIME_BTW_COMMANDS seconds
        diff = time.time() * 1000 - self.last_received_command
        if diff < self.TIME_BTW_COMMANDS:
            time.sleep(diff)

        print('Send command: ' + command)
        timestamp = int(time.time() * 1000)

        self.clientSocket.sendto(command.encode('utf-8'), self.address)

        while self.response is None:
            if (time.time() * 1000) - timestamp > self.RESPONSE_TIMEOUT * 1000:
                print('Timeout exceed on command ' + command)
```

```
                return False

        print('Response: ' + str(self.response))

        response = self.response.decode('utf-8')

        self.response = None

        self.last_received_command = time.time() * 1000

        return response

    @accepts(command=str)
    def send_command_without_return(self, command):
        """Send command to Tello without expecting a response. Use this method when you
want to send a command
        continuously
            - go x y z speed: Tello fly to x y z in speed (cm/s)
                x: 20-500
                y: 20-500
                z: 20-500
                speed: 10-100
            - curve x1 y1 z1 x2 y2 z2 speed: Tello fly a curve defined by the current and two
given coordinates with
                speed (cm/s). If the arc radius is not within the range of 0.5-10 meters, it responses
false.
                x/y/z can't be between -20 – 20 at the same time .
                x1, x2: 20-500
                y1, y2: 20-500
                z1, z2: 20-500
                speed: 10-60
            - rc a b c d: Send RC control via four channels.
                a: left/right (-100~100)
                b: forward/backward (-100~100)
                c: up/down (-100~100)
                d: yaw (-100~100)
        """
        # Commands very consecutive makes the drone not respond to them. So wait at least
self.TIME_BTW_COMMANDS seconds

        print('Send command (no expect response): ' + command)
        self.clientSocket.sendto(command.encode('utf-8'), self.address)

    @accepts(command=str)
    def send_control_command(self, command):
        """Send control command to Tello and wait for its response. Possible control
commands:
            - command: entry SDK mode
            - takeoff: Tello auto takeoff
            - land: Tello auto land
            - streamon: Set video stream on
            - streamoff: Set video stream off
            - emergency: Stop all motors immediately
            - up x: Tello fly up with distance x cm. x: 20-500
            - down x: Tello fly down with distance x cm. x: 20-500
            - left x: Tello fly left with distance x cm. x: 20-500
            - right x: Tello fly right with distance x cm. x: 20-500
```

```
        - forward x: Tello fly forward with distance x cm. x: 20-500
        - back x: Tello fly back with distance x cm. x: 20-500
        - cw x: Tello rotate x degree clockwise x: 1-3600
        - ccw x: Tello rotate x degree counter- clockwise. x: 1-3600
        - flip x: Tello fly flip x
            l (left)
            r (right)
            f (forward)
            b (back)
        - speed x: set speed to x cm/s. x: 10-100
        - wifi ssid pass: Set Wi-Fi with SSID password

    Return:
        bool: True for successful, False for unsuccessful
    """

    response = self.send_command_with_return(command)

    if response == 'OK' or response == 'ok':
        return True
    else:
        return self.return_error_on_send_command(command, response)

@accepts(command=str)
def send_read_command(self, command):
    """Send set command to Tello and wait for its response. Possible set commands:
        - speed?: get current speed (cm/s): x: 1-100
        - battery?: get current battery percentage: x: 0-100
        - time?: get current fly time (s): time
        - height?: get height (cm): x: 0-3000
        - temp?: get temperature (°C): x: 0-90
        - attitude?: get IMU attitude data: pitch roll yaw
        - baro?: get barometer value (m): x
        - tof?: get distance value from TOF (cm): x: 30-1000
        - wifi?: get Wi-Fi SNR: snr

    Return:
        bool: True for successful, False for unsuccessful
    """

    response = self.send_command_with_return(command)

    try:
        response = str(response)
    except TypeError as e:
        print(e)
        pass

    if ('error' not in response) and ('ERROR' not in response) and ('False' not in response):
        if response.isdigit():
            return int(response)
        else:
            return response
    else:
        return self.return_error_on_send_command(command, response)

@staticmethod
```

```
    def return_error_on_send_command(command, response):
        """Returns False and print an informative result code to show unsuccessful response"""
        print('Command ' + command + ' was unsuccessful. Message: ' + str(response))
        return False

    def connect(self):
        """Entry SDK mode
        Returns:
            bool: True for successful, False for unsuccessful
        """
        return self.send_control_command("command")

    def takeoff(self):
        """Tello auto takeoff
        Returns:
            bool: True for successful, False for unsuccessful
            False: Unsuccessful
        """
        return self.send_control_command("takeoff")

    def land(self):
        """Tello auto land
        Returns:
            bool: True for successful, False for unsuccessful
        """
        return self.send_control_command("land")

    def palm_land(self):

        """Tell drone to land"""
        return self.send_control_command("palmland")

    def streamon(self):
        """Set video stream on. If the response is 'Unknown command' means you have to
update the Tello firmware. That
        can be done through the Tello app.
        Returns:
            bool: True for successful, False for unsuccessful
        """
        result = self.send_control_command("streamon")
        if result is True:
            self.stream_on = True
        return result

    def streamoff(self):
        """Set video stream off
        Returns:
            bool: True for successful, False for unsuccessful
        """
        result = self.send_control_command("streamoff")
        if result is True:
            self.stream_on = False
        return result

    def emergency(self):
        """Stop all motors immediately
        Returns:
```

```
        bool: True for successful, False for unsuccessful
    """
    return self.send_control_command("emergency")

@accepts(direction=str, x=int)
def move(self, direction, x):
    """Tello fly up, down, left, right, forward or back with distance x cm.
    Arguments:
        direction: up, down, left, right, forward or back
        x: 20-500

    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.send_control_command(direction + ' ' + str(x))

@accepts(x=int)
def move_up(self, x):
    """Tello fly up with distance x cm.
    Arguments:
        x: 20-500

    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.move("up", x)


@accepts(x=int)
def move_down(self, x):
    """Tello fly down with distance x cm.
    Arguments:
        x: 20-500

    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.move("down", x)

@accepts(x=int)
def move_left(self, x):
    """Tello fly left with distance x cm.
    Arguments:
        x: 20-500

    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.move("left", x)

@accepts(x=int)
def move_right(self, x):
    """Tello fly right with distance x cm.
    Arguments:
        x: 20-500

    Returns:
```

```
        bool: True for successful, False for unsuccessful
    """
    return self.move("right", x)

@accepts(x=int)
def move_forward(self, x):
    """Tello fly forward with distance x cm.
    Arguments:
        x: 20-500

    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.move("forward", x)

@accepts(x=int)
def move_back(self, x):
    """Tello fly back with distance x cm.
    Arguments:
        x: 20-500

    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.move("back", x)

@accepts(x=int)
def move_up(self, x):
    """Tello fly up with distance x cm.
    Arguments:
        x: 20-500

    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.move("up", x)

@accepts(x=int)
def rotate_clockwise(self, x):
    """Tello rotate x degree clockwise.
    Arguments:
        x: 1-360

    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.send_control_command("cw " + str(x))

@accepts(x=int)
def rotate_counter_clockwise(self, x):
    """Tello rotate x degree counter-clockwise.
    Arguments:
        x: 1-3600

    Returns:
        bool: True for successful, False for unsuccessful
    """
```

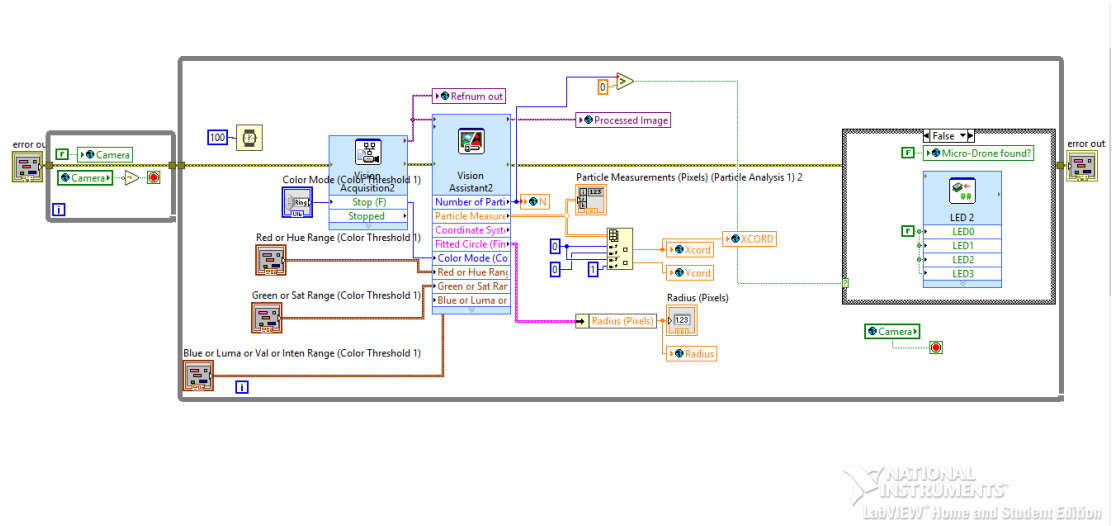
```
    return self.send_control_command("ccw " + str(x))

@accepts(x=str)
def flip(self, direction):
    """Tello fly flip.
    Arguments:
        direction: l (left), r (right), f (forward) or b (back)

    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.send_control_command("flip " + direction)

def flip_left(self):
    """Tello fly flip left.
    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.flip("l")

def flip_right(self):
    """Tello fly flip left.
    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.flip("r")

def flip_forward(self):
    """Tello fly flip left.
    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.flip("f")

def flip_back(self):
    """Tello fly flip left.
    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.flip("b")

@accepts(x=int, y=int, z=int, speed=int)
def go_xyz_speed(self, x, y, z, speed):
    """Tello fly to x y z in speed (cm/s)
    Arguments:
        x: 20-500
        y: 20-500
        z: 20-500
        speed: 10-100
    Returns:
        bool: True for successful, False for unsuccessful
    """
    return self.send_command_without_return('go %s %s %s %s' % (x, y, z, speed))

@accepts(x1=int, y1=int, z1=int, x2=int, y2=int, z2=int, speed=int)
def go_xyz_speed1(self, x1, y1, z1, x2, y2, z2, speed):
    """Tello fly a curve defined by the current and two given coordinates with speed (cm/s).
```

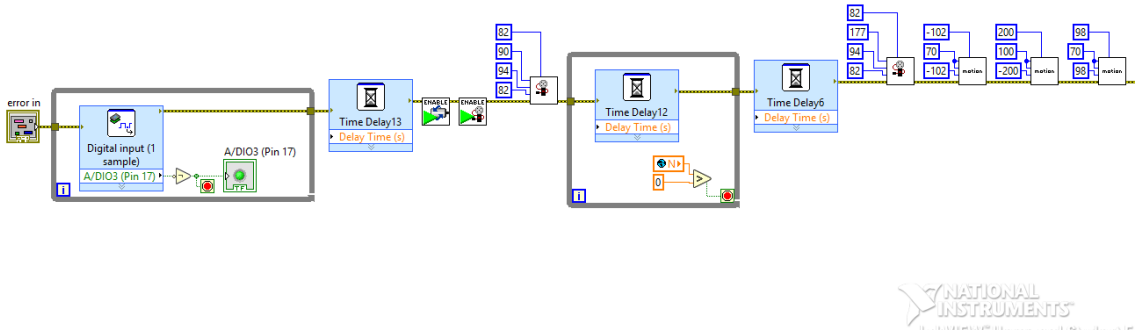
```
            - If the arc radius is not within the range of 0.5-10 meters, it responses false.
            - x/y/z can't be between -20 – 20 at the same time.
        Arguments:
            x1: 20-500
            x2: 20-500
            y1: 20-500
            y2: 20-500
            z1: 20-500
            z2: 20-500
            speed: 10-60
        Returns:
            bool: True for successful, False for unsuccessful
        """
        return self.send_command_without_return('curve %s %s %s %s %s %s %s' % (x1, y1,
z1, x2, y2, z2, speed))

    @accepts(x=int)
    def set_speed(self, x):
        """Set speed to x cm/s.
        Arguments:
            x: 10-100

        Returns:
            bool: True for successful, False for unsuccessful
        """
        return self.send_control_command("speed " + str(x))

    last_rc_control_sent = 0

    @accepts(left_right_velocity=int, forward_backward_velocity=int, up_down_velocity=int,
yaw_velocity=int)
    def send_rc_control(self, left_right_velocity, forward_backward_velocity,
up_down_velocity, yaw_velocity):
        """Send RC control via four channels. Command is sent every
self.TIME_BTW_RC_CONTROL_COMMANDS seconds.
        Arguments:
            left_right_velocity: -100~100 (left/right)
            forward_backward_velocity: -100~100 (forward/backward)
            up_down_velocity: -100~100 (up/down)
            yaw_velocity: -100~100 (yaw)
        Returns:
            bool: True for successful, False for unsuccessful
        """
        if int(time.time() * 1000) - self.last_rc_control_sent <
self.TIME_BTW_RC_CONTROL_COMMANDS:
            pass
        else:
            self.last_rc_control_sent = int(time.time() * 1000)
            return self.send_command_without_return('rc %s %s %s %s' % (left_right_velocity,
forward_backward_velocity,
                                                        up_down_velocity, yaw_velocity))

    def set_wifi_with_ssid_password(self):
        """Set Wi-Fi with SSID password.
        Returns:
            bool: True for successful, False for unsuccessful
        """
```

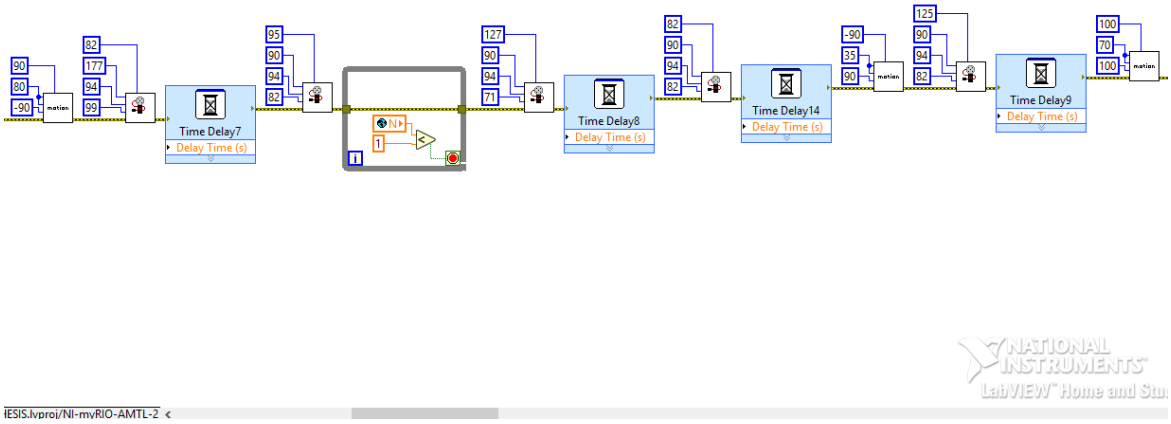
```
        return self.send_control_command('wifi ssid pass')

    def get_speed(self):
        """Get current speed (cm/s)
        Returns:
            False: Unsuccessful
            int: 1-100
        """
        return self.send_read_command('speed?')

    def get_battery(self):
        """Get current battery percentage
        Returns:
            False: Unsuccessful
            int: -100
        """
        return self.send_read_command('battery?')

    def get_flight_time(self):
        """Get current fly time (s)
        Returns:
            False: Unsuccessful
            int: Seconds elapsed during flight.
        """
        return self.send_read_command('time?')

    def get_height(self):
        """Get height (cm)
        Returns:
            False: Unsuccessful
            int: 0-3000
        """
        return self.send_read_command('height?')

    def get_temperature(self):
        """Get temperature (°C)
        Returns:
            False: Unsuccessful
            int: 0-90
        """
        return self.send_read_command('temperature?')

    def get_attitude(self):
        """Get IMU attitude data
        Returns:
            False: Unsuccessful
            int: pitch roll yaw
        """
        return self.send_read_command('attitude?')

    def get_barometer(self):
        """Get barometer value (m)
        Returns:
            False: Unsuccessful
            int: 0-100
        """
        return self.send_read_command('baro?')
```

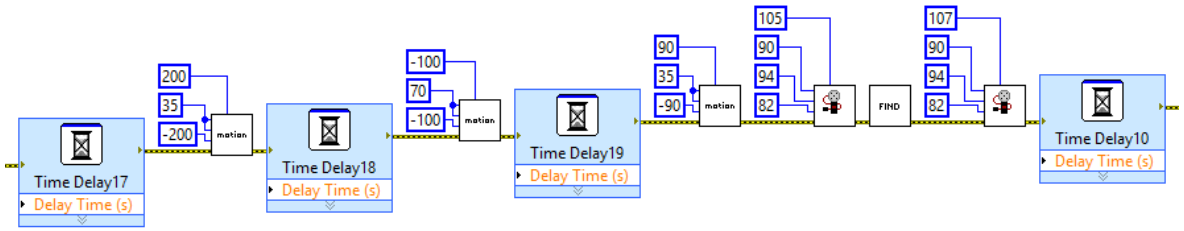
```
    def get_distance_tof(self):
        """Get distance value from TOF (cm)
        Returns:
            False: Unsuccessful
            int: 30-1000
        """
        return self.send_read_command('tof?')

    def get_wifi(self):
        """Get Wi-Fi SNR
        Returns:
            False: Unsuccessful
            str: snr
        """
        return self.send_read_command('wifi?')

    def end(self):
        """Call this method when you want to end the tello object"""
        if self.stream_on:
            self.streamoff()
        if self.background_frame_read is not None:
            self.background_frame_read.stop()
        if self.cap is not None:
            self.cap.release()


class BackgroundFrameRead:
    """
    This class read frames from a VideoCapture in background. Then, just call backgroundFrameRead.frame to get the
    actual one.
    """

    def __init__(self, tello, address):
        tello.cap = cv2.VideoCapture(address)
        self.cap = tello.cap

        if not self.cap.isOpened():
            self.cap.open(address)

        self.grabbed, self.frame = self.cap.read()
        self.stopped = False

    def start(self):
        Thread(target=self.update_frame, args=()).start()
        return self

    def update_frame(self):
        while not self.stopped:
            if not self.grabbed or not self.cap.isOpened():
                self.stop()
            else:
                (self.grabbed, self.frame) = self.cap.read()

    def stop(self):
        self.stopped = True
```

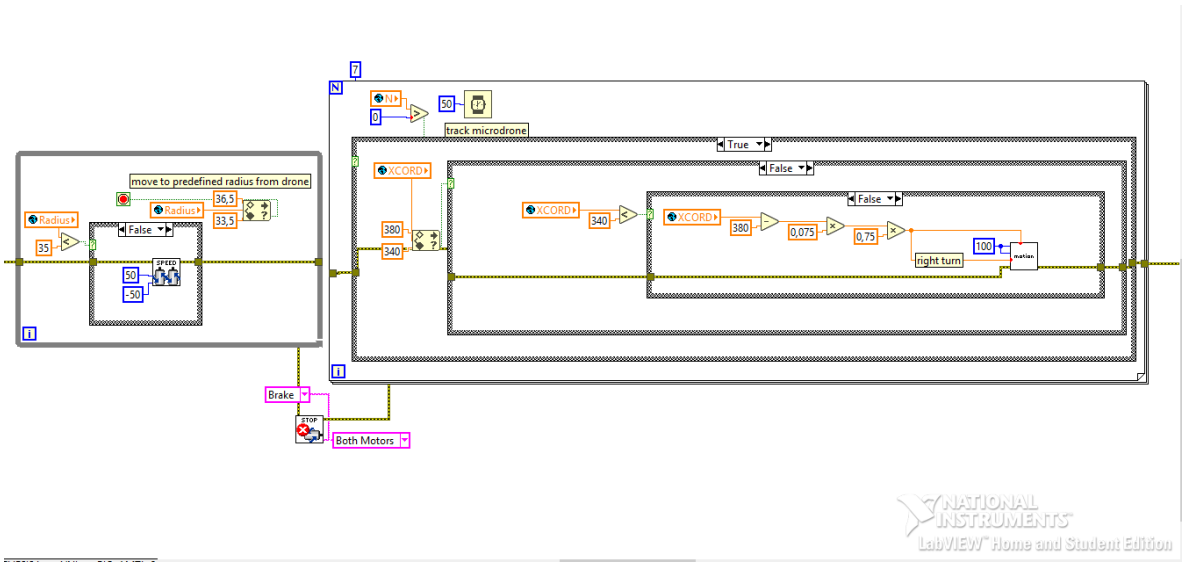
APPENDIX C: LABVIEW PROGRAM FOR GROUND STATION

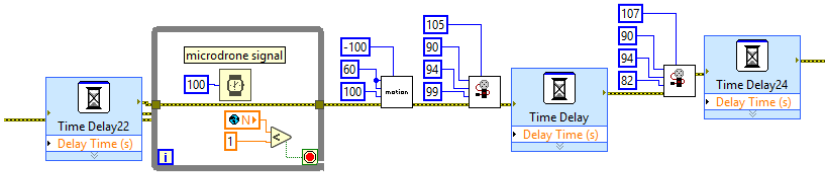


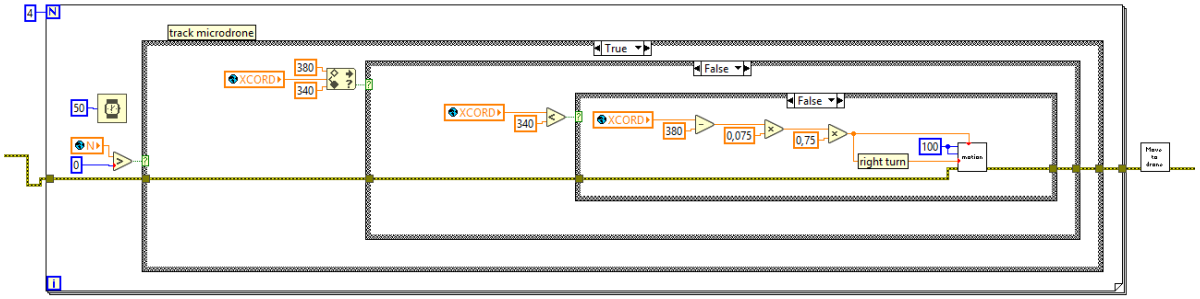


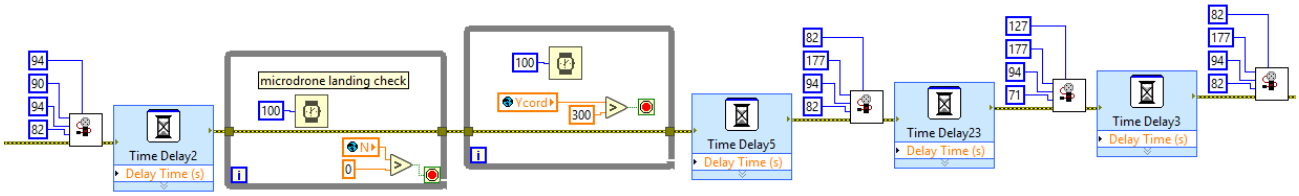


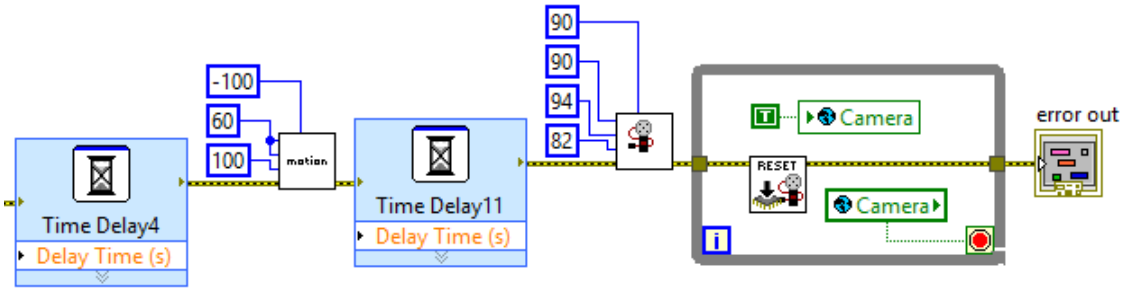

Time Delay17
Delay Time (s)
200
35
-200
motion
Time Delay18
Delay Time (s)
-100
70
-100
motion
Time Delay19
Delay Time (s)
90
35
-90
motion
105
90
94
82
FIND
107
90
94
82
Time Delay10
Delay Time (s)


move to predefined radius from drone
Radius
36,5
33,5
35
False
SPEED
50
-50
track microdrone
True
False
XCORD
380
340
0,075
0,75
right turn
100
Brake
STOP
Both Motors
NATIONAL INSTRUMENTS
LabVIEW Home and Student Edition


Time Delay22
Delay Time (s)
microdrone signal
100
-100
60
100
motion
105
90
94
99
Time Delay
Delay Time (s)
107
90
94
82
Time Delay24
Delay Time (s)


track microdrone
True
False
XCORD
380
340
50
0
0,075
0,75
right turn
100


94
90
94
82
Time Delay2
Delay Time (s)
microdrone landing check
100
0
100
Vcord
300
Time Delay5
Delay Time (s)
82
177
94
82
Time Delay23
Delay Time (s)
127
177
94
71
Time Delay3
Delay Time (s)
82
177
94
82


Time Delay4
Delay Time (s)
-100
60
100
motion
Time Delay11
Delay Time (s)
90
90
94
82
Camera
RESET
Camera
error out